Project: Perform Facial Recognition with Deep Learning in Keras Using CNN

Project Description

Problem Statement: Facial recognition is a biometric alternative that measures unique characteristics of a human face. Applications available today include flight check in, tagging friends and family members in photos, and “tailored” advertising. You are a computer vision engineer who needs to develop a face recognition programme with deep convolutional neural networks.

Objective: Use a deep convolutional neural network to perform facial recognition using Keras.

Dataset Details: ORL face database composed of 400 images of size 112 x 92. There are 40 people, 10 images per person. The images were taken at different times, lighting and facial expressions. The faces are in an upright position in frontal view, with a slight left-right rotation.

Input the required libraries

In [1]:

```
# Data science libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import itertools

#Scikit-Learn libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve,auc

#Keras API Tensorflow 2 libraries
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, Activation, LeakyReLU
from keras.layers.noise import AlphaDropout
from tensorflow.keras.optimizers import Adam

from keras.utils.generic_utils import get_custom_objects
```

```
from keras import backend as K
from keras.callbacks import TensorBoard
from keras.utils.np_utils import to_categorical

print('Tensorflow version:', tf.__version__)
```

```
Tensorflow version: 2.8.0
```

In [2]:

```
df = np.load('ORL_faces (2).npz')
```

In [31]:

```
df
```

Out[31]:

```
<numpy.lib.npyio.NpzFile at 0x1caacc48490>
```

Load the dataset and preprocess the data

In [4]:

```
# Loading train and test dataset (data is already split into)
x_train = df['trainX']
y_train = df['trainY']
x_test = df['testX']
y_test = df['testY']
```

In [5]:

```
# Normalizing each image as each image is between 0-255 pixels
x_train = x_train.astype(np.float32) / 255.0
x_test = x_test.astype(np.float32) / 255.0

print('Training dataset shape: ',x_train.shape)
print('Testing dataset shape: ',x_test.shape)
```

```
Training dataset shape:  (240, 10304)
Testing dataset shape:  (160, 10304)
```

Split the dataset

Split is done from Xtrain dataset into x_train and x_valid dataset

Here we considered only 10 % of the training dataset as validation dataset as number of images overall is very low (240)

In [6]:
```
x_train, x_valid, y_train, y_valid = train_test_split(x_train,y_train,test_size=0.1,random_state=42)
```

Transform the images to equal sizes to feed in CNN

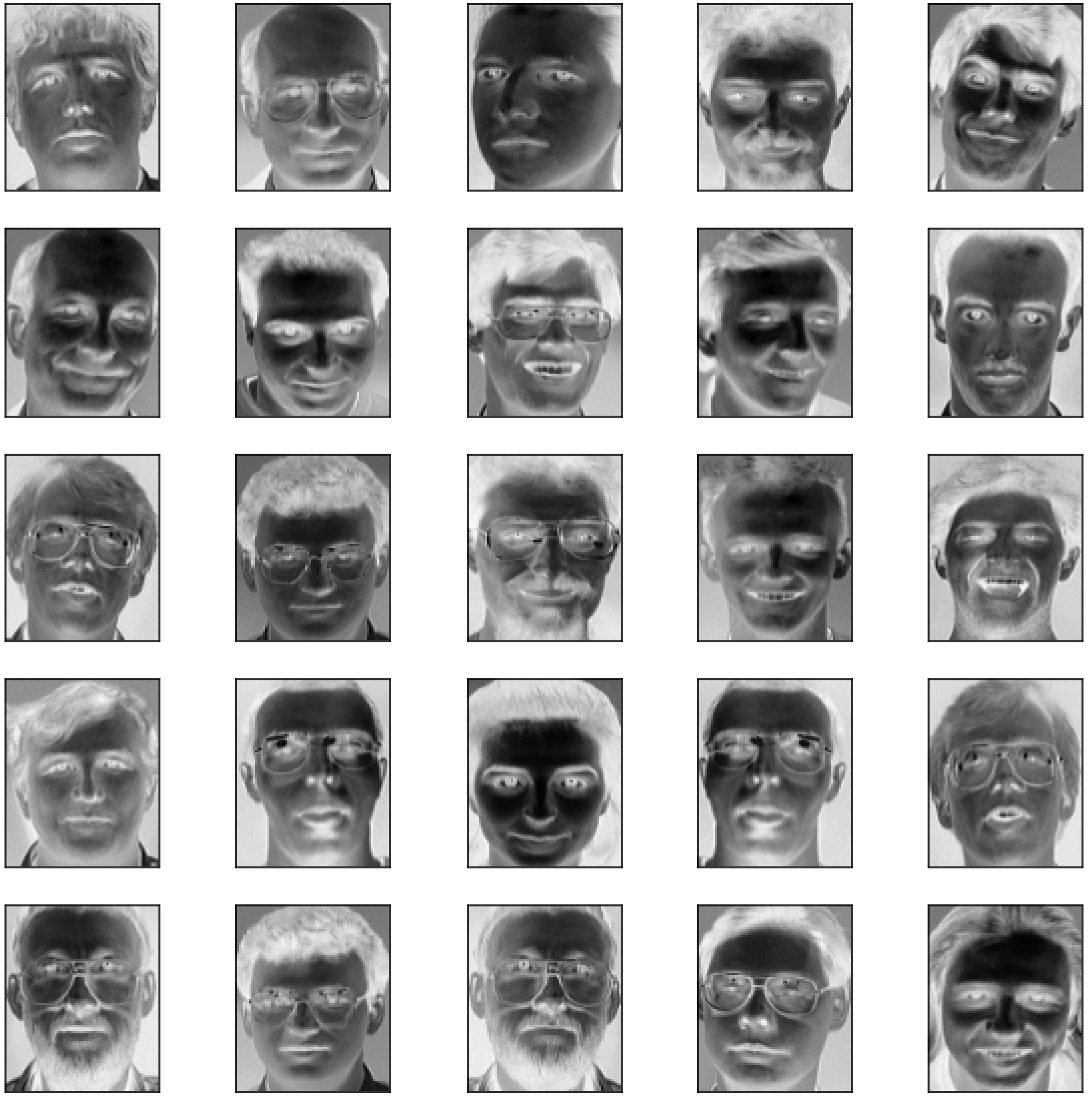
- When we feed images in CNN the size of each image must be same.
- We will define the shape of image in terms of rows, columns
- To make equal size of all images (train, test, and valid dataset), we will use Reshape function

In [7]:
```
# Shape of image definition
rows = 112
columns = 92
image_shape = (rows,columns,1)
```

In [8]:
```
# Reshape function
x_train = x_train.reshape(x_train.shape[0],*image_shape)
x_test = x_test.reshape(x_test.shape[0],*image_shape)
x_valid = x_valid.reshape(x_valid.shape[0],*image_shape)
```

Visualize images in different colormap

In [9]:
```
#visualize some images 5 x 5 grid images in gray scale
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary) # for gray scale
plt.show()
```



In [10]:

```
#visualize some inages 5 x 5 grid images in autumn
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
```

```
    plt.imshow(x_train[i],cmap=plt.cm.autumn) # for autumn
plt.show()
```

In [11]:
```
#visualize some inages 5 x 5 grid images by default
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
```

```
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
plt.show()
```

Build a CNN model that has 3 main layers:

- Convolutional Layer
- Pooling Layer
- Fully Connected Layer

The objective here is to build and train a CNN model which has accuracy above 90%. It depends upon number of iterations (Epochs) performed and what type of activation function is chosen to train the model. Before deciding the type of activation function chosen for our final model, we will train the model for different types of activation functions and then use that defined function for final prediction.

- Activation functions tested: ['sigmoid', 'relu', 'elu', 'leaky-relu', 'selu']

- For 'selu' (Scaled Exponential Linear Unit), we need to use a kernel initializer 'lecun_normal' and a special form of dropout 'AlphaDropout()'

In [12]:

```
Convolutional Layer
Pooling Layer
Fully Connected Layer
The objective here is to build and train a CNN model which has accuracy above 90%. It depends upon number of iterations (Epochs) p

Activation functions tested: ['sigmoid', 'relu', 'elu', 'leaky-relu', 'selu']
For 'selu' (Scaled Exponential Linear Unit), we need to use a kernel initializer 'lecun_normal' and a special form of dropout 'Alp
def cnn_model(activation,
              dropout_rate,
              optimizer):

    model = Sequential() #initialize Sequential model

    #we created if else version for program to 'selu' version or otheractivation functions

    if(activation == 'selu'):
        model.add(Conv2D(32, kernel_size=3,
                  activation=activation,
                  input_shape=image_shape,
                  kernel_initializer='lecun_normal')) #32 filter with kernel size of 3 with input shape
        model.add(MaxPooling2D(pool_size=2))

        model.add(Conv2D(64, 3, activation=activation,
                         kernel_initializer='lecun_normal')) #64 filter with kernel size of 3 x 3
        model.add(MaxPooling2D(pool_size=2)) #Max pool with size of 2

        model.add(Flatten())
        model.add(Dense(2024, activation=activation,
```

```
                        kernel_initializer='lecun_normal'))
        model.add(AlphaDropout(0.5))

        model.add(Dense(1024, activation=activation,
                        kernel_initializer='lecun_normal'))
        model.add(AlphaDropout(0.5))

        model.add(Dense(512, activation=activation,
                        kernel_initializer='lecun_normal'))
        model.add(AlphaDropout(0.5))

        model.add(Dense(20, activation='softmax')) #Output layer
    else:
        model.add(Conv2D(32, kernel_size=3,
                  activation=activation,
                  input_shape=image_shape)) #32 filter with kernel size of 3 x 3 with input shape
        model.add(MaxPooling2D(pool_size=2))

        model.add(Conv2D(64,3, activation=activation)) #64 filter with kernel size of 3 x 3
        model.add(MaxPooling2D(pool_size=2)) #Max pool with size of 2

        model.add(Flatten())

        model.add(Dense(2024, activation=activation))
        model.add(Dropout(0.5))
        model.add(Dense(1024, activation=activation))
        model.add(Dropout(0.5))
        model.add(Dense(512, activation=activation))
        model.add(Dropout(0.5))

        model.add(Dense(20, activation='softmax')) #Output layer

    model.compile(
        loss='sparse_categorical_crossentropy',
        optimizer=optimizer,
        metrics=['accuracy']
    ) #compile model with loss, optimizer chosen and accuracy as metrics

    return model
```

In [13]:

```
#For Leaky-Rely function we need to define aplha parameters using get_custom_objects
```

```
get_custom_objects().update({'leaky-relu': Activation(LeakyReLU(alpha=0.2))})

# Defining the type of activation functions to be tested
activation_function = ['relu', 'elu', 'leaky-relu', 'selu']
```

Building model and train for all chosen activation functions

In [14]:

```
activation_results = [] #creating an empty matrix for storing results for activations

for activation in activation_function:
    print('\nTraining with {0} activation function\n'.format(activation))

    model = cnn_model(activation=activation,
                      dropout_rate=0.2,
                      optimizer=Adam(clipvalue=0.5)) #using 'adam' optimizer with clipvalue of 0.5

    history = model.fit(np.array(x_train), np.array(y_train),
                        batch_size=512,
                        epochs=75,
                        verbose=2,
                        validation_data=(np.array(x_valid),np.array(y_valid)))

    activation_results.append(history) #store results

    K.clear_session()
    del model

print(activation_results)
```

```
Training with relu activation function

Epoch 1/75
1/1 - 2s - loss: 3.0148 - accuracy: 0.0509 - val_loss: 3.3937 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 2/75
1/1 - 2s - loss: 3.7702 - accuracy: 0.0694 - val_loss: 3.0037 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 3/75
1/1 - 2s - loss: 3.5771 - accuracy: 0.0324 - val_loss: 2.9659 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 4/75
1/1 - 2s - loss: 3.2939 - accuracy: 0.0463 - val_loss: 2.9776 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 5/75
1/1 - 2s - loss: 3.0632 - accuracy: 0.0741 - val_loss: 2.9863 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 6/75
```

```
1/1 - 2s - loss: 3.0623 - accuracy: 0.0741 - val_loss: 3.0003 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 7/75
1/1 - 2s - loss: 2.9611 - accuracy: 0.0648 - val_loss: 3.0007 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 8/75
1/1 - 2s - loss: 2.9872 - accuracy: 0.0556 - val_loss: 3.0028 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 9/75
1/1 - 2s - loss: 2.9849 - accuracy: 0.0648 - val_loss: 3.0079 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 10/75
1/1 - 2s - loss: 2.9870 - accuracy: 0.0648 - val_loss: 3.0072 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 11/75
1/1 - 2s - loss: 2.9662 - accuracy: 0.0648 - val_loss: 3.0043 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 12/75
1/1 - 2s - loss: 2.9667 - accuracy: 0.0787 - val_loss: 3.0016 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 13/75
1/1 - 2s - loss: 2.9624 - accuracy: 0.0787 - val_loss: 2.9975 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 14/75
1/1 - 2s - loss: 2.9496 - accuracy: 0.1389 - val_loss: 2.9933 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 15/75
1/1 - 2s - loss: 2.9281 - accuracy: 0.1759 - val_loss: 2.9861 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 16/75
1/1 - 2s - loss: 2.9276 - accuracy: 0.1667 - val_loss: 2.9773 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 17/75
1/1 - 2s - loss: 2.8817 - accuracy: 0.1667 - val_loss: 2.9712 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 18/75
1/1 - 2s - loss: 2.8682 - accuracy: 0.1481 - val_loss: 2.9552 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 19/75
1/1 - 2s - loss: 2.8487 - accuracy: 0.1713 - val_loss: 2.9287 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 20/75
1/1 - 2s - loss: 2.7598 - accuracy: 0.1667 - val_loss: 2.8801 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 21/75
1/1 - 2s - loss: 2.7520 - accuracy: 0.1620 - val_loss: 2.8073 - val_accuracy: 0.1667 - 2s/epoch - 2s/step
Epoch 22/75
1/1 - 2s - loss: 2.6443 - accuracy: 0.1713 - val_loss: 2.7264 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 23/75
1/1 - 2s - loss: 2.5165 - accuracy: 0.2130 - val_loss: 2.6285 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 24/75
1/1 - 2s - loss: 2.4635 - accuracy: 0.2500 - val_loss: 2.5275 - val_accuracy: 0.3750 - 2s/epoch - 2s/step
Epoch 25/75
1/1 - 2s - loss: 2.3113 - accuracy: 0.3241 - val_loss: 2.3773 - val_accuracy: 0.4583 - 2s/epoch - 2s/step
Epoch 26/75
1/1 - 2s - loss: 2.1962 - accuracy: 0.3194 - val_loss: 2.1803 - val_accuracy: 0.5833 - 2s/epoch - 2s/step
Epoch 27/75
1/1 - 2s - loss: 2.2189 - accuracy: 0.3194 - val_loss: 2.0289 - val_accuracy: 0.6667 - 2s/epoch - 2s/step
Epoch 28/75
```

```
1/1 - 2s - loss: 2.0015 - accuracy: 0.3750 - val_loss: 1.8732 - val_accuracy: 0.7083 - 2s/epoch - 2s/step
Epoch 29/75
1/1 - 2s - loss: 1.8188 - accuracy: 0.4676 - val_loss: 1.5808 - val_accuracy: 0.7500 - 2s/epoch - 2s/step
Epoch 30/75
1/1 - 2s - loss: 1.6583 - accuracy: 0.4583 - val_loss: 1.3263 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 31/75
1/1 - 2s - loss: 1.5799 - accuracy: 0.4954 - val_loss: 1.1591 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 32/75
1/1 - 2s - loss: 1.3625 - accuracy: 0.6019 - val_loss: 0.9959 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 33/75
1/1 - 2s - loss: 1.2384 - accuracy: 0.6250 - val_loss: 0.8208 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 34/75
1/1 - 2s - loss: 1.1946 - accuracy: 0.6111 - val_loss: 0.7798 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 35/75
1/1 - 2s - loss: 1.0151 - accuracy: 0.6759 - val_loss: 0.6721 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 36/75
1/1 - 2s - loss: 0.7938 - accuracy: 0.7500 - val_loss: 0.5560 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 37/75
1/1 - 2s - loss: 0.7918 - accuracy: 0.7731 - val_loss: 0.4547 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 38/75
1/1 - 2s - loss: 0.7133 - accuracy: 0.7870 - val_loss: 0.4143 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 39/75
1/1 - 2s - loss: 0.6043 - accuracy: 0.8056 - val_loss: 0.3111 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 40/75
1/1 - 2s - loss: 0.5137 - accuracy: 0.8241 - val_loss: 0.2311 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 41/75
1/1 - 2s - loss: 0.3846 - accuracy: 0.8796 - val_loss: 0.1978 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 42/75
1/1 - 2s - loss: 0.4289 - accuracy: 0.8426 - val_loss: 0.2476 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 43/75
1/1 - 2s - loss: 0.3565 - accuracy: 0.8981 - val_loss: 0.1723 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 44/75
1/1 - 3s - loss: 0.2521 - accuracy: 0.9352 - val_loss: 0.1382 - val_accuracy: 0.9583 - 3s/epoch - 3s/step
Epoch 45/75
1/1 - 3s - loss: 0.1848 - accuracy: 0.9444 - val_loss: 0.1127 - val_accuracy: 0.9583 - 3s/epoch - 3s/step
Epoch 46/75
1/1 - 3s - loss: 0.1802 - accuracy: 0.9491 - val_loss: 0.0997 - val_accuracy: 1.0000 - 3s/epoch - 3s/step
Epoch 47/75
1/1 - 2s - loss: 0.1243 - accuracy: 0.9861 - val_loss: 0.0942 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 48/75
1/1 - 2s - loss: 0.1631 - accuracy: 0.9630 - val_loss: 0.0624 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 49/75
1/1 - 2s - loss: 0.0884 - accuracy: 0.9815 - val_loss: 0.0525 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 50/75
```

```
1/1 - 2s - loss: 0.1147 - accuracy: 0.9769 - val_loss: 0.0541 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 51/75
1/1 - 2s - loss: 0.0644 - accuracy: 0.9861 - val_loss: 0.0681 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 52/75
1/1 - 2s - loss: 0.1171 - accuracy: 0.9583 - val_loss: 0.0304 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 53/75
1/1 - 2s - loss: 0.0627 - accuracy: 0.9861 - val_loss: 0.0148 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 54/75
1/1 - 2s - loss: 0.0548 - accuracy: 0.9907 - val_loss: 0.0095 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 55/75
1/1 - 2s - loss: 0.0370 - accuracy: 0.9907 - val_loss: 0.0082 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 56/75
1/1 - 2s - loss: 0.0364 - accuracy: 0.9861 - val_loss: 0.0073 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 57/75
1/1 - 2s - loss: 0.0611 - accuracy: 0.9861 - val_loss: 0.0062 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 58/75
1/1 - 2s - loss: 0.0497 - accuracy: 0.9861 - val_loss: 0.0052 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 59/75
1/1 - 2s - loss: 0.0262 - accuracy: 0.9907 - val_loss: 0.0075 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 60/75
1/1 - 2s - loss: 0.0715 - accuracy: 0.9630 - val_loss: 0.0110 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 61/75
1/1 - 2s - loss: 0.0184 - accuracy: 0.9907 - val_loss: 0.0122 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 62/75
1/1 - 2s - loss: 0.0418 - accuracy: 0.9815 - val_loss: 0.0055 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 63/75
1/1 - 2s - loss: 0.0389 - accuracy: 0.9861 - val_loss: 0.0047 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 64/75
1/1 - 2s - loss: 0.0681 - accuracy: 0.9861 - val_loss: 0.0048 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 65/75
1/1 - 2s - loss: 0.0096 - accuracy: 1.0000 - val_loss: 0.0051 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 66/75
1/1 - 2s - loss: 0.0234 - accuracy: 0.9954 - val_loss: 0.0040 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 67/75
1/1 - 2s - loss: 0.0100 - accuracy: 1.0000 - val_loss: 0.0033 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 68/75
1/1 - 2s - loss: 0.0102 - accuracy: 1.0000 - val_loss: 0.0026 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 69/75
1/1 - 2s - loss: 0.0186 - accuracy: 0.9954 - val_loss: 0.0017 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 70/75
1/1 - 2s - loss: 0.0342 - accuracy: 0.9815 - val_loss: 0.0014 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 71/75
1/1 - 2s - loss: 0.0157 - accuracy: 0.9954 - val_loss: 0.0013 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 72/75
```

```
1/1 - 2s - loss: 0.0137 - accuracy: 1.0000 - val_loss: 0.0014 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 73/75
1/1 - 2s - loss: 0.0161 - accuracy: 0.9954 - val_loss: 0.0014 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 74/75
1/1 - 2s - loss: 0.0093 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 75/75
1/1 - 2s - loss: 0.0078 - accuracy: 1.0000 - val_loss: 0.0022 - val_accuracy: 1.0000 - 2s/epoch - 2s/step

Training with elu activation function

Epoch 1/75
1/1 - 3s - loss: 3.1395 - accuracy: 0.0417 - val_loss: 6.5662 - val_accuracy: 0.0000e+00 - 3s/epoch - 3s/step
Epoch 2/75
1/1 - 2s - loss: 7.7285 - accuracy: 0.0463 - val_loss: 3.5118 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 3/75
1/1 - 2s - loss: 8.1460 - accuracy: 0.0509 - val_loss: 4.0792 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 4/75
1/1 - 2s - loss: 5.2946 - accuracy: 0.0741 - val_loss: 18.6829 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 5/75
1/1 - 2s - loss: 17.2321 - accuracy: 0.0556 - val_loss: 3.4671 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 6/75
1/1 - 2s - loss: 6.8194 - accuracy: 0.1019 - val_loss: 3.1851 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 7/75
1/1 - 2s - loss: 4.9840 - accuracy: 0.0972 - val_loss: 4.1188 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 8/75
1/1 - 2s - loss: 4.6215 - accuracy: 0.0694 - val_loss: 4.9419 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 9/75
1/1 - 2s - loss: 4.9597 - accuracy: 0.0556 - val_loss: 3.9295 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 10/75
1/1 - 2s - loss: 4.8085 - accuracy: 0.1111 - val_loss: 2.9874 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 11/75
1/1 - 2s - loss: 3.7144 - accuracy: 0.1435 - val_loss: 2.4912 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 12/75
1/1 - 2s - loss: 3.2314 - accuracy: 0.1852 - val_loss: 2.0037 - val_accuracy: 0.3750 - 2s/epoch - 2s/step
Epoch 13/75
1/1 - 2s - loss: 2.7145 - accuracy: 0.2870 - val_loss: 2.0860 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 14/75
1/1 - 2s - loss: 2.4747 - accuracy: 0.3056 - val_loss: 1.0873 - val_accuracy: 0.5833 - 2s/epoch - 2s/step
Epoch 15/75
1/1 - 2s - loss: 2.0736 - accuracy: 0.4954 - val_loss: 4.0054 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 16/75
1/1 - 2s - loss: 4.1934 - accuracy: 0.2037 - val_loss: 1.5656 - val_accuracy: 0.4583 - 2s/epoch - 2s/step
Epoch 17/75
1/1 - 2s - loss: 3.6408 - accuracy: 0.3287 - val_loss: 1.6573 - val_accuracy: 0.4583 - 2s/epoch - 2s/step
```

```
Epoch 18/75
1/1 - 2s - loss: 3.1310 - accuracy: 0.3935 - val_loss: 1.2098 - val_accuracy: 0.6250 - 2s/epoch - 2s/step
Epoch 19/75
1/1 - 2s - loss: 2.4122 - accuracy: 0.3380 - val_loss: 1.0619 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 20/75
1/1 - 2s - loss: 2.1973 - accuracy: 0.3657 - val_loss: 1.0494 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 21/75
1/1 - 2s - loss: 1.4712 - accuracy: 0.5370 - val_loss: 1.5981 - val_accuracy: 0.5417 - 2s/epoch - 2s/step
Epoch 22/75
1/1 - 3s - loss: 1.8391 - accuracy: 0.4491 - val_loss: 1.4180 - val_accuracy: 0.5833 - 3s/epoch - 3s/step
Epoch 23/75
1/1 - 2s - loss: 1.1675 - accuracy: 0.6389 - val_loss: 1.7370 - val_accuracy: 0.5000 - 2s/epoch - 2s/step
Epoch 24/75
1/1 - 3s - loss: 1.6033 - accuracy: 0.5741 - val_loss: 1.0961 - val_accuracy: 0.6250 - 3s/epoch - 3s/step
Epoch 25/75
1/1 - 2s - loss: 0.8567 - accuracy: 0.7454 - val_loss: 0.9743 - val_accuracy: 0.5417 - 2s/epoch - 2s/step
Epoch 26/75
1/1 - 2s - loss: 0.8191 - accuracy: 0.7315 - val_loss: 0.4657 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 27/75
1/1 - 2s - loss: 0.8758 - accuracy: 0.7269 - val_loss: 0.3662 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 28/75
1/1 - 2s - loss: 0.6058 - accuracy: 0.8287 - val_loss: 0.5919 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 29/75
1/1 - 2s - loss: 0.5043 - accuracy: 0.8241 - val_loss: 0.5809 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 30/75
1/1 - 2s - loss: 0.4371 - accuracy: 0.8843 - val_loss: 0.5947 - val_accuracy: 0.7500 - 2s/epoch - 2s/step
Epoch 31/75
1/1 - 2s - loss: 0.4902 - accuracy: 0.8565 - val_loss: 0.4316 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 32/75
1/1 - 2s - loss: 0.3823 - accuracy: 0.8889 - val_loss: 0.4202 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 33/75
1/1 - 2s - loss: 0.3849 - accuracy: 0.8519 - val_loss: 0.1991 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 34/75
1/1 - 2s - loss: 0.2301 - accuracy: 0.9491 - val_loss: 0.2683 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 35/75
1/1 - 2s - loss: 0.3717 - accuracy: 0.8843 - val_loss: 0.3885 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 36/75
1/1 - 2s - loss: 0.3255 - accuracy: 0.8935 - val_loss: 0.2889 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 37/75
1/1 - 2s - loss: 0.1847 - accuracy: 0.9306 - val_loss: 0.2214 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 38/75
1/1 - 2s - loss: 0.1593 - accuracy: 0.9583 - val_loss: 0.2595 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 39/75
1/1 - 2s - loss: 0.1666 - accuracy: 0.9676 - val_loss: 0.2300 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
```

```
Epoch 40/75
1/1 - 2s - loss: 0.0961 - accuracy: 0.9676 - val_loss: 0.3127 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 41/75
1/1 - 2s - loss: 0.1276 - accuracy: 0.9722 - val_loss: 0.3117 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 42/75
1/1 - 2s - loss: 0.0917 - accuracy: 0.9722 - val_loss: 0.2377 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 43/75
1/1 - 2s - loss: 0.1064 - accuracy: 0.9630 - val_loss: 0.1295 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 44/75
1/1 - 2s - loss: 0.0596 - accuracy: 0.9815 - val_loss: 0.0525 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 45/75
1/1 - 2s - loss: 0.0416 - accuracy: 0.9907 - val_loss: 0.0342 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 46/75
1/1 - 2s - loss: 0.0677 - accuracy: 0.9769 - val_loss: 0.0336 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 47/75
1/1 - 2s - loss: 0.0637 - accuracy: 0.9861 - val_loss: 0.0391 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 48/75
1/1 - 2s - loss: 0.0546 - accuracy: 0.9815 - val_loss: 0.0392 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 49/75
1/1 - 2s - loss: 0.0278 - accuracy: 0.9907 - val_loss: 0.0259 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 50/75
1/1 - 2s - loss: 0.0459 - accuracy: 0.9861 - val_loss: 0.0409 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 51/75
1/1 - 2s - loss: 0.0487 - accuracy: 0.9861 - val_loss: 0.0673 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 52/75
1/1 - 2s - loss: 0.0181 - accuracy: 0.9954 - val_loss: 0.0537 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 53/75
1/1 - 2s - loss: 0.0193 - accuracy: 0.9954 - val_loss: 0.0303 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 54/75
1/1 - 2s - loss: 0.0251 - accuracy: 0.9861 - val_loss: 0.0225 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 55/75
1/1 - 2s - loss: 0.0117 - accuracy: 1.0000 - val_loss: 0.0214 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 56/75
1/1 - 2s - loss: 0.0083 - accuracy: 1.0000 - val_loss: 0.0256 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 57/75
1/1 - 2s - loss: 0.0089 - accuracy: 1.0000 - val_loss: 0.0336 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 58/75
1/1 - 2s - loss: 0.0189 - accuracy: 0.9954 - val_loss: 0.0363 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 59/75
1/1 - 2s - loss: 0.0193 - accuracy: 0.9907 - val_loss: 0.0423 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 60/75
1/1 - 2s - loss: 0.0190 - accuracy: 0.9954 - val_loss: 0.0509 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 61/75
1/1 - 2s - loss: 0.0066 - accuracy: 1.0000 - val_loss: 0.0540 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
```

```
Epoch 62/75
1/1 - 2s - loss: 0.0117 - accuracy: 1.0000 - val_loss: 0.0355 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 63/75
1/1 - 2s - loss: 0.0050 - accuracy: 1.0000 - val_loss: 0.0200 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 64/75
1/1 - 2s - loss: 0.0060 - accuracy: 1.0000 - val_loss: 0.0140 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 65/75
1/1 - 2s - loss: 0.0061 - accuracy: 1.0000 - val_loss: 0.0135 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 66/75
1/1 - 2s - loss: 0.0063 - accuracy: 1.0000 - val_loss: 0.0138 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 67/75
1/1 - 2s - loss: 0.0086 - accuracy: 1.0000 - val_loss: 0.0121 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 68/75
1/1 - 2s - loss: 0.0080 - accuracy: 1.0000 - val_loss: 0.0164 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 69/75
1/1 - 2s - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.0309 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 70/75
1/1 - 2s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 0.0420 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 71/75
1/1 - 2s - loss: 0.0091 - accuracy: 1.0000 - val_loss: 0.0228 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 72/75
1/1 - 2s - loss: 0.0053 - accuracy: 1.0000 - val_loss: 0.0100 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 73/75
1/1 - 2s - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.0069 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 74/75
1/1 - 2s - loss: 0.0089 - accuracy: 0.9954 - val_loss: 0.0084 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 75/75
1/1 - 2s - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.0121 - val_accuracy: 1.0000 - 2s/epoch - 2s/step

Training with leaky-relu activation function

Epoch 1/75
1/1 - 4s - loss: 3.0182 - accuracy: 0.0324 - val_loss: 3.7627 - val_accuracy: 0.0000e+00 - 4s/epoch - 4s/step
Epoch 2/75
1/1 - 3s - loss: 5.1099 - accuracy: 0.0556 - val_loss: 3.4814 - val_accuracy: 0.0417 - 3s/epoch - 3s/step
Epoch 3/75
1/1 - 2s - loss: 5.3761 - accuracy: 0.0833 - val_loss: 2.8619 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 4/75
1/1 - 2s - loss: 3.6540 - accuracy: 0.0787 - val_loss: 2.9637 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 5/75
1/1 - 2s - loss: 3.0478 - accuracy: 0.0787 - val_loss: 2.9330 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 6/75
1/1 - 2s - loss: 2.9081 - accuracy: 0.1157 - val_loss: 3.1532 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 7/75
```

```
1/1 - 2s - loss: 4.8119 - accuracy: 0.0787 - val_loss: 2.9465 - val_accuracy: 0.2917 - 2s/epoch - 2s/step
Epoch 8/75
1/1 - 2s - loss: 3.6157 - accuracy: 0.1111 - val_loss: 2.8991 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 9/75
1/1 - 2s - loss: 2.9630 - accuracy: 0.0880 - val_loss: 2.9330 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 10/75
1/1 - 2s - loss: 2.8714 - accuracy: 0.1157 - val_loss: 2.9234 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 11/75
1/1 - 2s - loss: 2.9516 - accuracy: 0.1296 - val_loss: 2.8649 - val_accuracy: 0.1250 - 2s/epoch - 2s/step
Epoch 12/75
1/1 - 2s - loss: 2.8392 - accuracy: 0.1528 - val_loss: 2.7876 - val_accuracy: 0.1250 - 2s/epoch - 2s/step
Epoch 13/75
1/1 - 2s - loss: 2.8044 - accuracy: 0.1667 - val_loss: 2.7068 - val_accuracy: 0.2500 - 2s/epoch - 2s/step
Epoch 14/75
1/1 - 2s - loss: 2.7580 - accuracy: 0.1806 - val_loss: 2.6399 - val_accuracy: 0.2917 - 2s/epoch - 2s/step
Epoch 15/75
1/1 - 2s - loss: 2.5943 - accuracy: 0.2130 - val_loss: 2.5853 - val_accuracy: 0.2500 - 2s/epoch - 2s/step
Epoch 16/75
1/1 - 2s - loss: 2.4404 - accuracy: 0.3056 - val_loss: 2.5250 - val_accuracy: 0.5000 - 2s/epoch - 2s/step
Epoch 17/75
1/1 - 2s - loss: 2.3718 - accuracy: 0.3519 - val_loss: 2.4230 - val_accuracy: 0.5417 - 2s/epoch - 2s/step
Epoch 18/75
1/1 - 2s - loss: 2.2434 - accuracy: 0.3519 - val_loss: 2.2645 - val_accuracy: 0.6250 - 2s/epoch - 2s/step
Epoch 19/75
1/1 - 2s - loss: 2.0157 - accuracy: 0.4167 - val_loss: 2.0437 - val_accuracy: 0.7083 - 2s/epoch - 2s/step
Epoch 20/75
1/1 - 2s - loss: 1.8759 - accuracy: 0.4769 - val_loss: 1.8088 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 21/75
1/1 - 2s - loss: 1.7284 - accuracy: 0.5324 - val_loss: 1.5877 - val_accuracy: 0.7500 - 2s/epoch - 2s/step
Epoch 22/75
1/1 - 2s - loss: 1.5189 - accuracy: 0.5741 - val_loss: 1.3752 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 23/75
1/1 - 2s - loss: 1.2918 - accuracy: 0.6481 - val_loss: 1.1845 - val_accuracy: 0.8333 - 2s/epoch - 2s/step
Epoch 24/75
1/1 - 2s - loss: 1.1300 - accuracy: 0.6713 - val_loss: 1.0015 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 25/75
1/1 - 2s - loss: 1.0626 - accuracy: 0.6991 - val_loss: 0.8561 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 26/75
1/1 - 2s - loss: 0.8315 - accuracy: 0.8009 - val_loss: 0.6843 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 27/75
1/1 - 2s - loss: 0.7342 - accuracy: 0.8148 - val_loss: 0.5593 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 28/75
1/1 - 2s - loss: 0.5683 - accuracy: 0.8611 - val_loss: 0.4627 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 29/75
```

```
1/1 - 2s - loss: 0.5429 - accuracy: 0.8611 - val_loss: 0.3904 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 30/75
1/1 - 2s - loss: 0.3581 - accuracy: 0.9028 - val_loss: 0.3101 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 31/75
1/1 - 2s - loss: 0.3888 - accuracy: 0.9074 - val_loss: 0.3215 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 32/75
1/1 - 2s - loss: 0.3184 - accuracy: 0.9167 - val_loss: 0.2011 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 33/75
1/1 - 2s - loss: 0.2380 - accuracy: 0.9259 - val_loss: 0.1846 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 34/75
1/1 - 2s - loss: 0.2457 - accuracy: 0.9213 - val_loss: 0.1357 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 35/75
1/1 - 2s - loss: 0.1902 - accuracy: 0.9306 - val_loss: 0.1103 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 36/75
1/1 - 2s - loss: 0.1126 - accuracy: 0.9630 - val_loss: 0.0870 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 37/75
1/1 - 2s - loss: 0.0747 - accuracy: 0.9907 - val_loss: 0.0758 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 38/75
1/1 - 2s - loss: 0.0838 - accuracy: 0.9722 - val_loss: 0.0586 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 39/75
1/1 - 2s - loss: 0.1050 - accuracy: 0.9676 - val_loss: 0.0526 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 40/75
1/1 - 2s - loss: 0.0619 - accuracy: 0.9815 - val_loss: 0.0918 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 41/75
1/1 - 2s - loss: 0.0479 - accuracy: 0.9907 - val_loss: 0.0678 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 42/75
1/1 - 2s - loss: 0.0391 - accuracy: 0.9907 - val_loss: 0.0286 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 43/75
1/1 - 2s - loss: 0.0254 - accuracy: 1.0000 - val_loss: 0.0162 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 44/75
1/1 - 2s - loss: 0.0310 - accuracy: 0.9861 - val_loss: 0.0220 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 45/75
1/1 - 2s - loss: 0.0329 - accuracy: 0.9907 - val_loss: 0.0306 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 46/75
1/1 - 2s - loss: 0.0510 - accuracy: 0.9861 - val_loss: 0.0164 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 47/75
1/1 - 2s - loss: 0.0499 - accuracy: 0.9907 - val_loss: 0.0088 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 48/75
1/1 - 2s - loss: 0.0111 - accuracy: 1.0000 - val_loss: 0.0181 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 49/75
1/1 - 2s - loss: 0.0219 - accuracy: 0.9954 - val_loss: 0.0233 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 50/75
1/1 - 2s - loss: 0.0260 - accuracy: 0.9954 - val_loss: 0.0190 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 51/75
```

```
1/1 - 2s - loss: 0.0220 - accuracy: 0.9954 - val_loss: 0.0172 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 52/75
1/1 - 2s - loss: 0.0131 - accuracy: 0.9954 - val_loss: 0.0397 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 53/75
1/1 - 2s - loss: 0.0160 - accuracy: 1.0000 - val_loss: 0.0602 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 54/75
1/1 - 2s - loss: 0.0168 - accuracy: 0.9954 - val_loss: 0.0439 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 55/75
1/1 - 2s - loss: 0.0311 - accuracy: 0.9907 - val_loss: 0.0225 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 56/75
1/1 - 2s - loss: 0.0153 - accuracy: 0.9907 - val_loss: 0.0550 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 57/75
1/1 - 2s - loss: 0.0152 - accuracy: 0.9954 - val_loss: 0.0773 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 58/75
1/1 - 2s - loss: 0.0141 - accuracy: 1.0000 - val_loss: 0.0469 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 59/75
1/1 - 2s - loss: 0.0204 - accuracy: 0.9954 - val_loss: 0.0055 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 60/75
1/1 - 2s - loss: 0.0061 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 61/75
1/1 - 2s - loss: 0.0160 - accuracy: 0.9907 - val_loss: 0.0021 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 62/75
1/1 - 2s - loss: 0.0106 - accuracy: 0.9954 - val_loss: 0.0048 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 63/75
1/1 - 2s - loss: 0.0116 - accuracy: 1.0000 - val_loss: 0.0135 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 64/75
1/1 - 2s - loss: 0.0106 - accuracy: 0.9954 - val_loss: 0.0242 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 65/75
1/1 - 2s - loss: 0.0059 - accuracy: 1.0000 - val_loss: 0.0362 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 66/75
1/1 - 2s - loss: 0.0069 - accuracy: 0.9954 - val_loss: 0.0624 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 67/75
1/1 - 2s - loss: 0.0088 - accuracy: 1.0000 - val_loss: 0.0372 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 68/75
1/1 - 2s - loss: 0.0068 - accuracy: 1.0000 - val_loss: 0.0128 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 69/75
1/1 - 2s - loss: 0.0064 - accuracy: 1.0000 - val_loss: 0.0041 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 70/75
1/1 - 2s - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.0029 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 71/75
1/1 - 2s - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.0028 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 72/75
1/1 - 2s - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.0032 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 73/75
```

```
1/1 - 2s - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.0040 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 74/75
1/1 - 2s - loss: 0.0033 - accuracy: 1.0000 - val_loss: 0.0046 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 75/75
1/1 - 2s - loss: 0.0081 - accuracy: 1.0000 - val_loss: 0.0028 - val_accuracy: 1.0000 - 2s/epoch - 2s/step

Training with selu activation function

Epoch 1/75
1/1 - 3s - loss: 3.6956 - accuracy: 0.0556 - val_loss: 29.1731 - val_accuracy: 0.0000e+00 - 3s/epoch - 3s/step
Epoch 2/75
1/1 - 2s - loss: 6.3088 - accuracy: 0.0417 - val_loss: 12.3740 - val_accuracy: 0.1250 - 2s/epoch - 2s/step
Epoch 3/75
1/1 - 2s - loss: 6.6593 - accuracy: 0.0556 - val_loss: 10.0289 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 4/75
1/1 - 2s - loss: 5.6990 - accuracy: 0.0509 - val_loss: 10.7807 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 5/75
1/1 - 2s - loss: 4.1549 - accuracy: 0.0741 - val_loss: 8.8263 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 6/75
1/1 - 2s - loss: 3.6017 - accuracy: 0.0463 - val_loss: 32.7174 - val_accuracy: 0.1250 - 2s/epoch - 2s/step
Epoch 7/75
1/1 - 2s - loss: 9.3214 - accuracy: 0.0556 - val_loss: 16.5631 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 8/75
1/1 - 2s - loss: 4.7474 - accuracy: 0.0509 - val_loss: 6.9918 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 9/75
1/1 - 2s - loss: 3.9066 - accuracy: 0.0370 - val_loss: 7.9381 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 10/75
1/1 - 2s - loss: 4.0568 - accuracy: 0.0926 - val_loss: 8.3966 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 11/75
1/1 - 2s - loss: 3.8730 - accuracy: 0.0741 - val_loss: 6.3244 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 12/75
1/1 - 2s - loss: 3.8658 - accuracy: 0.0880 - val_loss: 5.1341 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 13/75
1/1 - 2s - loss: 3.6276 - accuracy: 0.0741 - val_loss: 5.0156 - val_accuracy: 0.1250 - 2s/epoch - 2s/step
Epoch 14/75
1/1 - 2s - loss: 3.7671 - accuracy: 0.0787 - val_loss: 4.2942 - val_accuracy: 0.1250 - 2s/epoch - 2s/step
Epoch 15/75
1/1 - 2s - loss: 3.5442 - accuracy: 0.0741 - val_loss: 3.7269 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 16/75
1/1 - 2s - loss: 3.4444 - accuracy: 0.0833 - val_loss: 3.2002 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 17/75
1/1 - 2s - loss: 3.3748 - accuracy: 0.0926 - val_loss: 3.3771 - val_accuracy: 0.1667 - 2s/epoch - 2s/step
Epoch 18/75
1/1 - 2s - loss: 3.3524 - accuracy: 0.1019 - val_loss: 3.5439 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
```

```
Epoch 19/75
1/1 - 2s - loss: 3.3234 - accuracy: 0.0648 - val_loss: 3.4980 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 20/75
1/1 - 2s - loss: 3.2832 - accuracy: 0.0880 - val_loss: 3.3848 - val_accuracy: 0.2500 - 2s/epoch - 2s/step
Epoch 21/75
1/1 - 2s - loss: 3.0803 - accuracy: 0.1157 - val_loss: 3.3353 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 22/75
1/1 - 2s - loss: 3.1131 - accuracy: 0.0741 - val_loss: 3.2527 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 23/75
1/1 - 2s - loss: 3.0027 - accuracy: 0.1528 - val_loss: 3.2933 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 24/75
1/1 - 2s - loss: 2.9487 - accuracy: 0.1343 - val_loss: 3.3029 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 25/75
1/1 - 2s - loss: 2.7328 - accuracy: 0.1528 - val_loss: 3.1287 - val_accuracy: 0.0833 - 2s/epoch - 2s/step
Epoch 26/75
1/1 - 2s - loss: 2.7515 - accuracy: 0.1620 - val_loss: 2.7467 - val_accuracy: 0.0417 - 2s/epoch - 2s/step
Epoch 27/75
1/1 - 2s - loss: 2.5718 - accuracy: 0.2269 - val_loss: 2.6015 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 28/75
1/1 - 2s - loss: 2.6237 - accuracy: 0.2222 - val_loss: 2.7218 - val_accuracy: 0.2500 - 2s/epoch - 2s/step
Epoch 29/75
1/1 - 2s - loss: 2.4836 - accuracy: 0.2824 - val_loss: 2.5472 - val_accuracy: 0.2917 - 2s/epoch - 2s/step
Epoch 30/75
1/1 - 2s - loss: 2.3757 - accuracy: 0.2546 - val_loss: 2.4189 - val_accuracy: 0.2917 - 2s/epoch - 2s/step
Epoch 31/75
1/1 - 2s - loss: 2.3611 - accuracy: 0.2500 - val_loss: 2.4067 - val_accuracy: 0.2500 - 2s/epoch - 2s/step
Epoch 32/75
1/1 - 2s - loss: 2.0765 - accuracy: 0.3287 - val_loss: 2.3196 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 33/75
1/1 - 2s - loss: 2.2231 - accuracy: 0.3056 - val_loss: 1.9487 - val_accuracy: 0.2083 - 2s/epoch - 2s/step
Epoch 34/75
1/1 - 2s - loss: 1.9419 - accuracy: 0.3750 - val_loss: 1.6367 - val_accuracy: 0.3750 - 2s/epoch - 2s/step
Epoch 35/75
1/1 - 2s - loss: 1.8801 - accuracy: 0.4167 - val_loss: 1.2355 - val_accuracy: 0.6250 - 2s/epoch - 2s/step
Epoch 36/75
1/1 - 2s - loss: 1.8843 - accuracy: 0.4259 - val_loss: 1.1349 - val_accuracy: 0.7083 - 2s/epoch - 2s/step
Epoch 37/75
1/1 - 2s - loss: 1.7156 - accuracy: 0.4676 - val_loss: 1.3733 - val_accuracy: 0.6667 - 2s/epoch - 2s/step
Epoch 38/75
1/1 - 2s - loss: 1.6278 - accuracy: 0.4583 - val_loss: 1.3175 - val_accuracy: 0.6667 - 2s/epoch - 2s/step
Epoch 39/75
1/1 - 2s - loss: 1.4153 - accuracy: 0.5648 - val_loss: 0.9651 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 40/75
1/1 - 2s - loss: 1.3226 - accuracy: 0.5880 - val_loss: 1.1879 - val_accuracy: 0.7500 - 2s/epoch - 2s/step
```

```
Epoch 41/75
1/1 - 2s - loss: 1.4059 - accuracy: 0.5324 - val_loss: 1.8593 - val_accuracy: 0.5417 - 2s/epoch - 2s/step
Epoch 42/75
1/1 - 2s - loss: 1.2518 - accuracy: 0.5972 - val_loss: 1.0506 - val_accuracy: 0.7500 - 2s/epoch - 2s/step
Epoch 43/75
1/1 - 3s - loss: 1.1190 - accuracy: 0.6620 - val_loss: 0.8599 - val_accuracy: 0.8750 - 3s/epoch - 3s/step
Epoch 44/75
1/1 - 2s - loss: 1.0549 - accuracy: 0.6389 - val_loss: 1.0910 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 45/75
1/1 - 2s - loss: 0.9549 - accuracy: 0.7037 - val_loss: 0.6572 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 46/75
1/1 - 2s - loss: 0.9314 - accuracy: 0.6852 - val_loss: 0.5861 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 47/75
1/1 - 2s - loss: 0.8446 - accuracy: 0.7269 - val_loss: 0.8417 - val_accuracy: 0.7917 - 2s/epoch - 2s/step
Epoch 48/75
1/1 - 2s - loss: 0.7744 - accuracy: 0.7315 - val_loss: 0.5861 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 49/75
1/1 - 2s - loss: 0.6489 - accuracy: 0.8148 - val_loss: 0.6234 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 50/75
1/1 - 2s - loss: 0.5657 - accuracy: 0.8148 - val_loss: 0.9869 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 51/75
1/1 - 2s - loss: 0.5750 - accuracy: 0.8102 - val_loss: 0.8019 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 52/75
1/1 - 2s - loss: 0.5581 - accuracy: 0.8148 - val_loss: 0.9652 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 53/75
1/1 - 2s - loss: 0.5901 - accuracy: 0.8148 - val_loss: 0.4539 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 54/75
1/1 - 2s - loss: 0.4946 - accuracy: 0.8472 - val_loss: 0.3165 - val_accuracy: 0.8750 - 2s/epoch - 2s/step
Epoch 55/75
1/1 - 2s - loss: 0.4237 - accuracy: 0.8611 - val_loss: 0.6295 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 56/75
1/1 - 2s - loss: 0.4190 - accuracy: 0.8611 - val_loss: 0.7331 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 57/75
1/1 - 2s - loss: 0.3881 - accuracy: 0.8657 - val_loss: 0.2273 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 58/75
1/1 - 3s - loss: 0.3233 - accuracy: 0.8889 - val_loss: 0.2367 - val_accuracy: 0.9583 - 3s/epoch - 3s/step
Epoch 59/75
1/1 - 2s - loss: 0.3125 - accuracy: 0.8889 - val_loss: 0.2932 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 60/75
1/1 - 2s - loss: 0.2181 - accuracy: 0.9398 - val_loss: 0.4858 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 61/75
1/1 - 2s - loss: 0.2337 - accuracy: 0.9398 - val_loss: 0.4873 - val_accuracy: 0.9167 - 2s/epoch - 2s/step
Epoch 62/75
1/1 - 2s - loss: 0.2193 - accuracy: 0.9352 - val_loss: 0.2050 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
```

```
Epoch 63/75
1/1 - 2s - loss: 0.2015 - accuracy: 0.9398 - val_loss: 0.1699 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 64/75
1/1 - 2s - loss: 0.2334 - accuracy: 0.9028 - val_loss: 0.0915 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 65/75
1/1 - 2s - loss: 0.1459 - accuracy: 0.9769 - val_loss: 0.0542 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 66/75
1/1 - 2s - loss: 0.1326 - accuracy: 0.9722 - val_loss: 0.0348 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 67/75
1/1 - 2s - loss: 0.1681 - accuracy: 0.9491 - val_loss: 0.0014 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 68/75
1/1 - 2s - loss: 0.1217 - accuracy: 0.9630 - val_loss: 2.6234e-05 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 69/75
1/1 - 2s - loss: 0.1036 - accuracy: 0.9630 - val_loss: 0.0036 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 70/75
1/1 - 2s - loss: 0.0823 - accuracy: 0.9815 - val_loss: 0.1490 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 71/75
1/1 - 2s - loss: 0.0635 - accuracy: 0.9861 - val_loss: 0.2543 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 72/75
1/1 - 2s - loss: 0.0611 - accuracy: 0.9907 - val_loss: 0.1664 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 73/75
1/1 - 2s - loss: 0.0605 - accuracy: 0.9907 - val_loss: 0.0427 - val_accuracy: 0.9583 - 2s/epoch - 2s/step
Epoch 74/75
1/1 - 2s - loss: 0.0445 - accuracy: 1.0000 - val_loss: 0.0032 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 75/75
1/1 - 2s - loss: 0.0707 - accuracy: 0.9907 - val_loss: 5.5756e-05 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
[<keras.callbacks.History object at 0x000001CABA6A6CD0>, <keras.callbacks.History object at 0x000001CABBB8AF10>, <keras.callbacks.
History object at 0x000001CABBE9BDC0>, <keras.callbacks.History object at 0x000001CABBB66FA0>]
```

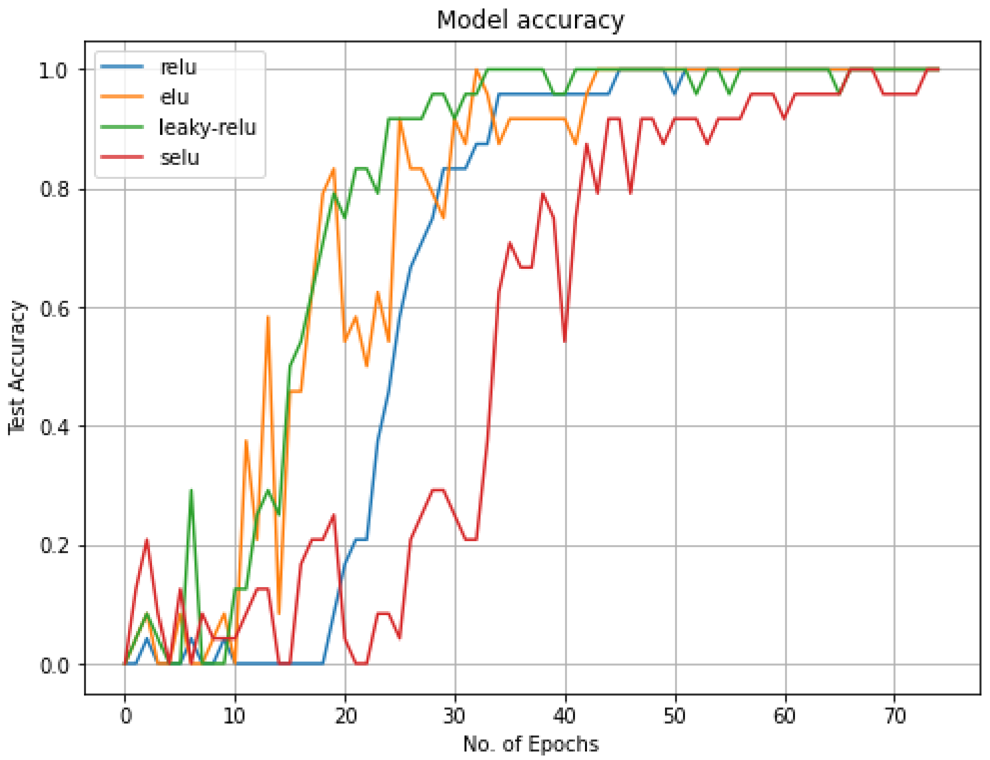
In [17]:

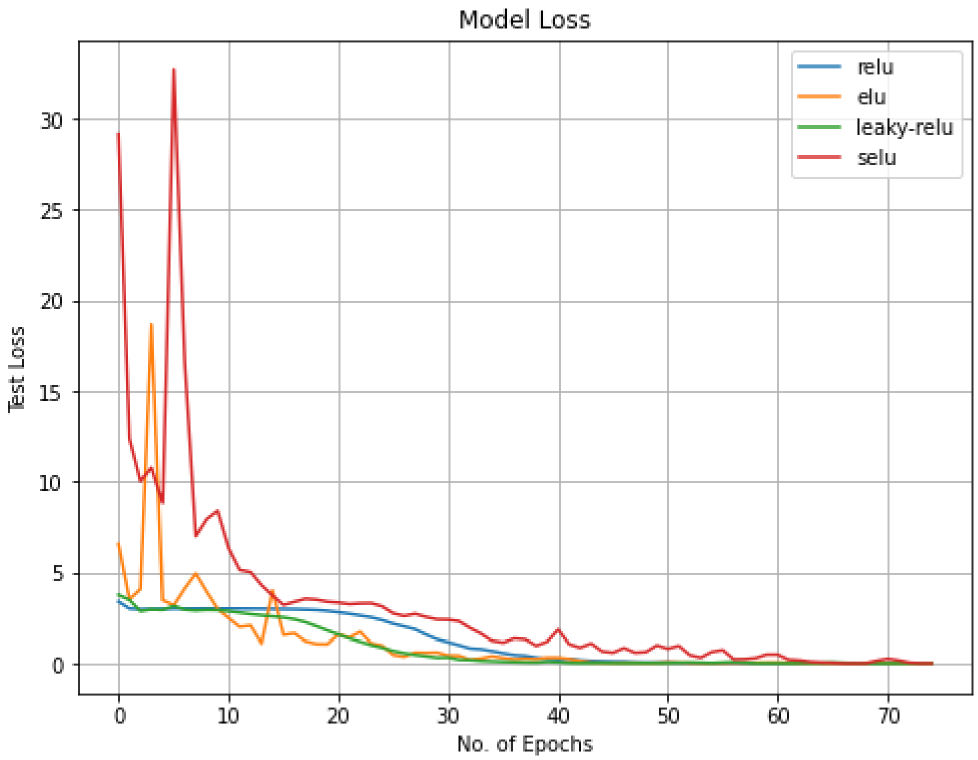
```
# Lets try to plot the Model accuracy and Model loss for each activation function used above
# Just to make sure, we don't change the above data, so we store it in new matrix

activation_list = activation_function[0:]
results_new = activation_results[0:]

def plot_results(activation_results,activation_functions_new =[]):

    plt.figure(figsize=(8,6))

    # Model accuracy values plot
    for activation_function in activation_results:
        plt.plot(activation_function.history['val_accuracy'])
```

```
    plt.title('Model accuracy')
    plt.ylabel('Test Accuracy')
    plt.xlabel('No. of Epochs')
    plt.legend(activation_functions_new)
    plt.grid()
    plt.show()

    # Model loss values plot

    plt.figure(figsize=(8,6))

    for activation_function in activation_results:
        plt.plot(activation_function.history['val_loss'])

    plt.title('Model Loss')
    plt.ylabel('Test Loss')
    plt.xlabel('No. of Epochs')
    plt.legend(activation_functions_new)
    plt.grid()
    plt.show()
```

In [18]:

```
plot_results(results_new, activation_list)
```


Model accuracy
relu
elu
leaky-relu
selu
Test Accuracy
No. of Epochs
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70



Here it is seen that **'leaky-relu'** and **'relu'** both perform well with minimum loss at lower epochs as compared to other activation functions

Looking at the plots above all activation functions converge with minimum loss and high accuracy at training and validation set but **'leaky-relu'** is able to converge for higher accuracy at lower epochs with minimum loss, so we choose **'leaky-relu'** for final model training and plotting results.

In [19]:

```
activation_func_final ='leaky-relu'

model_final = cnn_model(activation=activation_func_final,
                    dropout_rate=0.2,
                    optimizer=Adam(clipvalue=0.5)) #using 'adam' optimizer with clipvalue of 0.5

history_final = model_final.fit(np.array(x_train), np.array(y_train),
                    batch_size=512,
                    epochs=75,
                    verbose=2,
                    validation_data=(np.array(x_valid),np.array(y_valid)))
```

```
Epoch 1/75
1/1 - 6s - loss: 3.0117 - accuracy: 0.0556 - val_loss: 3.3707 - val_accuracy: 0.0000e+00 - 6s/epoch - 6s/step
Epoch 2/75
1/1 - 6s - loss: 3.8684 - accuracy: 0.0463 - val_loss: 3.2134 - val_accuracy: 0.0000e+00 - 6s/epoch - 6s/step
Epoch 3/75
1/1 - 4s - loss: 4.0588 - accuracy: 0.1111 - val_loss: 3.2333 - val_accuracy: 0.0000e+00 - 4s/epoch - 4s/step
Epoch 4/75
1/1 - 4s - loss: 3.4661 - accuracy: 0.0741 - val_loss: 3.0045 - val_accuracy: 0.0417 - 4s/epoch - 4s/step
Epoch 5/75
1/1 - 4s - loss: 3.0125 - accuracy: 0.0833 - val_loss: 2.9058 - val_accuracy: 0.2083 - 4s/epoch - 4s/step
Epoch 6/75
1/1 - 4s - loss: 2.8633 - accuracy: 0.1343 - val_loss: 2.9110 - val_accuracy: 0.2500 - 4s/epoch - 4s/step
Epoch 7/75
1/1 - 5s - loss: 3.4702 - accuracy: 0.0972 - val_loss: 2.8493 - val_accuracy: 0.1250 - 5s/epoch - 5s/step
Epoch 8/75
1/1 - 4s - loss: 2.9254 - accuracy: 0.1157 - val_loss: 2.8402 - val_accuracy: 0.1250 - 4s/epoch - 4s/step
Epoch 9/75
1/1 - 3s - loss: 2.6751 - accuracy: 0.2083 - val_loss: 2.8437 - val_accuracy: 0.1250 - 3s/epoch - 3s/step
Epoch 10/75
1/1 - 4s - loss: 2.6975 - accuracy: 0.2176 - val_loss: 2.7949 - val_accuracy: 0.0833 - 4s/epoch - 4s/step
Epoch 11/75
1/1 - 4s - loss: 2.6711 - accuracy: 0.2315 - val_loss: 2.7173 - val_accuracy: 0.1667 - 4s/epoch - 4s/step
Epoch 12/75
1/1 - 4s - loss: 2.5661 - accuracy: 0.2361 - val_loss: 2.5936 - val_accuracy: 0.3750 - 4s/epoch - 4s/step
Epoch 13/75
1/1 - 4s - loss: 2.3623 - accuracy: 0.3472 - val_loss: 2.4515 - val_accuracy: 0.4167 - 4s/epoch - 4s/step
Epoch 14/75
1/1 - 4s - loss: 2.2695 - accuracy: 0.3657 - val_loss: 2.2721 - val_accuracy: 0.4583 - 4s/epoch - 4s/step
Epoch 15/75
1/1 - 3s - loss: 2.1145 - accuracy: 0.3981 - val_loss: 2.0787 - val_accuracy: 0.5833 - 3s/epoch - 3s/step
Epoch 16/75
1/1 - 4s - loss: 1.9322 - accuracy: 0.4722 - val_loss: 1.8825 - val_accuracy: 0.5833 - 4s/epoch - 4s/step
Epoch 17/75
1/1 - 4s - loss: 1.6845 - accuracy: 0.5463 - val_loss: 1.6253 - val_accuracy: 0.7083 - 4s/epoch - 4s/step
Epoch 18/75
1/1 - 4s - loss: 1.4026 - accuracy: 0.6157 - val_loss: 1.3390 - val_accuracy: 0.6667 - 4s/epoch - 4s/step
Epoch 19/75
1/1 - 4s - loss: 1.2406 - accuracy: 0.6250 - val_loss: 1.1051 - val_accuracy: 0.8333 - 4s/epoch - 4s/step
Epoch 20/75
1/1 - 4s - loss: 1.1321 - accuracy: 0.6944 - val_loss: 0.7954 - val_accuracy: 0.9167 - 4s/epoch - 4s/step
Epoch 21/75
1/1 - 4s - loss: 0.8889 - accuracy: 0.7222 - val_loss: 0.6387 - val_accuracy: 0.9167 - 4s/epoch - 4s/step
Epoch 22/75
1/1 - 4s - loss: 0.7380 - accuracy: 0.7917 - val_loss: 0.5737 - val_accuracy: 0.8750 - 4s/epoch - 4s/step
```

```
Epoch 23/75
1/1 - 4s - loss: 0.5493 - accuracy: 0.8472 - val_loss: 0.4350 - val_accuracy: 0.9167 - 4s/epoch - 4s/step
Epoch 24/75
1/1 - 4s - loss: 0.3771 - accuracy: 0.9120 - val_loss: 0.3106 - val_accuracy: 0.8750 - 4s/epoch - 4s/step
Epoch 25/75
1/1 - 4s - loss: 0.3615 - accuracy: 0.9074 - val_loss: 0.2491 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 26/75
1/1 - 4s - loss: 0.3151 - accuracy: 0.8935 - val_loss: 0.3074 - val_accuracy: 0.8750 - 4s/epoch - 4s/step
Epoch 27/75
1/1 - 4s - loss: 0.2334 - accuracy: 0.9352 - val_loss: 0.2912 - val_accuracy: 0.8750 - 4s/epoch - 4s/step
Epoch 28/75
1/1 - 4s - loss: 0.1755 - accuracy: 0.9537 - val_loss: 0.1851 - val_accuracy: 0.9167 - 4s/epoch - 4s/step
Epoch 29/75
1/1 - 4s - loss: 0.1419 - accuracy: 0.9722 - val_loss: 0.1147 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 30/75
1/1 - 4s - loss: 0.1163 - accuracy: 0.9676 - val_loss: 0.0752 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 31/75
1/1 - 4s - loss: 0.1017 - accuracy: 0.9722 - val_loss: 0.0843 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 32/75
1/1 - 4s - loss: 0.0716 - accuracy: 0.9861 - val_loss: 0.1194 - val_accuracy: 0.9167 - 4s/epoch - 4s/step
Epoch 33/75
1/1 - 4s - loss: 0.1002 - accuracy: 0.9722 - val_loss: 0.0643 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 34/75
1/1 - 4s - loss: 0.0485 - accuracy: 0.9954 - val_loss: 0.0360 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 35/75
1/1 - 4s - loss: 0.0614 - accuracy: 0.9815 - val_loss: 0.0389 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 36/75
1/1 - 4s - loss: 0.0663 - accuracy: 0.9769 - val_loss: 0.0248 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 37/75
1/1 - 4s - loss: 0.0452 - accuracy: 0.9815 - val_loss: 0.0212 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 38/75
1/1 - 4s - loss: 0.0400 - accuracy: 0.9861 - val_loss: 0.0412 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 39/75
1/1 - 4s - loss: 0.0267 - accuracy: 0.9954 - val_loss: 0.0752 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 40/75
1/1 - 4s - loss: 0.0155 - accuracy: 1.0000 - val_loss: 0.0858 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 41/75
1/1 - 4s - loss: 0.0413 - accuracy: 0.9815 - val_loss: 0.0539 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 42/75
1/1 - 4s - loss: 0.0119 - accuracy: 1.0000 - val_loss: 0.0280 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 43/75
1/1 - 4s - loss: 0.0166 - accuracy: 0.9954 - val_loss: 0.0183 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 44/75
1/1 - 4s - loss: 0.0107 - accuracy: 1.0000 - val_loss: 0.0114 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
```

```
Epoch 45/75
1/1 - 3s - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.0076 - val_accuracy: 1.0000 - 3s/epoch - 3s/step
Epoch 46/75
1/1 - 4s - loss: 0.0101 - accuracy: 1.0000 - val_loss: 0.0054 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 47/75
1/1 - 4s - loss: 0.0152 - accuracy: 0.9907 - val_loss: 0.0049 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 48/75
1/1 - 4s - loss: 0.0083 - accuracy: 1.0000 - val_loss: 0.0054 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 49/75
1/1 - 4s - loss: 0.0187 - accuracy: 0.9954 - val_loss: 0.0085 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 50/75
1/1 - 4s - loss: 0.0184 - accuracy: 0.9954 - val_loss: 0.0174 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 51/75
1/1 - 4s - loss: 0.0045 - accuracy: 1.0000 - val_loss: 0.0313 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 52/75
1/1 - 4s - loss: 0.0058 - accuracy: 1.0000 - val_loss: 0.0525 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 53/75
1/1 - 4s - loss: 0.0064 - accuracy: 1.0000 - val_loss: 0.0786 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 54/75
1/1 - 4s - loss: 0.0060 - accuracy: 1.0000 - val_loss: 0.1027 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 55/75
1/1 - 4s - loss: 0.0182 - accuracy: 0.9907 - val_loss: 0.0953 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 56/75
1/1 - 4s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 0.0851 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 57/75
1/1 - 3s - loss: 0.0080 - accuracy: 1.0000 - val_loss: 0.0634 - val_accuracy: 0.9583 - 3s/epoch - 3s/step
Epoch 58/75
1/1 - 4s - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.0452 - val_accuracy: 0.9583 - 4s/epoch - 4s/step
Epoch 59/75
1/1 - 4s - loss: 0.0048 - accuracy: 1.0000 - val_loss: 0.0305 - val_accuracy: 1.0000 - 4s/epoch - 4s/step
Epoch 60/75
1/1 - 3s - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.0214 - val_accuracy: 1.0000 - 3s/epoch - 3s/step
Epoch 61/75
1/1 - 2s - loss: 0.0073 - accuracy: 0.9954 - val_loss: 0.0143 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 62/75
1/1 - 2s - loss: 0.0033 - accuracy: 1.0000 - val_loss: 0.0108 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 63/75
1/1 - 2s - loss: 0.0046 - accuracy: 1.0000 - val_loss: 0.0096 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 64/75
1/1 - 2s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 0.0090 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 65/75
1/1 - 2s - loss: 0.0191 - accuracy: 0.9954 - val_loss: 0.0085 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 66/75
1/1 - 2s - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.0093 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
```

```
Epoch 67/75
1/1 - 2s - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.0106 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 68/75
1/1 - 2s - loss: 8.8134e-04 - accuracy: 1.0000 - val_loss: 0.0125 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 69/75
1/1 - 2s - loss: 6.7348e-04 - accuracy: 1.0000 - val_loss: 0.0145 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 70/75
1/1 - 2s - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.0163 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 71/75
1/1 - 2s - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0170 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 72/75
1/1 - 2s - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0162 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 73/75
1/1 - 2s - loss: 6.1394e-04 - accuracy: 1.0000 - val_loss: 0.0154 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 74/75
1/1 - 2s - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.0140 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
Epoch 75/75
1/1 - 2s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0130 - val_accuracy: 1.0000 - 2s/epoch - 2s/step
```

In [20]:

```
result_score = model_final.evaluate(np.array(x_test),np.array(y_test),verbose=0)

print('Test Loss {:.4f}'.format(result_score[0]))
print('Test Accuracy {:.4f}'.format(result_score[1]))
```

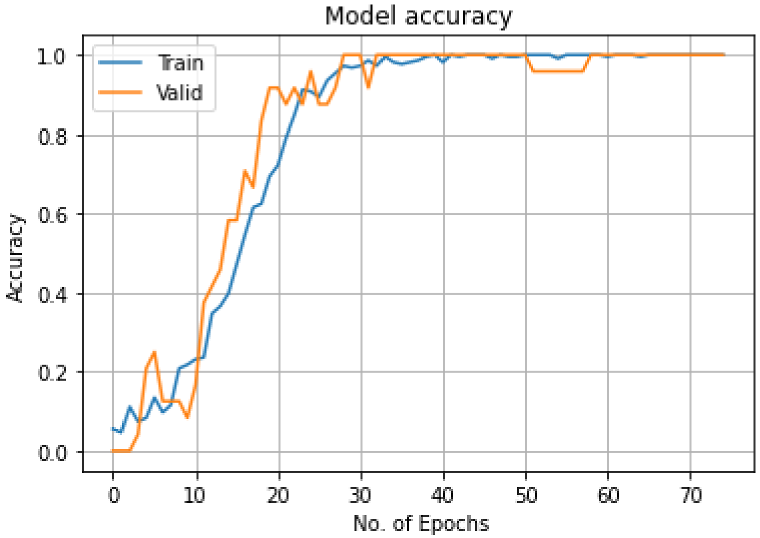
```
Test Loss 0.3551
Test Accuracy 0.9375
```

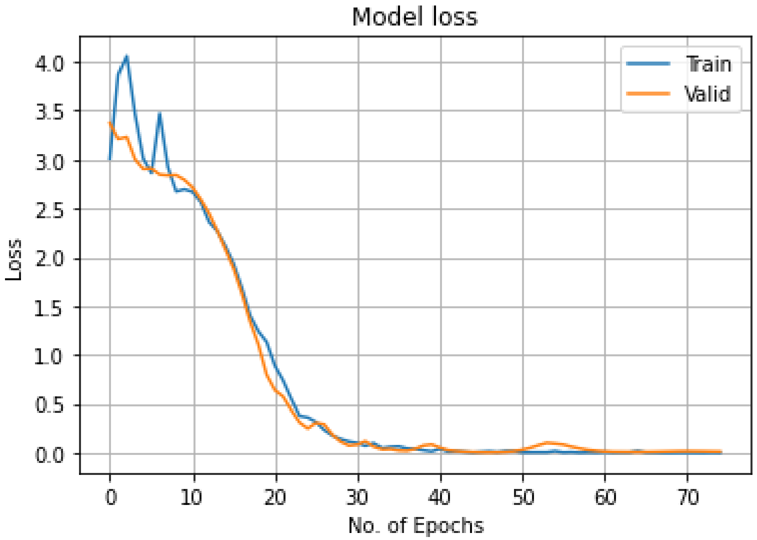
In [21]:

```
# Data in history

print(history_final.history.keys())

# Plotting Accuracy for final model
plt.plot(history_final.history['accuracy'])
plt.plot(history_final.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel(' No. of Epochs')
plt.legend(['Train', 'Valid'])
plt.grid()
plt.show()

# Plotting Loss for Final Model
```

```
plt.plot(history_final.history['loss'])
plt.plot(history_final.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('No. of Epochs')
plt.legend(['Train', 'Valid'])
plt.grid()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```





Conclusion

- Here in this project we analyzed ORL faces images (train and test sets were given). We used CNN method to build the model and train it.
- The analysis for different activation functions is fisrt observed to find that 'leaky-relu' activation function is one of the activation functions that can be used for out final model
- The model training is done using x_train and y_train with validation data as x_valid and y_valid. owever for evaluating model, we use x_test and y_test which gives us loss ~0.2435 with an accuracy of 93.75%

Submitted by: Sambit Mahanta

In [ ]:

```

```