In [61]:
```python
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 [62]:
```python
import tensorflow as tf
from tensorflow import keras
```

In [63]:
```python
# Import required modules
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
```

In [64]:
```python
from tensorflow.keras import layers
```

In [65]:
```python
# 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 [66]:
```python
# Create empty lists for the images and labels
images = []
labels = []

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

In [67]:
```python
# 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 [68]:
```python
# 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 [69]:
```python
# Convert the lists to numpy arrays
X = np.array(images)
y = np.array(labels)
```

```
C:\Users\arivu\AppData\Local\Temp\ipykernel_37112\1953660164.py:2: VisibleDeprecat
ionWarning: Creating an ndarray from ragged nested sequences (which is a list-or-t
uple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecate
d. If you meant to do this, you must specify 'dtype=object' when creating the ndar
ray.
  X = np.array(images)
```

In [70]:
```python
# One-hot encode the labels
encoder = OneHotEncoder(sparse=False)
y = encoder.fit_transform(y.reshape(-1, 1))
```

In [11]:
```
# Create empty list for the images
defective_images = []
non_defective_images = []
```

In [12]:
```
# 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)
    img = cv2.resize(img, (224, 224))  # optional: resize the image to a fixed size
    img = img.astype('float32') / 255.0  # normalize the pixel values to the range
    defective_images.append(img)
```

In [13]:
```
# 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)
    img = cv2.resize(img, (224, 224))  # optional: resize the image to a fixed size
    img = img.astype('float32') / 255.0  # normalize the pixel values to the range
    non_defective_images.append(img)
```

In [14]:
```
# Convert the lists to numpy arrays
X_defective = np.array(defective_images)
X_non_defective = np.array(non_defective_images)
```

In [15]:
```
print(X_defective.shape)
print(X_non_defective.shape)
```

```
(756, 224, 224)
(756, 224, 224)
```

In [72]:
```
# Split the data into training and testing sets
#X_train, X_test, y_train, y_test = train_test_split(X_defective, X_non_defective,
```

In [73]:
```
# 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 [76]:
```
print('X_train: ',X_train.shape)
print('X_test: ',X_test.shape)
print('y_train: ',y_train.shape)
print('y_test: ',y_test.shape)
```

```
X_train:  (1209,)
X_test:  (303,)
y_train:  (1209, 2)
y_test:  (303, 2)
```

In [ ]:

In [20]:
```
import numpy as np
from sklearn.neural_network import BernoulliRBM
from sklearn.pipeline import Pipeline
```

In [ ]:
```
# Train a DBN model
rbm1 = BernoulliRBM(n_components=100, learning_rate=0.01, n_iter=100, random_state
rbm2 = BernoulliRBM(n_components=50, learning_rate=0.01, n_iter=100, random_state=
classifier = Pipeline(steps=[('rbm1', rbm1), ('rbm2', rbm2)])
```

In [22]:
```
X_train1 = np.reshape(X_train, (X_train.shape[0], -1))
y_train1 = np.reshape(y_train, (y_train.shape[0], -1))
```

```python
X_test1 = np.reshape(X_test, (X_test.shape[0], -1))
y_test1 = np.reshape(y_test, (y_test.shape[0], -1))
```

In [24]:

```python
classifier.fit(X_train1, y_train1)
```

```
[BernoulliRBM] Iteration 1, pseudo-likelihood = -32403.03, time = 4.08s
[BernoulliRBM] Iteration 2, pseudo-likelihood = -30507.68, time = 4.67s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -28947.84, time = 4.80s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -27632.40, time = 4.25s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -26514.76, time = 4.36s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -25570.20, time = 4.41s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -24741.63, time = 4.90s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -24020.96, time = 4.91s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -23422.70, time = 4.75s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -22912.06, time = 4.43s
[BernoulliRBM] Iteration 11, pseudo-likelihood = -22431.82, time = 4.53s
[BernoulliRBM] Iteration 12, pseudo-likelihood = -22048.02, time = 4.45s
[BernoulliRBM] Iteration 13, pseudo-likelihood = -21664.85, time = 5.14s
[BernoulliRBM] Iteration 14, pseudo-likelihood = -21352.40, time = 4.71s
[BernoulliRBM] Iteration 15, pseudo-likelihood = -21066.64, time = 5.15s
[BernoulliRBM] Iteration 16, pseudo-likelihood = -20750.54, time = 5.49s
[BernoulliRBM] Iteration 17, pseudo-likelihood = -20609.35, time = 5.03s
[BernoulliRBM] Iteration 18, pseudo-likelihood = -20314.56, time = 4.99s
[BernoulliRBM] Iteration 19, pseudo-likelihood = -20158.48, time = 4.59s
[BernoulliRBM] Iteration 20, pseudo-likelihood = -20137.22, time = 4.56s
[BernoulliRBM] Iteration 21, pseudo-likelihood = -19805.59, time = 4.44s
[BernoulliRBM] Iteration 22, pseudo-likelihood = -19708.31, time = 4.43s
[BernoulliRBM] Iteration 23, pseudo-likelihood = -19595.10, time = 4.66s
[BernoulliRBM] Iteration 24, pseudo-likelihood = -19473.64, time = 4.31s
[BernoulliRBM] Iteration 25, pseudo-likelihood = -19393.37, time = 4.54s
[BernoulliRBM] Iteration 26, pseudo-likelihood = -19331.41, time = 4.46s
[BernoulliRBM] Iteration 27, pseudo-likelihood = -19221.70, time = 4.44s
[BernoulliRBM] Iteration 28, pseudo-likelihood = -19145.14, time = 4.83s
[BernoulliRBM] Iteration 29, pseudo-likelihood = -19019.12, time = 4.40s
[BernoulliRBM] Iteration 30, pseudo-likelihood = -18938.31, time = 4.26s
[BernoulliRBM] Iteration 31, pseudo-likelihood = -18821.91, time = 4.34s
[BernoulliRBM] Iteration 32, pseudo-likelihood = -18816.80, time = 4.31s
[BernoulliRBM] Iteration 33, pseudo-likelihood = -18700.67, time = 4.48s
[BernoulliRBM] Iteration 34, pseudo-likelihood = -18944.43, time = 4.27s
[BernoulliRBM] Iteration 35, pseudo-likelihood = -18815.19, time = 4.36s
[BernoulliRBM] Iteration 36, pseudo-likelihood = -18763.98, time = 4.40s
[BernoulliRBM] Iteration 37, pseudo-likelihood = -18636.51, time = 4.32s
[BernoulliRBM] Iteration 38, pseudo-likelihood = -18566.23, time = 4.48s
[BernoulliRBM] Iteration 39, pseudo-likelihood = -18519.12, time = 4.34s
[BernoulliRBM] Iteration 40, pseudo-likelihood = -18445.21, time = 4.35s
[BernoulliRBM] Iteration 41, pseudo-likelihood = -18403.25, time = 4.29s
[BernoulliRBM] Iteration 42, pseudo-likelihood = -18398.80, time = 4.53s
[BernoulliRBM] Iteration 43, pseudo-likelihood = -18468.53, time = 4.34s
[BernoulliRBM] Iteration 44, pseudo-likelihood = -18408.68, time = 4.35s
[BernoulliRBM] Iteration 45, pseudo-likelihood = -18454.48, time = 4.44s
[BernoulliRBM] Iteration 46, pseudo-likelihood = -18476.11, time = 4.38s
[BernoulliRBM] Iteration 47, pseudo-likelihood = -18424.23, time = 4.92s
[BernoulliRBM] Iteration 48, pseudo-likelihood = -18269.10, time = 4.13s
[BernoulliRBM] Iteration 49, pseudo-likelihood = -18324.67, time = 4.26s
[BernoulliRBM] Iteration 50, pseudo-likelihood = -18463.44, time = 4.32s
[BernoulliRBM] Iteration 51, pseudo-likelihood = -18309.25, time = 4.31s
[BernoulliRBM] Iteration 52, pseudo-likelihood = -18228.59, time = 4.41s
[BernoulliRBM] Iteration 53, pseudo-likelihood = -18251.30, time = 4.26s
[BernoulliRBM] Iteration 54, pseudo-likelihood = -18395.55, time = 4.50s
[BernoulliRBM] Iteration 55, pseudo-likelihood = -18303.91, time = 5.07s
[BernoulliRBM] Iteration 56, pseudo-likelihood = -18221.20, time = 4.44s
[BernoulliRBM] Iteration 57, pseudo-likelihood = -18157.73, time = 4.47s
[BernoulliRBM] Iteration 58, pseudo-likelihood = -18095.87, time = 4.39s
[BernoulliRBM] Iteration 59, pseudo-likelihood = -18256.35, time = 4.46s
[BernoulliRBM] Iteration 60, pseudo-likelihood = -18255.31, time = 4.64s
[BernoulliRBM] Iteration 61, pseudo-likelihood = -18190.27, time = 4.45s
[BernoulliRBM] Iteration 62, pseudo-likelihood = -18256.75, time = 4.38s
[BernoulliRBM] Iteration 63, pseudo-likelihood = -18156.42, time = 4.43s
[BernoulliRBM] Iteration 64, pseudo-likelihood = -18247.56, time = 4.40s
```

```
[BernoulliRBM] Iteration 65, pseudo-likelihood = -18270.03, time = 4.59s
[BernoulliRBM] Iteration 66, pseudo-likelihood = -18143.28, time = 4.31s
[BernoulliRBM] Iteration 67, pseudo-likelihood = -18211.24, time = 4.37s
[BernoulliRBM] Iteration 68, pseudo-likelihood = -18262.56, time = 4.39s
[BernoulliRBM] Iteration 69, pseudo-likelihood = -18191.26, time = 4.34s
[BernoulliRBM] Iteration 70, pseudo-likelihood = -18216.49, time = 4.39s
[BernoulliRBM] Iteration 71, pseudo-likelihood = -18132.25, time = 4.37s
[BernoulliRBM] Iteration 72, pseudo-likelihood = -18059.66, time = 4.52s
[BernoulliRBM] Iteration 73, pseudo-likelihood = -18030.91, time = 4.84s
[BernoulliRBM] Iteration 74, pseudo-likelihood = -18132.15, time = 4.81s
[BernoulliRBM] Iteration 75, pseudo-likelihood = -17998.81, time = 4.46s
[BernoulliRBM] Iteration 76, pseudo-likelihood = -18167.70, time = 4.35s
[BernoulliRBM] Iteration 77, pseudo-likelihood = -18120.52, time = 4.49s
[BernoulliRBM] Iteration 78, pseudo-likelihood = -18075.93, time = 4.49s
[BernoulliRBM] Iteration 79, pseudo-likelihood = -18220.80, time = 4.85s
[BernoulliRBM] Iteration 80, pseudo-likelihood = -18101.14, time = 4.86s
[BernoulliRBM] Iteration 81, pseudo-likelihood = -18092.97, time = 5.00s
[BernoulliRBM] Iteration 82, pseudo-likelihood = -18111.10, time = 4.58s
[BernoulliRBM] Iteration 83, pseudo-likelihood = -18166.53, time = 4.51s
[BernoulliRBM] Iteration 84, pseudo-likelihood = -18105.99, time = 4.66s
[BernoulliRBM] Iteration 85, pseudo-likelihood = -18020.30, time = 4.55s
[BernoulliRBM] Iteration 86, pseudo-likelihood = -17916.55, time = 4.55s
[BernoulliRBM] Iteration 87, pseudo-likelihood = -18092.54, time = 4.83s
[BernoulliRBM] Iteration 88, pseudo-likelihood = -17993.85, time = 4.46s
[BernoulliRBM] Iteration 89, pseudo-likelihood = -17979.78, time = 4.50s
[BernoulliRBM] Iteration 90, pseudo-likelihood = -18212.53, time = 4.62s
[BernoulliRBM] Iteration 91, pseudo-likelihood = -18106.06, time = 4.89s
[BernoulliRBM] Iteration 92, pseudo-likelihood = -18096.96, time = 5.09s
[BernoulliRBM] Iteration 93, pseudo-likelihood = -18103.80, time = 4.41s
[BernoulliRBM] Iteration 94, pseudo-likelihood = -18095.83, time = 4.39s
[BernoulliRBM] Iteration 95, pseudo-likelihood = -18003.27, time = 4.37s
[BernoulliRBM] Iteration 96, pseudo-likelihood = -18005.35, time = 4.43s
[BernoulliRBM] Iteration 97, pseudo-likelihood = -18186.54, time = 4.42s
[BernoulliRBM] Iteration 98, pseudo-likelihood = -18090.66, time = 4.76s
[BernoulliRBM] Iteration 99, pseudo-likelihood = -18107.15, time = 4.67s
[BernoulliRBM] Iteration 100, pseudo-likelihood = -18081.05, time = 5.18s
[BernoulliRBM] Iteration 1, pseudo-likelihood = -11.45, time = 0.00s
[BernoulliRBM] Iteration 2, pseudo-likelihood = -10.93, time = 0.03s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -10.77, time = 0.03s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -10.60, time = 0.03s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -10.45, time = 0.04s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -10.28, time = 0.02s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -10.07, time = 0.02s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -9.88, time = 0.03s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -9.65, time = 0.02s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -9.37, time = 0.03s
[BernoulliRBM] Iteration 11, pseudo-likelihood = -8.96, time = 0.03s
[BernoulliRBM] Iteration 12, pseudo-likelihood = -8.45, time = 0.02s
[BernoulliRBM] Iteration 13, pseudo-likelihood = -7.94, time = 0.04s
[BernoulliRBM] Iteration 14, pseudo-likelihood = -7.44, time = 0.02s
[BernoulliRBM] Iteration 15, pseudo-likelihood = -7.02, time = 0.02s
[BernoulliRBM] Iteration 16, pseudo-likelihood = -6.71, time = 0.03s
[BernoulliRBM] Iteration 17, pseudo-likelihood = -6.48, time = 0.02s
[BernoulliRBM] Iteration 18, pseudo-likelihood = -6.34, time = 0.01s
[BernoulliRBM] Iteration 19, pseudo-likelihood = -6.26, time = 0.02s
[BernoulliRBM] Iteration 20, pseudo-likelihood = -6.15, time = 0.03s
[BernoulliRBM] Iteration 21, pseudo-likelihood = -6.09, time = 0.03s
[BernoulliRBM] Iteration 22, pseudo-likelihood = -6.04, time = 0.02s
[BernoulliRBM] Iteration 23, pseudo-likelihood = -6.01, time = 0.02s
[BernoulliRBM] Iteration 24, pseudo-likelihood = -5.97, time = 0.02s
[BernoulliRBM] Iteration 25, pseudo-likelihood = -5.94, time = 0.02s
[BernoulliRBM] Iteration 26, pseudo-likelihood = -5.96, time = 0.02s
[BernoulliRBM] Iteration 27, pseudo-likelihood = -5.93, time = 0.02s
[BernoulliRBM] Iteration 28, pseudo-likelihood = -5.92, time = 0.02s
```

```
[BernoulliRBM] Iteration 29, pseudo-likelihood = -5.87, time = 0.02s
[BernoulliRBM] Iteration 30, pseudo-likelihood = -5.86, time = 0.02s
[BernoulliRBM] Iteration 31, pseudo-likelihood = -5.86, time = 0.02s
[BernoulliRBM] Iteration 32, pseudo-likelihood = -5.83, time = 0.01s
[BernoulliRBM] Iteration 33, pseudo-likelihood = -5.81, time = 0.03s
[BernoulliRBM] Iteration 34, pseudo-likelihood = -5.86, time = 0.02s
[BernoulliRBM] Iteration 35, pseudo-likelihood = -5.76, time = 0.02s
[BernoulliRBM] Iteration 36, pseudo-likelihood = -5.83, time = 0.03s
[BernoulliRBM] Iteration 37, pseudo-likelihood = -5.73, time = 0.02s
[BernoulliRBM] Iteration 38, pseudo-likelihood = -5.70, time = 0.02s
[BernoulliRBM] Iteration 39, pseudo-likelihood = -5.66, time = 0.02s
[BernoulliRBM] Iteration 40, pseudo-likelihood = -5.65, time = 0.02s
[BernoulliRBM] Iteration 41, pseudo-likelihood = -5.63, time = 0.03s
[BernoulliRBM] Iteration 42, pseudo-likelihood = -5.57, time = 0.04s
[BernoulliRBM] Iteration 43, pseudo-likelihood = -5.54, time = 0.02s
[BernoulliRBM] Iteration 44, pseudo-likelihood = -5.58, time = 0.02s
[BernoulliRBM] Iteration 45, pseudo-likelihood = -5.44, time = 0.02s
[BernoulliRBM] Iteration 46, pseudo-likelihood = -5.42, time = 0.03s
[BernoulliRBM] Iteration 47, pseudo-likelihood = -5.33, time = 0.02s
[BernoulliRBM] Iteration 48, pseudo-likelihood = -5.29, time = 0.03s
[BernoulliRBM] Iteration 49, pseudo-likelihood = -5.19, time = 0.03s
[BernoulliRBM] Iteration 50, pseudo-likelihood = -5.11, time = 0.01s
[BernoulliRBM] Iteration 51, pseudo-likelihood = -5.04, time = 0.05s
[BernoulliRBM] Iteration 52, pseudo-likelihood = -4.99, time = 0.03s
[BernoulliRBM] Iteration 53, pseudo-likelihood = -4.92, time = 0.03s
[BernoulliRBM] Iteration 54, pseudo-likelihood = -4.82, time = 0.01s
[BernoulliRBM] Iteration 55, pseudo-likelihood = -4.71, time = 0.03s
[BernoulliRBM] Iteration 56, pseudo-likelihood = -4.67, time = 0.02s
[BernoulliRBM] Iteration 57, pseudo-likelihood = -4.54, time = 0.02s
[BernoulliRBM] Iteration 58, pseudo-likelihood = -4.49, time = 0.04s
[BernoulliRBM] Iteration 59, pseudo-likelihood = -4.36, time = 0.02s
[BernoulliRBM] Iteration 60, pseudo-likelihood = -4.24, time = 0.02s
[BernoulliRBM] Iteration 61, pseudo-likelihood = -4.15, time = 0.03s
[BernoulliRBM] Iteration 62, pseudo-likelihood = -4.04, time = 0.03s
[BernoulliRBM] Iteration 63, pseudo-likelihood = -3.98, time = 0.02s
[BernoulliRBM] Iteration 64, pseudo-likelihood = -3.85, time = 0.02s
[BernoulliRBM] Iteration 65, pseudo-likelihood = -3.78, time = 0.02s
[BernoulliRBM] Iteration 66, pseudo-likelihood = -3.64, time = 0.02s
[BernoulliRBM] Iteration 67, pseudo-likelihood = -3.56, time = 0.02s
[BernoulliRBM] Iteration 68, pseudo-likelihood = -3.50, time = 0.02s
[BernoulliRBM] Iteration 69, pseudo-likelihood = -3.44, time = 0.01s
[BernoulliRBM] Iteration 70, pseudo-likelihood = -3.34, time = 0.03s
[BernoulliRBM] Iteration 71, pseudo-likelihood = -3.22, time = 0.02s
[BernoulliRBM] Iteration 72, pseudo-likelihood = -3.14, time = 0.02s
[BernoulliRBM] Iteration 73, pseudo-likelihood = -3.09, time = 0.02s
[BernoulliRBM] Iteration 74, pseudo-likelihood = -3.06, time = 0.02s
[BernoulliRBM] Iteration 75, pseudo-likelihood = -3.01, time = 0.02s
[BernoulliRBM] Iteration 76, pseudo-likelihood = -2.97, time = 0.02s
[BernoulliRBM] Iteration 77, pseudo-likelihood = -2.90, time = 0.03s
[BernoulliRBM] Iteration 78, pseudo-likelihood = -2.90, time = 0.02s
[BernoulliRBM] Iteration 79, pseudo-likelihood = -2.82, time = 0.02s
[BernoulliRBM] Iteration 80, pseudo-likelihood = -2.78, time = 0.01s
[BernoulliRBM] Iteration 81, pseudo-likelihood = -2.77, time = 0.02s
[BernoulliRBM] Iteration 82, pseudo-likelihood = -2.71, time = 0.02s
[BernoulliRBM] Iteration 83, pseudo-likelihood = -2.68, time = 0.03s
[BernoulliRBM] Iteration 84, pseudo-likelihood = -2.65, time = 0.02s
[BernoulliRBM] Iteration 85, pseudo-likelihood = -2.64, time = 0.01s
[BernoulliRBM] Iteration 86, pseudo-likelihood = -2.60, time = 0.02s
[BernoulliRBM] Iteration 87, pseudo-likelihood = -2.63, time = 0.03s
[BernoulliRBM] Iteration 88, pseudo-likelihood = -2.54, time = 0.02s
[BernoulliRBM] Iteration 89, pseudo-likelihood = -2.54, time = 0.03s
[BernoulliRBM] Iteration 90, pseudo-likelihood = -2.52, time = 0.03s
[BernoulliRBM] Iteration 91, pseudo-likelihood = -2.54, time = 0.02s
[BernoulliRBM] Iteration 92, pseudo-likelihood = -2.49, time = 0.03s
```

```
[BernoulliRBM] Iteration 93, pseudo-likelihood = -2.49, time = 0.01s
[BernoulliRBM] Iteration 94, pseudo-likelihood = -2.46, time = 0.04s
[BernoulliRBM] Iteration 95, pseudo-likelihood = -2.49, time = 0.04s
[BernoulliRBM] Iteration 96, pseudo-likelihood = -2.49, time = 0.03s
[BernoulliRBM] Iteration 97, pseudo-likelihood = -2.42, time = 0.02s
[BernoulliRBM] Iteration 98, pseudo-likelihood = -2.53, time = 0.04s
[BernoulliRBM] Iteration 99, pseudo-likelihood = -2.39, time = 0.03s
[BernoulliRBM] Iteration 100, pseudo-likelihood = -2.43, time = 0.01s
```

Out[24]:

```
Pipeline(steps=[('rbm1',
                 BernoulliRBM(learning_rate=0.01, n_components=20, n_iter=100,
                              random_state=0, verbose=True)),
                ('rbm2',
                 BernoulliRBM(learning_rate=0.01, n_components=50, n_iter=100,
                              random_state=0, verbose=True))])
```

In [77]:

```python
# 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 [78]:

```python
# 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=['accu
    model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, va
```

In [79]:

```python
# 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 [80]:

```python
X_train_resized.shape
```

Out[80]:

```
(1209, 28, 28, 1)
```

In [81]:

```python
# 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 [82]:
```
X_test_resized.shape
```

Out[82]:
```
(303, 28, 28, 1)
```

In [83]:
```
# Define input shape and number of classes
input_shape = (28, 28, 1)
num_classes = 10
```

In [84]:
```
# Define model
model = define_model(input_shape, num_classes)
```

In [85]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 13, 13, 32)       0         
 2D)                                                             
                                                                 
 dropout_9 (Dropout)         (None, 13, 13, 32)        0         
                                                                 
 conv2d_7 (Conv2D)           (None, 11, 11, 64)        18496     
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             
                                                                 
 dropout_10 (Dropout)        (None, 5, 5, 64)          0         
                                                                 
 flatten_3 (Flatten)         (None, 1600)              0         
                                                                 
 dense_6 (Dense)             (None, 128)               204928    
                                                                 
 dropout_11 (Dropout)        (None, 128)               0         
                                                                 
 dense_7 (Dense)             (None, 10)                1290      
                                                                 
=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
_________________________________________________________________
```

In [86]:
```
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 [87]:
```
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 [88]:
```
# 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
10/10 [==============================] - 6s 149ms/step - loss: 15.3606 - accuracy:
0.4748 - val_loss: 0.6470 - val_accuracy: 0.6799
Epoch 2/70
10/10 [==============================] - 1s 88ms/step - loss: 2.8379 - accuracy:
0.5988 - val_loss: 0.6307 - val_accuracy: 0.6634
Epoch 3/70
10/10 [==============================] - 1s 103ms/step - loss: 1.0779 - accuracy:
0.6253 - val_loss: 0.5156 - val_accuracy: 0.7327
Epoch 4/70
10/10 [==============================] - 1s 72ms/step - loss: 0.6631 - accuracy:
0.6551 - val_loss: 0.5041 - val_accuracy: 0.7063
Epoch 5/70
10/10 [==============================] - 1s 70ms/step - loss: 0.5883 - accuracy:
0.6634 - val_loss: 0.5171 - val_accuracy: 0.7162
Epoch 6/70
10/10 [==============================] - 1s 74ms/step - loss: 0.5493 - accuracy:
0.6832 - val_loss: 0.5216 - val_accuracy: 0.7591
Epoch 7/70
10/10 [==============================] - 1s 71ms/step - loss: 0.5296 - accuracy:
0.6816 - val_loss: 0.4996 - val_accuracy: 0.7492
Epoch 8/70
10/10 [==============================] - 1s 73ms/step - loss: 0.5084 - accuracy:
0.6931 - val_loss: 0.5047 - val_accuracy: 0.7657
Epoch 9/70
10/10 [==============================] - 1s 75ms/step - loss: 0.4986 - accuracy:
0.7089 - val_loss: 0.4890 - val_accuracy: 0.7591
Epoch 10/70
10/10 [==============================] - 1s 73ms/step - loss: 0.4793 - accuracy:
0.7262 - val_loss: 0.4809 - val_accuracy: 0.8086
Epoch 11/70
10/10 [==============================] - 1s 77ms/step - loss: 0.4639 - accuracy:
0.7477 - val_loss: 0.4592 - val_accuracy: 0.8152
Epoch 12/70
10/10 [==============================] - 1s 90ms/step - loss: 0.4586 - accuracy:
0.7494 - val_loss: 0.4386 - val_accuracy: 0.8812
Epoch 13/70
10/10 [==============================] - 1s 103ms/step - loss: 0.4463 - accuracy:
0.7560 - val_loss: 0.4181 - val_accuracy: 0.8746
Epoch 14/70
10/10 [==============================] - 1s 138ms/step - loss: 0.4352 - accuracy:
0.7543 - val_loss: 0.3886 - val_accuracy: 0.9142
Epoch 15/70
10/10 [==============================] - 1s 116ms/step - loss: 0.4176 - accuracy:
0.7750 - val_loss: 0.3760 - val_accuracy: 0.9208
Epoch 16/70
10/10 [==============================] - 1s 127ms/step - loss: 0.4087 - accuracy:
0.7907 - val_loss: 0.3765 - val_accuracy: 0.9373
Epoch 17/70
10/10 [==============================] - 1s 121ms/step - loss: 0.4006 - accuracy:
0.7949 - val_loss: 0.3605 - val_accuracy: 0.9472
Epoch 18/70
10/10 [==============================] - 1s 129ms/step - loss: 0.3907 - accuracy:
0.8065 - val_loss: 0.3172 - val_accuracy: 0.9472
Epoch 19/70
10/10 [==============================] - 1s 111ms/step - loss: 0.3676 - accuracy:
0.8296 - val_loss: 0.2771 - val_accuracy: 0.9439
Epoch 20/70
10/10 [==============================] - 1s 134ms/step - loss: 0.3340 - accuracy:
0.8462 - val_loss: 0.2859 - val_accuracy: 0.9274
Epoch 21/70
10/10 [==============================] - 1s 110ms/step - loss: 0.3157 - accuracy:
0.8610 - val_loss: 0.2467 - val_accuracy: 0.9241
Epoch 22/70
```

```
10/10 [==============================] - 1s 106ms/step - loss: 0.3270 - accuracy: 
0.8734 - val_loss: 0.2617 - val_accuracy: 0.9208
Epoch 23/70
10/10 [==============================] - 1s 98ms/step - loss: 0.3129 - accuracy: 
0.8635 - val_loss: 0.2331 - val_accuracy: 0.9274
Epoch 24/70
10/10 [==============================] - 1s 93ms/step - loss: 0.2944 - accuracy: 
0.8792 - val_loss: 0.2358 - val_accuracy: 0.9241
Epoch 25/70
10/10 [==============================] - 1s 88ms/step - loss: 0.2946 - accuracy: 
0.8817 - val_loss: 0.2449 - val_accuracy: 0.9076
Epoch 26/70
10/10 [==============================] - 1s 92ms/step - loss: 0.2796 - accuracy: 
0.8925 - val_loss: 0.2172 - val_accuracy: 0.9505
Epoch 27/70
10/10 [==============================] - 1s 127ms/step - loss: 0.2618 - accuracy: 
0.8842 - val_loss: 0.2100 - val_accuracy: 0.9274
Epoch 28/70
10/10 [==============================] - 1s 104ms/step - loss: 0.2614 - accuracy: 
0.8991 - val_loss: 0.2213 - val_accuracy: 0.9439
Epoch 29/70
10/10 [==============================] - 1s 118ms/step - loss: 0.2669 - accuracy: 
0.8925 - val_loss: 0.1988 - val_accuracy: 0.9505
Epoch 30/70
10/10 [==============================] - 1s 136ms/step - loss: 0.2502 - accuracy: 
0.8974 - val_loss: 0.1883 - val_accuracy: 0.9274
Epoch 31/70
10/10 [==============================] - 1s 127ms/step - loss: 0.2382 - accuracy: 
0.9074 - val_loss: 0.1886 - val_accuracy: 0.9142
Epoch 32/70
10/10 [==============================] - 1s 118ms/step - loss: 0.2291 - accuracy: 
0.8974 - val_loss: 0.2569 - val_accuracy: 0.8680
Epoch 33/70
10/10 [==============================] - 1s 112ms/step - loss: 0.3414 - accuracy: 
0.8586 - val_loss: 0.2989 - val_accuracy: 0.9109
Epoch 34/70
10/10 [==============================] - 1s 99ms/step - loss: 0.3124 - accuracy: 
0.8743 - val_loss: 0.1831 - val_accuracy: 0.9538
Epoch 35/70
10/10 [==============================] - 1s 106ms/step - loss: 0.2588 - accuracy: 
0.8850 - val_loss: 0.1904 - val_accuracy: 0.9241
Epoch 36/70
10/10 [==============================] - 1s 101ms/step - loss: 0.2183 - accuracy: 
0.9123 - val_loss: 0.1951 - val_accuracy: 0.9010
Epoch 37/70
10/10 [==============================] - 1s 114ms/step - loss: 0.2199 - accuracy: 
0.9115 - val_loss: 0.1779 - val_accuracy: 0.9274
Epoch 38/70
10/10 [==============================] - 1s 117ms/step - loss: 0.1954 - accuracy: 
0.9256 - val_loss: 0.1783 - val_accuracy: 0.9505
Epoch 39/70
10/10 [==============================] - 1s 108ms/step - loss: 0.1879 - accuracy: 
0.9297 - val_loss: 0.1797 - val_accuracy: 0.9175
Epoch 40/70
10/10 [==============================] - 1s 98ms/step - loss: 0.1994 - accuracy: 
0.9222 - val_loss: 0.1708 - val_accuracy: 0.9076
Epoch 41/70
10/10 [==============================] - 1s 89ms/step - loss: 0.1638 - accuracy: 
0.9363 - val_loss: 0.1766 - val_accuracy: 0.9208
Epoch 42/70
10/10 [==============================] - 1s 89ms/step - loss: 0.1835 - accuracy: 
0.9264 - val_loss: 0.1484 - val_accuracy: 0.9340
Epoch 43/70
10/10 [==============================] - 1s 94ms/step - loss: 0.1631 - accuracy: 
```

```
0.9363 - val_loss: 0.1969 - val_accuracy: 0.9538
Epoch 44/70
10/10 [==============================] - 1s 93ms/step - loss: 0.1573 - accuracy:
0.9322 - val_loss: 0.1466 - val_accuracy: 0.9307
Epoch 45/70
10/10 [==============================] - 1s 92ms/step - loss: 0.1651 - accuracy:
0.9388 - val_loss: 0.1316 - val_accuracy: 0.9472
Epoch 46/70
10/10 [==============================] - 1s 94ms/step - loss: 0.1342 - accuracy:
0.9504 - val_loss: 0.1436 - val_accuracy: 0.9505
Epoch 47/70
10/10 [==============================] - 1s 100ms/step - loss: 0.1428 - accuracy:
0.9479 - val_loss: 0.1355 - val_accuracy: 0.9505
Epoch 48/70
10/10 [==============================] - 1s 120ms/step - loss: 0.1435 - accuracy:
0.9487 - val_loss: 0.1341 - val_accuracy: 0.9307
Epoch 49/70
10/10 [==============================] - 1s 90ms/step - loss: 0.1462 - accuracy:
0.9446 - val_loss: 0.1314 - val_accuracy: 0.9373
Epoch 50/70
10/10 [==============================] - 1s 101ms/step - loss: 0.1301 - accuracy:
0.9479 - val_loss: 0.1495 - val_accuracy: 0.9439
Epoch 51/70
10/10 [==============================] - 1s 100ms/step - loss: 0.1348 - accuracy:
0.9537 - val_loss: 0.1689 - val_accuracy: 0.9538
Epoch 52/70
10/10 [==============================] - 1s 130ms/step - loss: 0.1259 - accuracy:
0.9562 - val_loss: 0.1766 - val_accuracy: 0.9538
Epoch 53/70
10/10 [==============================] - 1s 138ms/step - loss: 0.1764 - accuracy:
0.9297 - val_loss: 0.1781 - val_accuracy: 0.9538
Epoch 54/70
10/10 [==============================] - 1s 128ms/step - loss: 0.1891 - accuracy:
0.9355 - val_loss: 0.1307 - val_accuracy: 0.9571
Epoch 55/70
10/10 [==============================] - 1s 125ms/step - loss: 0.1809 - accuracy:
0.9165 - val_loss: 0.1311 - val_accuracy: 0.9472
Epoch 56/70
10/10 [==============================] - 2s 161ms/step - loss: 0.1567 - accuracy:
0.9338 - val_loss: 0.1647 - val_accuracy: 0.9142
Epoch 57/70
10/10 [==============================] - 1s 127ms/step - loss: 0.1518 - accuracy:
0.9404 - val_loss: 0.1324 - val_accuracy: 0.9439
Epoch 58/70
10/10 [==============================] - 1s 120ms/step - loss: 0.1186 - accuracy:
0.9570 - val_loss: 0.1389 - val_accuracy: 0.9472
Epoch 59/70
10/10 [==============================] - 1s 121ms/step - loss: 0.1041 - accuracy:
0.9620 - val_loss: 0.1685 - val_accuracy: 0.9505
Epoch 60/70
10/10 [==============================] - 1s 138ms/step - loss: 0.1257 - accuracy:
0.9545 - val_loss: 0.1307 - val_accuracy: 0.9505
Epoch 61/70
10/10 [==============================] - 1s 124ms/step - loss: 0.1154 - accuracy:
0.9595 - val_loss: 0.1356 - val_accuracy: 0.9472
Epoch 62/70
10/10 [==============================] - 1s 121ms/step - loss: 0.1010 - accuracy:
0.9644 - val_loss: 0.1339 - val_accuracy: 0.9505
Epoch 63/70
10/10 [==============================] - 1s 122ms/step - loss: 0.1015 - accuracy:
0.9644 - val_loss: 0.1094 - val_accuracy: 0.9439
Epoch 64/70
10/10 [==============================] - 1s 116ms/step - loss: 0.1164 - accuracy:
0.9545 - val_loss: 0.1352 - val_accuracy: 0.9604
```

```
Epoch 65/70
10/10 [==============================] - 1s 118ms/step - loss: 0.1207 - accuracy:
0.9553 - val_loss: 0.1207 - val_accuracy: 0.9505
Epoch 66/70
10/10 [==============================] - 1s 116ms/step - loss: 0.0955 - accuracy:
0.9653 - val_loss: 0.1678 - val_accuracy: 0.9505
Epoch 67/70
10/10 [==============================] - 1s 116ms/step - loss: 0.1473 - accuracy:
0.9404 - val_loss: 0.1141 - val_accuracy: 0.9604
Epoch 68/70
10/10 [==============================] - 1s 115ms/step - loss: 0.1433 - accuracy:
0.9479 - val_loss: 0.1125 - val_accuracy: 0.9538
Epoch 69/70
10/10 [==============================] - 1s 116ms/step - loss: 0.1006 - accuracy:
0.9603 - val_loss: 0.1040 - val_accuracy: 0.9604
Epoch 70/70
10/10 [==============================] - 1s 123ms/step - loss: 0.0921 - accuracy:
0.9644 - val_loss: 0.1212 - val_accuracy: 0.9505
```

In [89]:

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

```
Epoch 1/70
38/38 [==============================] - 1s 30ms/step - loss: 0.1348 - accuracy:
0.9487 - val_loss: 0.1294 - val_accuracy: 0.9439
Epoch 2/70
38/38 [==============================] - 1s 30ms/step - loss: 0.1451 - accuracy:
0.9438 - val_loss: 0.1558 - val_accuracy: 0.9604
Epoch 3/70
38/38 [==============================] - 2s 40ms/step - loss: 0.1083 - accuracy:
0.9578 - val_loss: 0.0898 - val_accuracy: 0.9538
Epoch 4/70
38/38 [==============================] - 1s 36ms/step - loss: 0.1286 - accuracy:
0.9504 - val_loss: 0.1186 - val_accuracy: 0.9571
Epoch 5/70
38/38 [==============================] - 1s 35ms/step - loss: 0.1043 - accuracy:
0.9578 - val_loss: 0.0799 - val_accuracy: 0.9604
Epoch 6/70
38/38 [==============================] - 1s 33ms/step - loss: 0.1301 - accuracy:
0.9537 - val_loss: 0.1148 - val_accuracy: 0.9604
Epoch 7/70
38/38 [==============================] - 1s 39ms/step - loss: 0.1091 - accuracy:
0.9611 - val_loss: 0.1106 - val_accuracy: 0.9670
Epoch 8/70
38/38 [==============================] - 2s 44ms/step - loss: 0.0863 - accuracy:
0.9686 - val_loss: 0.1246 - val_accuracy: 0.9571
Epoch 9/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0872 - accuracy:
0.9661 - val_loss: 0.1230 - val_accuracy: 0.9571
Epoch 10/70
38/38 [==============================] - 1s 34ms/step - loss: 0.1049 - accuracy:
0.9586 - val_loss: 0.0660 - val_accuracy: 0.9637
Epoch 11/70
38/38 [==============================] - 1s 37ms/step - loss: 0.0886 - accuracy:
0.9677 - val_loss: 0.0928 - val_accuracy: 0.9604
Epoch 12/70
38/38 [==============================] - 1s 39ms/step - loss: 0.0678 - accuracy:
0.9744 - val_loss: 0.0700 - val_accuracy: 0.9637
Epoch 13/70
38/38 [==============================] - 2s 41ms/step - loss: 0.0560 - accuracy:
0.9735 - val_loss: 0.0880 - val_accuracy: 0.9604
Epoch 14/70
38/38 [==============================] - 2s 44ms/step - loss: 0.0595 - accuracy:
0.9793 - val_loss: 0.0956 - val_accuracy: 0.9637
Epoch 15/70
38/38 [==============================] - 1s 38ms/step - loss: 0.0710 - accuracy:
0.9711 - val_loss: 0.0830 - val_accuracy: 0.9472
Epoch 16/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0571 - accuracy:
0.9793 - val_loss: 0.0609 - val_accuracy: 0.9736
Epoch 17/70
38/38 [==============================] - 2s 42ms/step - loss: 0.0523 - accuracy:
0.9810 - val_loss: 0.0811 - val_accuracy: 0.9571
Epoch 18/70
38/38 [==============================] - 1s 34ms/step - loss: 0.0809 - accuracy:
0.9735 - val_loss: 0.0540 - val_accuracy: 0.9604
Epoch 19/70
38/38 [==============================] - 1s 34ms/step - loss: 0.0799 - accuracy:
0.9686 - val_loss: 0.0875 - val_accuracy: 0.9538
Epoch 20/70
38/38 [==============================] - 1s 34ms/step - loss: 0.0595 - accuracy:
0.9793 - val_loss: 0.0517 - val_accuracy: 0.9769
Epoch 21/70
38/38 [==============================] - 1s 38ms/step - loss: 0.0690 - accuracy:
0.9694 - val_loss: 0.0932 - val_accuracy: 0.9736
Epoch 22/70
```

```
38/38 [==============================] - 2s 43ms/step - loss: 0.0934 - accuracy: 
0.9719 - val_loss: 0.0502 - val_accuracy: 0.9835
Epoch 23/70
38/38 [==============================] - 1s 38ms/step - loss: 0.0523 - accuracy: 
0.9777 - val_loss: 0.0641 - val_accuracy: 0.9604
Epoch 24/70
38/38 [==============================] - 2s 41ms/step - loss: 0.0525 - accuracy: 
0.9826 - val_loss: 0.0730 - val_accuracy: 0.9571
Epoch 25/70
38/38 [==============================] - 1s 38ms/step - loss: 0.0497 - accuracy: 
0.9777 - val_loss: 0.0885 - val_accuracy: 0.9538
Epoch 26/70
38/38 [==============================] - 1s 35ms/step - loss: 0.0874 - accuracy: 
0.9669 - val_loss: 0.0593 - val_accuracy: 0.9802
Epoch 27/70
38/38 [==============================] - 1s 35ms/step - loss: 0.0520 - accuracy: 
0.9843 - val_loss: 0.1268 - val_accuracy: 0.9604
Epoch 28/70
38/38 [==============================] - 2s 40ms/step - loss: 0.0418 - accuracy: 
0.9835 - val_loss: 0.0925 - val_accuracy: 0.9604
Epoch 29/70
38/38 [==============================] - 2s 40ms/step - loss: 0.0557 - accuracy: 
0.9785 - val_loss: 0.0641 - val_accuracy: 0.9703
Epoch 30/70
38/38 [==============================] - 1s 34ms/step - loss: 0.0595 - accuracy: 
0.9777 - val_loss: 0.0949 - val_accuracy: 0.9637
Epoch 31/70
38/38 [==============================] - 1s 35ms/step - loss: 0.0610 - accuracy: 
0.9744 - val_loss: 0.0601 - val_accuracy: 0.9670
Epoch 32/70
38/38 [==============================] - 1s 33ms/step - loss: 0.0715 - accuracy: 
0.9801 - val_loss: 0.0681 - val_accuracy: 0.9703
Epoch 33/70
38/38 [==============================] - 1s 35ms/step - loss: 0.0491 - accuracy: 
0.9785 - val_loss: 0.0736 - val_accuracy: 0.9637
Epoch 34/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0538 - accuracy: 
0.9826 - val_loss: 0.0617 - val_accuracy: 0.9670
Epoch 35/70
38/38 [==============================] - 1s 35ms/step - loss: 0.0582 - accuracy: 
0.9826 - val_loss: 0.0450 - val_accuracy: 0.9736
Epoch 36/70
38/38 [==============================] - 1s 34ms/step - loss: 0.1478 - accuracy: 
0.9545 - val_loss: 0.1041 - val_accuracy: 0.9736
Epoch 37/70
38/38 [==============================] - 2s 42ms/step - loss: 0.1036 - accuracy: 
0.9562 - val_loss: 0.0695 - val_accuracy: 0.9571
Epoch 38/70
38/38 [==============================] - 1s 37ms/step - loss: 0.0616 - accuracy: 
0.9818 - val_loss: 0.0948 - val_accuracy: 0.9571
Epoch 39/70
38/38 [==============================] - 1s 37ms/step - loss: 0.0844 - accuracy: 
0.9719 - val_loss: 0.0891 - val_accuracy: 0.9571
Epoch 40/70
38/38 [==============================] - 2s 44ms/step - loss: 0.0605 - accuracy: 
0.9793 - val_loss: 0.0727 - val_accuracy: 0.9637
Epoch 41/70
38/38 [==============================] - 1s 35ms/step - loss: 0.0385 - accuracy: 
0.9884 - val_loss: 0.0779 - val_accuracy: 0.9571
Epoch 42/70
38/38 [==============================] - 1s 33ms/step - loss: 0.0360 - accuracy: 
0.9843 - val_loss: 0.1243 - val_accuracy: 0.9538
Epoch 43/70
38/38 [==============================] - 2s 49ms/step - loss: 0.0400 - accuracy: 
```

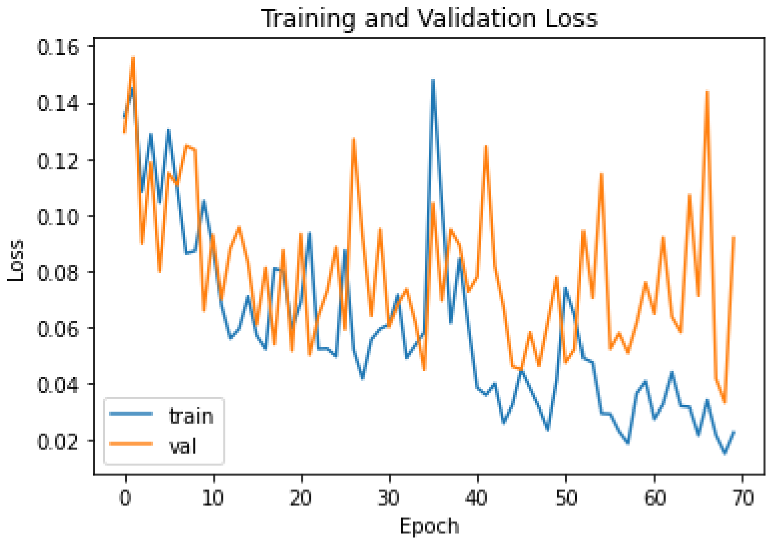
```
0.9810 - val_loss: 0.0817 - val_accuracy: 0.9604
Epoch 44/70
38/38 [==============================] - 1s 33ms/step - loss: 0.0262 - accuracy: 0.9934 - val_loss: 0.0671 - val_accuracy: 0.9736
Epoch 45/70
38/38 [==============================] - 2s 43ms/step - loss: 0.0326 - accuracy: 0.9892 - val_loss: 0.0461 - val_accuracy: 0.9637
Epoch 46/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0452 - accuracy: 0.9818 - val_loss: 0.0450 - val_accuracy: 0.9802
Epoch 47/70
38/38 [==============================] - 2s 41ms/step - loss: 0.0383 - accuracy: 0.9884 - val_loss: 0.0582 - val_accuracy: 0.9703
Epoch 48/70
38/38 [==============================] - 1s 34ms/step - loss: 0.0316 - accuracy: 0.9859 - val_loss: 0.0463 - val_accuracy: 0.9637
Epoch 49/70
38/38 [==============================] - 1s 37ms/step - loss: 0.0238 - accuracy: 0.9901 - val_loss: 0.0618 - val_accuracy: 0.9769
Epoch 50/70
38/38 [==============================] - 2s 40ms/step - loss: 0.0412 - accuracy: 0.9868 - val_loss: 0.0778 - val_accuracy: 0.9637
Epoch 51/70
38/38 [==============================] - 2s 51ms/step - loss: 0.0740 - accuracy: 0.9686 - val_loss: 0.0475 - val_accuracy: 0.9736
Epoch 52/70
38/38 [==============================] - 2s 49ms/step - loss: 0.0643 - accuracy: 0.9818 - val_loss: 0.0522 - val_accuracy: 0.9769
Epoch 53/70
38/38 [==============================] - 1s 39ms/step - loss: 0.0490 - accuracy: 0.9810 - val_loss: 0.0942 - val_accuracy: 0.9637
Epoch 54/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0475 - accuracy: 0.9818 - val_loss: 0.0705 - val_accuracy: 0.9604
Epoch 55/70
38/38 [==============================] - 1s 38ms/step - loss: 0.0296 - accuracy: 0.9851 - val_loss: 0.1145 - val_accuracy: 0.9637
Epoch 56/70
38/38 [==============================] - 2s 46ms/step - loss: 0.0295 - accuracy: 0.9859 - val_loss: 0.0523 - val_accuracy: 0.9670
Epoch 57/70
38/38 [==============================] - 2s 46ms/step - loss: 0.0232 - accuracy: 0.9917 - val_loss: 0.0580 - val_accuracy: 0.9703
Epoch 58/70
38/38 [==============================] - 2s 41ms/step - loss: 0.0190 - accuracy: 0.9926 - val_loss: 0.0509 - val_accuracy: 0.9736
Epoch 59/70
38/38 [==============================] - 2s 45ms/step - loss: 0.0366 - accuracy: 0.9818 - val_loss: 0.0617 - val_accuracy: 0.9703
Epoch 60/70
38/38 [==============================] - 1s 38ms/step - loss: 0.0409 - accuracy: 0.9868 - val_loss: 0.0759 - val_accuracy: 0.9736
Epoch 61/70
38/38 [==============================] - 1s 37ms/step - loss: 0.0276 - accuracy: 0.9892 - val_loss: 0.0649 - val_accuracy: 0.9670
Epoch 62/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0328 - accuracy: 0.9901 - val_loss: 0.0919 - val_accuracy: 0.9670
Epoch 63/70
38/38 [==============================] - 1s 36ms/step - loss: 0.0441 - accuracy: 0.9851 - val_loss: 0.0637 - val_accuracy: 0.9637
Epoch 64/70
38/38 [==============================] - 1s 39ms/step - loss: 0.0321 - accuracy: 0.9909 - val_loss: 0.0582 - val_accuracy: 0.9769
```

```
Epoch 65/70
38/38 [==============================] - 1s 34ms/step - loss: 0.0318 - accuracy:
0.9876 - val_loss: 0.1070 - val_accuracy: 0.9571
Epoch 66/70
38/38 [==============================] - 2s 43ms/step - loss: 0.0219 - accuracy:
0.9917 - val_loss: 0.0712 - val_accuracy: 0.9637
Epoch 67/70
38/38 [==============================] - 2s 45ms/step - loss: 0.0342 - accuracy:
0.9859 - val_loss: 0.1437 - val_accuracy: 0.9670
Epoch 68/70
38/38 [==============================] - 2s 46ms/step - loss: 0.0220 - accuracy:
0.9926 - val_loss: 0.0418 - val_accuracy: 0.9769
Epoch 69/70
38/38 [==============================] - 2s 42ms/step - loss: 0.0154 - accuracy:
0.9950 - val_loss: 0.0333 - val_accuracy: 0.9736
Epoch 70/70
38/38 [==============================] - 2s 44ms/step - loss: 0.0229 - accuracy:
0.9909 - val_loss: 0.0918 - val_accuracy: 0.9703
```

In [90]:

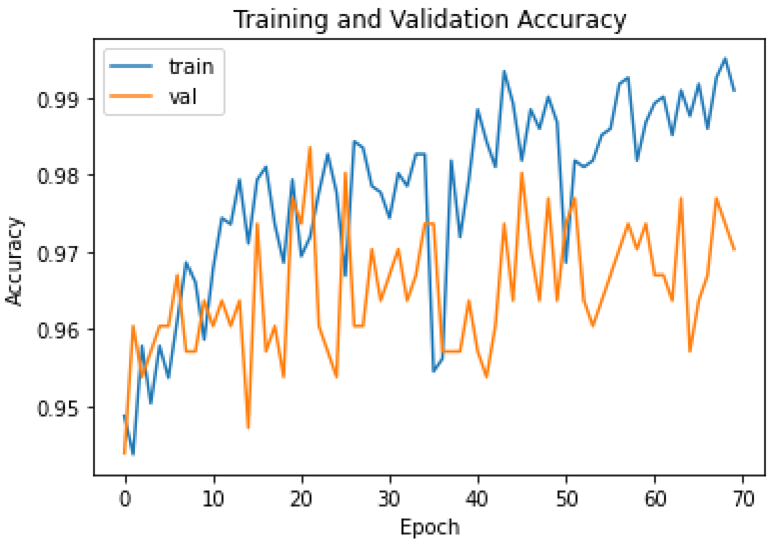
```python
import matplotlib.pyplot as plt

# 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 [91]:

```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 [92]:
```
# Predict classes for test set
y_pred = model.predict(X_test_resized)
```

```
10/10 [==============================] - 0s 10ms/step
```

In [93]:
```
# Get predicted class for each sample
y_pred_class = np.argmax(y_pred, axis=1)
```

In [94]:
```
# 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: 0.09176267683506012
Test accuracy: 0.9702970385551453
```

Save model

In [95]:
```
from keras.models import load_model

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

In [96]:
```
import numpy as np
from keras.preprocessing import image
from keras.applications.vgg16 import VGG16, preprocess_input
```

In [97]:
```
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 [98]:
```
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")
```

```
1/1 [==============================] - 0s 141ms/step
Defect detected
```

In [ ]:

In [ ]:

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 [ ]:

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
```

```python
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 [ ]:

```python

```

In [ ]:

```python

```