Shraman_Gupta_Perform_Facial_Recognition_with_Deep_Learning_in_K

February 11, 2023

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

Submitted by: Ezhilarasi

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.

Step 1: Input the required libraries

[1]:
```python
# 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 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__)
```

```
     ␣
↪---------------------------------------------------------------------------

       ImportError                               Traceback (most recent call␣
↪last)

       <ipython-input-1-93889226a037> in <module>
        18 from keras.layers import Dense, Dropout, Flatten, Conv2D,␣
↪MaxPooling2D, Activation, LeakyReLU
        19 from keras.layers.noise import AlphaDropout
   ---> 20 from keras.optimizers import Adam
        21
        22 from keras.utils.generic_utils import get_custom_objects


       ImportError: cannot import name 'Adam' from 'keras.optimizers' (/usr/
↪local/lib/python3.7/site-packages/keras/optimizers.py)
```

[2]:
```
df = np.load('ORL_faces.npz') #loading dataset
```

Step 2: Load the dataset and preprocess the data

[3]:
```
# 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']
```

[4]:
```
# 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)
```

Step 3: 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)

```
[5]: x_train, x_valid, y_train, y_valid =␣
      ↪train_test_split(x_train,y_train,test_size=0.1,random_state=42)
```

Step 4: 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

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

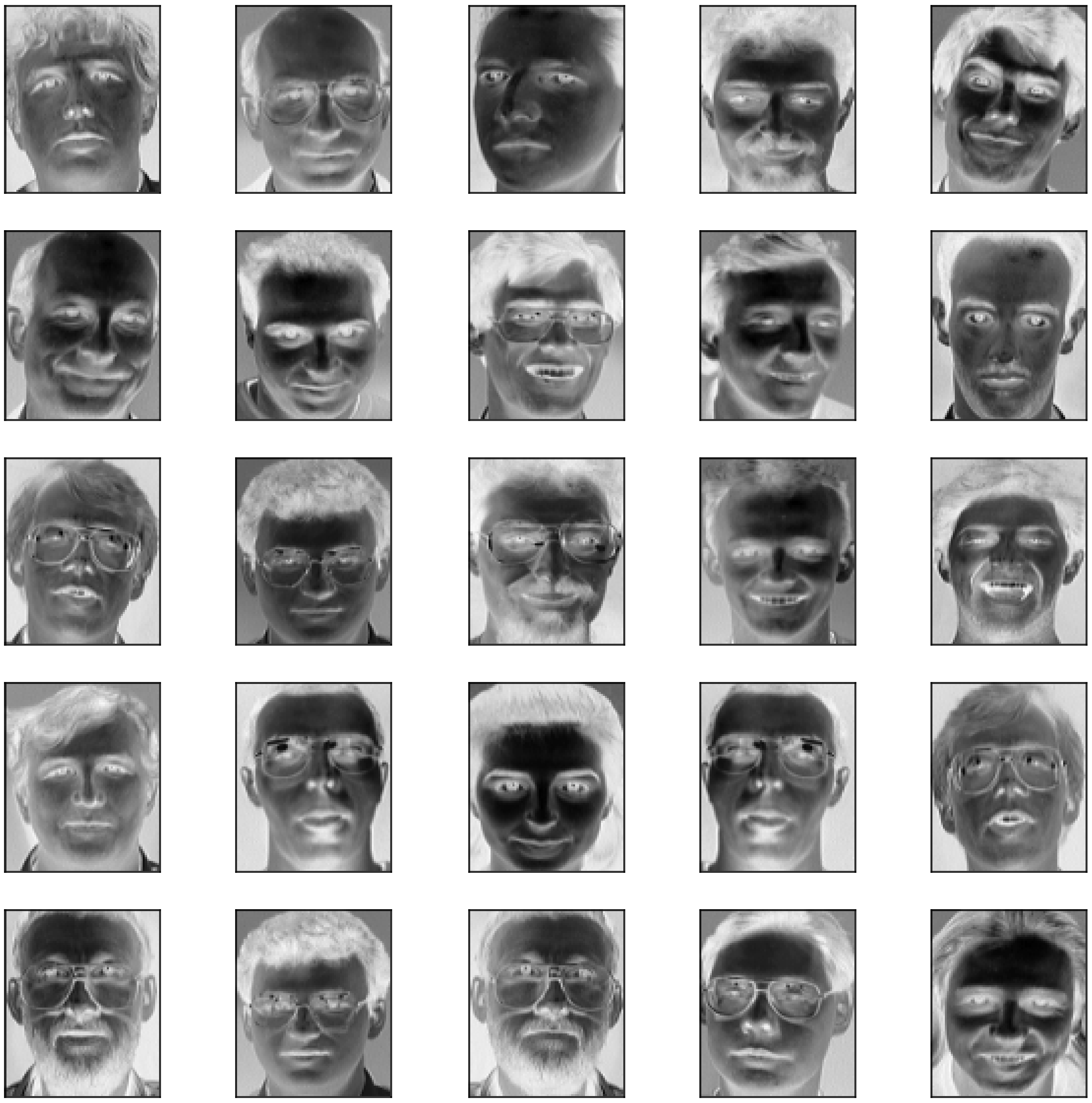
```
[7]: # Reshape function
     x_train = x_train.reshape(x_train.shape[0],*image_shape)
     x_test = x_test.reshape(x_test.shape[0],*image_shape)
     x_valid = x_valid.reshape(x_valid.shape[0],*image_shape)
```

```
[8]: print('Training dataset modified shape: ',x_train.shape)
     print('Testing dataset modified shape: ',x_test.shape)
     print('Validating dataset modified shape: ',x_valid.shape)
```

```
Training dataset modified shape:  (216, 112, 92, 1)
Testing dataset modified shape:  (160, 112, 92, 1)
Validating dataset modified shape:  (24, 112, 92, 1)
```

Visualize images in different colormap

```
[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()
```



[10]:
```
#visualize some images 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()
```

```
[11]: #visualize some images 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()
```

Step 5: 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()'

[12]:
```
# We will initialize our cnn model with activation function, dropout rate,␣
↪optimizer
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
```

[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

[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 - 0s - loss: 2.9842 - accuracy: 0.0833 - val_loss: 3.6024 - val_accuracy:
0.0417
Epoch 2/75
1/1 - 0s - loss: 4.4234 - accuracy: 0.0648 - val_loss: 3.0976 - val_accuracy:
0.0000e+00
Epoch 3/75
1/1 - 0s - loss: 4.0129 - accuracy: 0.0417 - val_loss: 2.9243 - val_accuracy:
0.0833
Epoch 4/75
1/1 - 0s - loss: 3.5444 - accuracy: 0.0694 - val_loss: 2.9880 - val_accuracy:
0.0417
Epoch 5/75
1/1 - 0s - loss: 3.1542 - accuracy: 0.0694 - val_loss: 3.0193 - val_accuracy:
0.0000e+00
Epoch 6/75
1/1 - 0s - loss: 3.0298 - accuracy: 0.0787 - val_loss: 3.0169 - val_accuracy:
0.0000e+00
Epoch 7/75
1/1 - 0s - loss: 2.9934 - accuracy: 0.0602 - val_loss: 3.0069 - val_accuracy:
0.0000e+00
Epoch 8/75
1/1 - 0s - loss: 2.9712 - accuracy: 0.0741 - val_loss: 3.0013 - val_accuracy:
0.0000e+00
Epoch 9/75
1/1 - 0s - loss: 2.9876 - accuracy: 0.0741 - val_loss: 2.9978 - val_accuracy:
0.0417
Epoch 10/75
1/1 - 0s - loss: 2.9628 - accuracy: 0.0880 - val_loss: 2.9983 - val_accuracy:
0.0417
Epoch 11/75
1/1 - 0s - loss: 2.9615 - accuracy: 0.0972 - val_loss: 2.9933 - val_accuracy:
0.0417
Epoch 12/75
```

```
1/1 - 0s - loss: 2.9692 - accuracy: 0.0648 - val_loss: 2.9851 - val_accuracy:
0.0000e+00
Epoch 13/75
1/1 - 0s - loss: 2.9559 - accuracy: 0.1065 - val_loss: 2.9780 - val_accuracy:
0.0000e+00
Epoch 14/75
1/1 - 0s - loss: 2.9472 - accuracy: 0.1389 - val_loss: 2.9675 - val_accuracy:
0.0417
Epoch 15/75
1/1 - 0s - loss: 2.9253 - accuracy: 0.1481 - val_loss: 2.9607 - val_accuracy:
0.0417
Epoch 16/75
1/1 - 0s - loss: 2.9126 - accuracy: 0.1343 - val_loss: 2.9494 - val_accuracy:
0.0417
Epoch 17/75
1/1 - 0s - loss: 2.8759 - accuracy: 0.1065 - val_loss: 2.9333 - val_accuracy:
0.0417
Epoch 18/75
1/1 - 0s - loss: 2.8374 - accuracy: 0.1574 - val_loss: 2.9071 - val_accuracy:
0.0417
Epoch 19/75
1/1 - 0s - loss: 2.7983 - accuracy: 0.1435 - val_loss: 2.8560 - val_accuracy:
0.0417
Epoch 20/75
1/1 - 0s - loss: 2.7451 - accuracy: 0.1528 - val_loss: 2.7972 - val_accuracy:
0.1250
Epoch 21/75
1/1 - 0s - loss: 2.6992 - accuracy: 0.1620 - val_loss: 2.7187 - val_accuracy:
0.1667
Epoch 22/75
1/1 - 0s - loss: 2.6243 - accuracy: 0.1991 - val_loss: 2.6155 - val_accuracy:
0.2500
Epoch 23/75
1/1 - 0s - loss: 2.5441 - accuracy: 0.2176 - val_loss: 2.4880 - val_accuracy:
0.5417
Epoch 24/75
1/1 - 0s - loss: 2.4062 - accuracy: 0.3380 - val_loss: 2.3234 - val_accuracy:
0.6667
Epoch 25/75
1/1 - 0s - loss: 2.2585 - accuracy: 0.3611 - val_loss: 2.1132 - val_accuracy:
0.7083
Epoch 26/75
1/1 - 0s - loss: 2.1200 - accuracy: 0.3981 - val_loss: 1.9291 - val_accuracy:
0.7500
Epoch 27/75
1/1 - 0s - loss: 2.0043 - accuracy: 0.4028 - val_loss: 1.7247 - val_accuracy:
0.7500
Epoch 28/75
```

```
1/1 - 0s - loss: 1.7984 - accuracy: 0.4583 - val_loss: 1.4198 - val_accuracy:
0.7500
Epoch 29/75
1/1 - 0s - loss: 1.6368 - accuracy: 0.5231 - val_loss: 1.2668 - val_accuracy:
0.8333
Epoch 30/75
1/1 - 0s - loss: 1.5005 - accuracy: 0.5741 - val_loss: 1.0890 - val_accuracy:
0.8333
Epoch 31/75
1/1 - 0s - loss: 1.2556 - accuracy: 0.6667 - val_loss: 0.9233 - val_accuracy:
0.8333
Epoch 32/75
1/1 - 0s - loss: 1.0980 - accuracy: 0.6991 - val_loss: 0.7732 - val_accuracy:
0.8750
Epoch 33/75
1/1 - 0s - loss: 0.9786 - accuracy: 0.6852 - val_loss: 0.6405 - val_accuracy:
0.8750
Epoch 34/75
1/1 - 0s - loss: 0.8506 - accuracy: 0.7315 - val_loss: 0.4938 - val_accuracy:
0.9167
Epoch 35/75
1/1 - 0s - loss: 0.6960 - accuracy: 0.8380 - val_loss: 0.3635 - val_accuracy:
0.9583
Epoch 36/75
1/1 - 0s - loss: 0.5012 - accuracy: 0.8611 - val_loss: 0.2979 - val_accuracy:
1.0000
Epoch 37/75
1/1 - 0s - loss: 0.4738 - accuracy: 0.8611 - val_loss: 0.3011 - val_accuracy:
0.8750
Epoch 38/75
1/1 - 0s - loss: 0.4599 - accuracy: 0.8472 - val_loss: 0.2759 - val_accuracy:
0.9167
Epoch 39/75
1/1 - 0s - loss: 0.3989 - accuracy: 0.8704 - val_loss: 0.2593 - val_accuracy:
0.9583
Epoch 40/75
1/1 - 0s - loss: 0.3010 - accuracy: 0.9352 - val_loss: 0.2195 - val_accuracy:
0.9583
Epoch 41/75
1/1 - 0s - loss: 0.2078 - accuracy: 0.9444 - val_loss: 0.1561 - val_accuracy:
0.9583
Epoch 42/75
1/1 - 0s - loss: 0.1476 - accuracy: 0.9537 - val_loss: 0.1051 - val_accuracy:
1.0000
Epoch 43/75
1/1 - 0s - loss: 0.1957 - accuracy: 0.9491 - val_loss: 0.0948 - val_accuracy:
1.0000
Epoch 44/75
```

```
1/1 - 0s - loss: 0.1362 - accuracy: 0.9630 - val_loss: 0.0751 - val_accuracy:
1.0000
Epoch 45/75
1/1 - 0s - loss: 0.1383 - accuracy: 0.9491 - val_loss: 0.0757 - val_accuracy:
0.9583
Epoch 46/75
1/1 - 0s - loss: 0.1324 - accuracy: 0.9630 - val_loss: 0.0587 - val_accuracy:
1.0000
Epoch 47/75
1/1 - 0s - loss: 0.0694 - accuracy: 0.9861 - val_loss: 0.0418 - val_accuracy:
1.0000
Epoch 48/75
1/1 - 0s - loss: 0.0720 - accuracy: 0.9815 - val_loss: 0.0362 - val_accuracy:
1.0000
Epoch 49/75
1/1 - 0s - loss: 0.0681 - accuracy: 0.9861 - val_loss: 0.0383 - val_accuracy:
1.0000
Epoch 50/75
1/1 - 0s - loss: 0.0397 - accuracy: 0.9954 - val_loss: 0.0383 - val_accuracy:
1.0000
Epoch 51/75
1/1 - 0s - loss: 0.0531 - accuracy: 0.9769 - val_loss: 0.0287 - val_accuracy:
1.0000
Epoch 52/75
1/1 - 0s - loss: 0.0246 - accuracy: 0.9954 - val_loss: 0.0157 - val_accuracy:
1.0000
Epoch 53/75
1/1 - 0s - loss: 0.0306 - accuracy: 0.9861 - val_loss: 0.0090 - val_accuracy:
1.0000
Epoch 54/75
1/1 - 0s - loss: 0.0376 - accuracy: 0.9815 - val_loss: 0.0065 - val_accuracy:
1.0000
Epoch 55/75
1/1 - 0s - loss: 0.0316 - accuracy: 0.9815 - val_loss: 0.0052 - val_accuracy:
1.0000
Epoch 56/75
1/1 - 0s - loss: 0.0202 - accuracy: 0.9907 - val_loss: 0.0053 - val_accuracy:
1.0000
Epoch 57/75
1/1 - 0s - loss: 0.0261 - accuracy: 0.9954 - val_loss: 0.0047 - val_accuracy:
1.0000
Epoch 58/75
1/1 - 0s - loss: 0.0309 - accuracy: 0.9954 - val_loss: 0.0040 - val_accuracy:
1.0000
Epoch 59/75
1/1 - 0s - loss: 0.0161 - accuracy: 0.9907 - val_loss: 0.0040 - val_accuracy:
1.0000
Epoch 60/75
```

```
1/1 - 0s - loss: 0.0297 - accuracy: 0.9861 - val_loss: 0.0045 - val_accuracy:
1.0000
Epoch 61/75
1/1 - 0s - loss: 0.0159 - accuracy: 0.9954 - val_loss: 0.0061 - val_accuracy:
1.0000
Epoch 62/75
1/1 - 0s - loss: 0.0205 - accuracy: 0.9954 - val_loss: 0.0071 - val_accuracy:
1.0000
Epoch 63/75
1/1 - 0s - loss: 0.0113 - accuracy: 1.0000 - val_loss: 0.0074 - val_accuracy:
1.0000
Epoch 64/75
1/1 - 0s - loss: 0.0192 - accuracy: 0.9954 - val_loss: 0.0056 - val_accuracy:
1.0000
Epoch 65/75
1/1 - 0s - loss: 0.0237 - accuracy: 0.9907 - val_loss: 0.0045 - val_accuracy:
1.0000
Epoch 66/75
1/1 - 0s - loss: 0.0315 - accuracy: 0.9954 - val_loss: 0.0066 - val_accuracy:
1.0000
Epoch 67/75
1/1 - 0s - loss: 0.0099 - accuracy: 1.0000 - val_loss: 0.0101 - val_accuracy:
1.0000
Epoch 68/75
1/1 - 0s - loss: 0.0063 - accuracy: 1.0000 - val_loss: 0.0145 - val_accuracy:
1.0000
Epoch 69/75
1/1 - 0s - loss: 0.0177 - accuracy: 0.9907 - val_loss: 0.0164 - val_accuracy:
1.0000
Epoch 70/75
1/1 - 0s - loss: 0.0318 - accuracy: 0.9907 - val_loss: 0.0150 - val_accuracy:
1.0000
Epoch 71/75
1/1 - 0s - loss: 0.0061 - accuracy: 1.0000 - val_loss: 0.0126 - val_accuracy:
1.0000
Epoch 72/75
1/1 - 0s - loss: 0.0052 - accuracy: 1.0000 - val_loss: 0.0099 - val_accuracy:
1.0000
Epoch 73/75
1/1 - 0s - loss: 0.0032 - accuracy: 1.0000 - val_loss: 0.0075 - val_accuracy:
1.0000
Epoch 74/75
1/1 - 0s - loss: 0.0213 - accuracy: 0.9954 - val_loss: 0.0057 - val_accuracy:
1.0000
Epoch 75/75
1/1 - 0s - loss: 0.0087 - accuracy: 1.0000 - val_loss: 0.0040 - val_accuracy:
1.0000
```

```
Training with elu activation function

Epoch 1/75
1/1 - 0s - loss: 3.1712 - accuracy: 0.0556 - val_loss: 8.3652 - val_accuracy:
0.0833
Epoch 2/75
1/1 - 0s - loss: 10.1480 - accuracy: 0.0602 - val_loss: 6.8461 - val_accuracy:
0.0833
Epoch 3/75
1/1 - 0s - loss: 10.5502 - accuracy: 0.0556 - val_loss: 4.7639 - val_accuracy:
0.0833
Epoch 4/75
1/1 - 0s - loss: 6.4788 - accuracy: 0.0787 - val_loss: 4.6739 - val_accuracy:
0.1250
Epoch 5/75
1/1 - 0s - loss: 4.7195 - accuracy: 0.1019 - val_loss: 17.0984 - val_accuracy:
0.0833
Epoch 6/75
1/1 - 0s - loss: 16.8569 - accuracy: 0.0463 - val_loss: 6.3732 - val_accuracy:
0.0417
Epoch 7/75
1/1 - 0s - loss: 8.1324 - accuracy: 0.1111 - val_loss: 5.7197 - val_accuracy:
0.0417
Epoch 8/75
1/1 - 0s - loss: 6.4880 - accuracy: 0.0694 - val_loss: 4.0184 - val_accuracy:
0.1667
Epoch 9/75
1/1 - 0s - loss: 5.6822 - accuracy: 0.0648 - val_loss: 3.2449 - val_accuracy:
0.0833
Epoch 10/75
1/1 - 0s - loss: 5.3538 - accuracy: 0.0648 - val_loss: 3.9112 - val_accuracy:
0.0833
Epoch 11/75
1/1 - 0s - loss: 5.5314 - accuracy: 0.0463 - val_loss: 3.2105 - val_accuracy:
0.0417
Epoch 12/75
1/1 - 0s - loss: 4.4224 - accuracy: 0.1065 - val_loss: 2.9710 - val_accuracy:
0.1667
Epoch 13/75
1/1 - 0s - loss: 3.6304 - accuracy: 0.1574 - val_loss: 2.2838 - val_accuracy:
0.2083
Epoch 14/75
1/1 - 0s - loss: 3.0146 - accuracy: 0.1852 - val_loss: 3.2553 - val_accuracy:
0.3333
Epoch 15/75
1/1 - 0s - loss: 3.1544 - accuracy: 0.2222 - val_loss: 6.0991 - val_accuracy:
0.0833
Epoch 16/75
```

```
1/1 - 0s - loss: 7.2886 - accuracy: 0.1157 - val_loss: 3.3946 - val_accuracy:
0.5000
Epoch 17/75
1/1 - 0s - loss: 4.0160 - accuracy: 0.2500 - val_loss: 3.7178 - val_accuracy:
0.1667
Epoch 18/75
1/1 - 0s - loss: 3.6952 - accuracy: 0.3056 - val_loss: 2.6655 - val_accuracy:
0.3750
Epoch 19/75
1/1 - 0s - loss: 2.8920 - accuracy: 0.3287 - val_loss: 2.0373 - val_accuracy:
0.4583
Epoch 20/75
1/1 - 0s - loss: 2.2367 - accuracy: 0.4583 - val_loss: 2.3172 - val_accuracy:
0.5000
Epoch 21/75
1/1 - 0s - loss: 2.5427 - accuracy: 0.4028 - val_loss: 1.5234 - val_accuracy:
0.5417
Epoch 22/75
1/1 - 0s - loss: 1.8051 - accuracy: 0.4907 - val_loss: 1.4727 - val_accuracy:
0.6250
Epoch 23/75
1/1 - 0s - loss: 1.7220 - accuracy: 0.4676 - val_loss: 1.6486 - val_accuracy:
0.5000
Epoch 24/75
1/1 - 0s - loss: 1.6762 - accuracy: 0.5185 - val_loss: 1.1323 - val_accuracy:
0.6667
Epoch 25/75
1/1 - 0s - loss: 1.0376 - accuracy: 0.6852 - val_loss: 1.2738 - val_accuracy:
0.4583
Epoch 26/75
1/1 - 0s - loss: 1.4034 - accuracy: 0.5741 - val_loss: 0.8520 - val_accuracy:
0.6667
Epoch 27/75
1/1 - 0s - loss: 0.7146 - accuracy: 0.7778 - val_loss: 0.8906 - val_accuracy:
0.7500
Epoch 28/75
1/1 - 0s - loss: 0.7431 - accuracy: 0.7639 - val_loss: 0.7687 - val_accuracy:
0.7917
Epoch 29/75
1/1 - 0s - loss: 0.5924 - accuracy: 0.8287 - val_loss: 0.8389 - val_accuracy:
0.7500
Epoch 30/75
1/1 - 0s - loss: 0.7294 - accuracy: 0.7639 - val_loss: 0.5253 - val_accuracy:
0.8333
Epoch 31/75
1/1 - 0s - loss: 0.5220 - accuracy: 0.8333 - val_loss: 0.5247 - val_accuracy:
0.8333
Epoch 32/75
```

```
1/1 - 0s - loss: 0.4547 - accuracy: 0.8611 - val_loss: 0.3426 - val_accuracy:
0.9583
Epoch 33/75
1/1 - 0s - loss: 0.4382 - accuracy: 0.8519 - val_loss: 0.3315 - val_accuracy:
0.9583
Epoch 34/75
1/1 - 0s - loss: 0.2932 - accuracy: 0.9120 - val_loss: 0.5468 - val_accuracy:
0.7083
Epoch 35/75
1/1 - 0s - loss: 0.3014 - accuracy: 0.9028 - val_loss: 0.2670 - val_accuracy:
0.8750
Epoch 36/75
1/1 - 0s - loss: 0.2055 - accuracy: 0.9398 - val_loss: 0.2646 - val_accuracy:
0.8750
Epoch 37/75
1/1 - 0s - loss: 0.2502 - accuracy: 0.9259 - val_loss: 0.2238 - val_accuracy:
0.9583
Epoch 38/75
1/1 - 0s - loss: 0.1257 - accuracy: 0.9537 - val_loss: 0.2313 - val_accuracy:
0.9583
Epoch 39/75
1/1 - 0s - loss: 0.1271 - accuracy: 0.9630 - val_loss: 0.1629 - val_accuracy:
0.9583
Epoch 40/75
1/1 - 0s - loss: 0.0759 - accuracy: 0.9815 - val_loss: 0.1239 - val_accuracy:
0.9583
Epoch 41/75
1/1 - 0s - loss: 0.0933 - accuracy: 0.9676 - val_loss: 0.0999 - val_accuracy:
0.9583
Epoch 42/75
1/1 - 0s - loss: 0.0691 - accuracy: 0.9815 - val_loss: 0.0766 - val_accuracy:
1.0000
Epoch 43/75
1/1 - 0s - loss: 0.0909 - accuracy: 0.9769 - val_loss: 0.0707 - val_accuracy:
1.0000
Epoch 44/75
1/1 - 0s - loss: 0.0728 - accuracy: 0.9815 - val_loss: 0.0750 - val_accuracy:
0.9583
Epoch 45/75
1/1 - 0s - loss: 0.0630 - accuracy: 0.9769 - val_loss: 0.1192 - val_accuracy:
0.9583
Epoch 46/75
1/1 - 0s - loss: 0.0651 - accuracy: 0.9769 - val_loss: 0.1202 - val_accuracy:
0.9583
Epoch 47/75
1/1 - 0s - loss: 0.0849 - accuracy: 0.9861 - val_loss: 0.0706 - val_accuracy:
0.9583
Epoch 48/75
```

```
1/1 - 0s - loss: 0.0396 - accuracy: 0.9954 - val_loss: 0.0891 - val_accuracy:
0.9583
Epoch 49/75
1/1 - 0s - loss: 0.0345 - accuracy: 0.9954 - val_loss: 0.0734 - val_accuracy:
0.9583
Epoch 50/75
1/1 - 0s - loss: 0.0484 - accuracy: 0.9907 - val_loss: 0.0880 - val_accuracy:
0.9583
Epoch 51/75
1/1 - 0s - loss: 0.0298 - accuracy: 0.9954 - val_loss: 0.0827 - val_accuracy:
0.9583
Epoch 52/75
1/1 - 0s - loss: 0.0174 - accuracy: 1.0000 - val_loss: 0.0703 - val_accuracy:
0.9583
Epoch 53/75
1/1 - 0s - loss: 0.0397 - accuracy: 0.9815 - val_loss: 0.0515 - val_accuracy:
0.9583
Epoch 54/75
1/1 - 0s - loss: 0.0292 - accuracy: 0.9907 - val_loss: 0.0510 - val_accuracy:
0.9583
Epoch 55/75
1/1 - 0s - loss: 0.0270 - accuracy: 0.9954 - val_loss: 0.0714 - val_accuracy:
0.9583
Epoch 56/75
1/1 - 0s - loss: 0.0483 - accuracy: 0.9907 - val_loss: 0.0470 - val_accuracy:
0.9583
Epoch 57/75
1/1 - 0s - loss: 0.0197 - accuracy: 1.0000 - val_loss: 0.0247 - val_accuracy:
1.0000
Epoch 58/75
1/1 - 0s - loss: 0.0190 - accuracy: 0.9907 - val_loss: 0.0297 - val_accuracy:
1.0000
Epoch 59/75
1/1 - 0s - loss: 0.0272 - accuracy: 0.9907 - val_loss: 0.0199 - val_accuracy:
1.0000
Epoch 60/75
1/1 - 0s - loss: 0.0339 - accuracy: 0.9954 - val_loss: 0.0403 - val_accuracy:
1.0000
Epoch 61/75
1/1 - 0s - loss: 0.0374 - accuracy: 0.9954 - val_loss: 0.0344 - val_accuracy:
1.0000
Epoch 62/75
1/1 - 0s - loss: 0.0160 - accuracy: 1.0000 - val_loss: 0.0228 - val_accuracy:
1.0000
Epoch 63/75
1/1 - 0s - loss: 0.0078 - accuracy: 1.0000 - val_loss: 0.0221 - val_accuracy:
1.0000
Epoch 64/75
```

```
1/1 - 0s - loss: 0.0065 - accuracy: 1.0000 - val_loss: 0.0306 - val_accuracy:
1.0000
Epoch 65/75
1/1 - 0s - loss: 0.0126 - accuracy: 0.9954 - val_loss: 0.0276 - val_accuracy:
1.0000
Epoch 66/75
1/1 - 0s - loss: 0.0145 - accuracy: 0.9954 - val_loss: 0.0250 - val_accuracy:
1.0000
Epoch 67/75
1/1 - 0s - loss: 0.0254 - accuracy: 0.9907 - val_loss: 0.0189 - val_accuracy:
1.0000
Epoch 68/75
1/1 - 0s - loss: 0.0196 - accuracy: 0.9907 - val_loss: 0.0084 - val_accuracy:
1.0000
Epoch 69/75
1/1 - 0s - loss: 0.0174 - accuracy: 0.9954 - val_loss: 0.0058 - val_accuracy:
1.0000
Epoch 70/75
1/1 - 0s - loss: 0.0065 - accuracy: 1.0000 - val_loss: 0.0051 - val_accuracy:
1.0000
Epoch 71/75
1/1 - 0s - loss: 0.0132 - accuracy: 0.9954 - val_loss: 0.0045 - val_accuracy:
1.0000
Epoch 72/75
1/1 - 0s - loss: 0.0233 - accuracy: 0.9954 - val_loss: 0.0037 - val_accuracy:
1.0000
Epoch 73/75
1/1 - 0s - loss: 0.0078 - accuracy: 0.9954 - val_loss: 0.0040 - val_accuracy:
1.0000
Epoch 74/75
1/1 - 0s - loss: 0.0050 - accuracy: 1.0000 - val_loss: 0.0080 - val_accuracy:
1.0000
Epoch 75/75
1/1 - 0s - loss: 0.0049 - accuracy: 1.0000 - val_loss: 0.0166 - val_accuracy:
1.0000

Training with leaky-relu activation function

Epoch 1/75
1/1 - 0s - loss: 2.9985 - accuracy: 0.0417 - val_loss: 3.5216 - val_accuracy:
0.0417
Epoch 2/75
1/1 - 0s - loss: 4.0359 - accuracy: 0.0741 - val_loss: 3.0730 - val_accuracy:
0.1250
Epoch 3/75
1/1 - 0s - loss: 4.2957 - accuracy: 0.0463 - val_loss: 2.9148 - val_accuracy:
0.0833
Epoch 4/75
```

```
1/1 - 0s - loss: 3.3465 - accuracy: 0.0278 - val_loss: 2.9341 - val_accuracy:
0.0417
Epoch 5/75
1/1 - 0s - loss: 2.9124 - accuracy: 0.1250 - val_loss: 2.9429 - val_accuracy:
0.0417
Epoch 6/75
1/1 - 0s - loss: 2.8420 - accuracy: 0.1944 - val_loss: 2.9731 - val_accuracy:
0.0417
Epoch 7/75
1/1 - 0s - loss: 3.3200 - accuracy: 0.1111 - val_loss: 2.8384 - val_accuracy:
0.0833
Epoch 8/75
1/1 - 0s - loss: 3.0208 - accuracy: 0.1065 - val_loss: 2.8517 - val_accuracy:
0.0417
Epoch 9/75
1/1 - 0s - loss: 2.6728 - accuracy: 0.2407 - val_loss: 2.8398 - val_accuracy:
0.1250
Epoch 10/75
1/1 - 0s - loss: 2.5941 - accuracy: 0.3102 - val_loss: 2.7904 - val_accuracy:
0.1667
Epoch 11/75
1/1 - 0s - loss: 2.5222 - accuracy: 0.3472 - val_loss: 2.6859 - val_accuracy:
0.1667
Epoch 12/75
1/1 - 0s - loss: 2.4288 - accuracy: 0.2870 - val_loss: 2.5153 - val_accuracy:
0.3333
Epoch 13/75
1/1 - 0s - loss: 2.2922 - accuracy: 0.3241 - val_loss: 2.3110 - val_accuracy:
0.5417
Epoch 14/75
1/1 - 0s - loss: 2.0461 - accuracy: 0.4769 - val_loss: 2.0845 - val_accuracy:
0.6250
Epoch 15/75
1/1 - 0s - loss: 1.7828 - accuracy: 0.5648 - val_loss: 1.8780 - val_accuracy:
0.5417
Epoch 16/75
1/1 - 0s - loss: 1.6163 - accuracy: 0.5324 - val_loss: 1.5926 - val_accuracy:
0.6250
Epoch 17/75
1/1 - 0s - loss: 1.4183 - accuracy: 0.6111 - val_loss: 1.2770 - val_accuracy:
0.7917
Epoch 18/75
1/1 - 0s - loss: 1.1241 - accuracy: 0.7130 - val_loss: 0.9907 - val_accuracy:
0.7917
Epoch 19/75
1/1 - 0s - loss: 0.9639 - accuracy: 0.7778 - val_loss: 0.7830 - val_accuracy:
0.8333
Epoch 20/75
```

```
1/1 - 0s - loss: 0.7903 - accuracy: 0.7731 - val_loss: 0.6044 - val_accuracy:
0.9583
Epoch 21/75
1/1 - 0s - loss: 0.6037 - accuracy: 0.8333 - val_loss: 0.5001 - val_accuracy:
0.8333
Epoch 22/75
1/1 - 0s - loss: 0.4462 - accuracy: 0.9028 - val_loss: 0.3836 - val_accuracy:
0.8750
Epoch 23/75
1/1 - 0s - loss: 0.3835 - accuracy: 0.8981 - val_loss: 0.3256 - val_accuracy:
0.8750
Epoch 24/75
1/1 - 0s - loss: 0.2851 - accuracy: 0.9306 - val_loss: 0.2975 - val_accuracy:
0.9167
Epoch 25/75
1/1 - 0s - loss: 0.2199 - accuracy: 0.9444 - val_loss: 0.2772 - val_accuracy:
0.9583
Epoch 26/75
1/1 - 0s - loss: 0.1701 - accuracy: 0.9676 - val_loss: 0.2160 - val_accuracy:
0.9583
Epoch 27/75
1/1 - 0s - loss: 0.1296 - accuracy: 0.9583 - val_loss: 0.1168 - val_accuracy:
0.9583
Epoch 28/75
1/1 - 0s - loss: 0.1329 - accuracy: 0.9583 - val_loss: 0.1004 - val_accuracy:
0.9583
Epoch 29/75
1/1 - 0s - loss: 0.0707 - accuracy: 0.9861 - val_loss: 0.1209 - val_accuracy:
1.0000
Epoch 30/75
1/1 - 0s - loss: 0.0894 - accuracy: 0.9676 - val_loss: 0.0845 - val_accuracy:
1.0000
Epoch 31/75
1/1 - 0s - loss: 0.0851 - accuracy: 0.9722 - val_loss: 0.0624 - val_accuracy:
0.9583
Epoch 32/75
1/1 - 0s - loss: 0.0891 - accuracy: 0.9722 - val_loss: 0.0736 - val_accuracy:
1.0000
Epoch 33/75
1/1 - 0s - loss: 0.0499 - accuracy: 0.9861 - val_loss: 0.1470 - val_accuracy:
0.9583
Epoch 34/75
1/1 - 0s - loss: 0.0555 - accuracy: 0.9861 - val_loss: 0.1779 - val_accuracy:
0.8750
Epoch 35/75
1/1 - 0s - loss: 0.1020 - accuracy: 0.9676 - val_loss: 0.0326 - val_accuracy:
1.0000
Epoch 36/75
```

```
1/1 - 0s - loss: 0.0331 - accuracy: 0.9815 - val_loss: 0.0203 - val_accuracy:
1.0000
Epoch 37/75
1/1 - 0s - loss: 0.0219 - accuracy: 0.9954 - val_loss: 0.0355 - val_accuracy:
1.0000
Epoch 38/75
1/1 - 0s - loss: 0.0584 - accuracy: 0.9815 - val_loss: 0.0671 - val_accuracy:
1.0000
Epoch 39/75
1/1 - 0s - loss: 0.0464 - accuracy: 0.9907 - val_loss: 0.0583 - val_accuracy:
1.0000
Epoch 40/75
1/1 - 0s - loss: 0.0161 - accuracy: 1.0000 - val_loss: 0.0497 - val_accuracy:
1.0000
Epoch 41/75
1/1 - 0s - loss: 0.0448 - accuracy: 0.9954 - val_loss: 0.0408 - val_accuracy:
1.0000
Epoch 42/75
1/1 - 0s - loss: 0.0135 - accuracy: 1.0000 - val_loss: 0.0384 - val_accuracy:
1.0000
Epoch 43/75
1/1 - 0s - loss: 0.0345 - accuracy: 0.9861 - val_loss: 0.0253 - val_accuracy:
1.0000
Epoch 44/75
1/1 - 0s - loss: 0.0200 - accuracy: 0.9954 - val_loss: 0.0186 - val_accuracy:
1.0000
Epoch 45/75
1/1 - 0s - loss: 0.0204 - accuracy: 0.9954 - val_loss: 0.0151 - val_accuracy:
1.0000
Epoch 46/75
1/1 - 0s - loss: 0.0118 - accuracy: 1.0000 - val_loss: 0.0162 - val_accuracy:
1.0000
Epoch 47/75
1/1 - 0s - loss: 0.0061 - accuracy: 1.0000 - val_loss: 0.0161 - val_accuracy:
1.0000
Epoch 48/75
1/1 - 0s - loss: 0.0086 - accuracy: 1.0000 - val_loss: 0.0125 - val_accuracy:
1.0000
Epoch 49/75
1/1 - 0s - loss: 0.0093 - accuracy: 0.9954 - val_loss: 0.0078 - val_accuracy:
1.0000
Epoch 50/75
1/1 - 0s - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.0070 - val_accuracy:
1.0000
Epoch 51/75
1/1 - 0s - loss: 0.0075 - accuracy: 1.0000 - val_loss: 0.0068 - val_accuracy:
1.0000
Epoch 52/75
```

```
1/1 - 0s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 0.0066 - val_accuracy:
1.0000
Epoch 53/75
1/1 - 0s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 0.0060 - val_accuracy:
1.0000
Epoch 54/75
1/1 - 0s - loss: 0.0062 - accuracy: 1.0000 - val_loss: 0.0042 - val_accuracy:
1.0000
Epoch 55/75
1/1 - 0s - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.0032 - val_accuracy:
1.0000
Epoch 56/75
1/1 - 0s - loss: 0.0234 - accuracy: 0.9954 - val_loss: 0.0035 - val_accuracy:
1.0000
Epoch 57/75
1/1 - 0s - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.0058 - val_accuracy:
1.0000
Epoch 58/75
1/1 - 0s - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0115 - val_accuracy:
1.0000
Epoch 59/75
1/1 - 0s - loss: 0.0072 - accuracy: 1.0000 - val_loss: 0.0165 - val_accuracy:
1.0000
Epoch 60/75
1/1 - 0s - loss: 0.0115 - accuracy: 0.9954 - val_loss: 0.0110 - val_accuracy:
1.0000
Epoch 61/75
1/1 - 0s - loss: 0.0052 - accuracy: 1.0000 - val_loss: 0.0086 - val_accuracy:
1.0000
Epoch 62/75
1/1 - 0s - loss: 0.0051 - accuracy: 1.0000 - val_loss: 0.0098 - val_accuracy:
1.0000
Epoch 63/75
1/1 - 0s - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.0133 - val_accuracy:
1.0000
Epoch 64/75
1/1 - 0s - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0188 - val_accuracy:
1.0000
Epoch 65/75
1/1 - 0s - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.0224 - val_accuracy:
1.0000
Epoch 66/75
1/1 - 0s - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.0210 - val_accuracy:
1.0000
Epoch 67/75
1/1 - 0s - loss: 0.0275 - accuracy: 0.9907 - val_loss: 0.0132 - val_accuracy:
1.0000
Epoch 68/75
```

```
1/1 - 0s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0087 - val_accuracy:
1.0000
Epoch 69/75
1/1 - 0s - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.0066 - val_accuracy:
1.0000
Epoch 70/75
1/1 - 0s - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.0054 - val_accuracy:
1.0000
Epoch 71/75
1/1 - 0s - loss: 9.2972e-04 - accuracy: 1.0000 - val_loss: 0.0049 -
val_accuracy: 1.0000
Epoch 72/75
1/1 - 0s - loss: 0.0025 - accuracy: 1.0000 - val_loss: 0.0047 - val_accuracy:
1.0000
Epoch 73/75
1/1 - 0s - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0044 - val_accuracy:
1.0000
Epoch 74/75
1/1 - 0s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0041 - val_accuracy:
1.0000
Epoch 75/75
1/1 - 0s - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.0038 - val_accuracy:
1.0000

Training with selu activation function

Epoch 1/75
1/1 - 0s - loss: 3.7572 - accuracy: 0.0463 - val_loss: 10.5641 - val_accuracy:
0.0417
Epoch 2/75
1/1 - 0s - loss: 5.2147 - accuracy: 0.0509 - val_loss: 10.2721 - val_accuracy:
0.0000e+00
Epoch 3/75
1/1 - 0s - loss: 4.9484 - accuracy: 0.0694 - val_loss: 9.1581 - val_accuracy:
0.0000e+00
Epoch 4/75
1/1 - 0s - loss: 4.1194 - accuracy: 0.0694 - val_loss: 4.3999 - val_accuracy:
0.0417
Epoch 5/75
1/1 - 0s - loss: 3.4382 - accuracy: 0.0370 - val_loss: 14.6527 - val_accuracy:
0.0833
Epoch 6/75
1/1 - 0s - loss: 6.7598 - accuracy: 0.0370 - val_loss: 7.1454 - val_accuracy:
0.0417
Epoch 7/75
1/1 - 0s - loss: 3.8075 - accuracy: 0.0787 - val_loss: 6.8843 - val_accuracy:
0.0833
Epoch 8/75
```

```
1/1 - 0s - loss: 3.7349 - accuracy: 0.0833 - val_loss: 4.6838 - val_accuracy:
0.0833
Epoch 9/75
1/1 - 0s - loss: 3.9162 - accuracy: 0.0741 - val_loss: 4.3444 - val_accuracy:
0.0833
Epoch 10/75
1/1 - 0s - loss: 3.9868 - accuracy: 0.0741 - val_loss: 5.9716 - val_accuracy:
0.1250
Epoch 11/75
1/1 - 0s - loss: 3.7739 - accuracy: 0.0833 - val_loss: 7.0227 - val_accuracy:
0.0833
Epoch 12/75
1/1 - 0s - loss: 3.6716 - accuracy: 0.1065 - val_loss: 5.8833 - val_accuracy:
0.0833
Epoch 13/75
1/1 - 0s - loss: 3.5272 - accuracy: 0.1019 - val_loss: 4.0280 - val_accuracy:
0.1667
Epoch 14/75
1/1 - 0s - loss: 3.2908 - accuracy: 0.1065 - val_loss: 3.1382 - val_accuracy:
0.2083
Epoch 15/75
1/1 - 0s - loss: 3.0366 - accuracy: 0.1157 - val_loss: 2.5522 - val_accuracy:
0.1250
Epoch 16/75
1/1 - 0s - loss: 3.0087 - accuracy: 0.1481 - val_loss: 2.2102 - val_accuracy:
0.2917
Epoch 17/75
1/1 - 0s - loss: 2.7174 - accuracy: 0.2130 - val_loss: 2.1589 - val_accuracy:
0.3750
Epoch 18/75
1/1 - 0s - loss: 2.7526 - accuracy: 0.1898 - val_loss: 2.0620 - val_accuracy:
0.4167
Epoch 19/75
1/1 - 0s - loss: 2.3501 - accuracy: 0.2824 - val_loss: 1.8052 - val_accuracy:
0.4583
Epoch 20/75
1/1 - 0s - loss: 2.3068 - accuracy: 0.2778 - val_loss: 1.9531 - val_accuracy:
0.3750
Epoch 21/75
1/1 - 0s - loss: 2.1669 - accuracy: 0.3102 - val_loss: 1.8360 - val_accuracy:
0.4167
Epoch 22/75
1/1 - 0s - loss: 1.8610 - accuracy: 0.4074 - val_loss: 1.8690 - val_accuracy:
0.6250
Epoch 23/75
1/1 - 0s - loss: 1.8425 - accuracy: 0.4583 - val_loss: 1.1186 - val_accuracy:
0.6667
Epoch 24/75
```

```
1/1 - 0s - loss: 1.6054 - accuracy: 0.4861 - val_loss: 1.1874 - val_accuracy:
0.6667
Epoch 25/75
1/1 - 0s - loss: 1.5279 - accuracy: 0.5370 - val_loss: 1.4535 - val_accuracy:
0.8333
Epoch 26/75
1/1 - 0s - loss: 1.4744 - accuracy: 0.5046 - val_loss: 1.0956 - val_accuracy:
0.8333
Epoch 27/75
1/1 - 0s - loss: 1.2711 - accuracy: 0.5880 - val_loss: 0.7582 - val_accuracy:
0.7917
Epoch 28/75
1/1 - 0s - loss: 1.1411 - accuracy: 0.6528 - val_loss: 0.8823 - val_accuracy:
0.7500
Epoch 29/75
1/1 - 0s - loss: 1.0942 - accuracy: 0.6250 - val_loss: 0.6580 - val_accuracy:
0.8750
Epoch 30/75
1/1 - 0s - loss: 0.9138 - accuracy: 0.6898 - val_loss: 0.6095 - val_accuracy:
0.9167
Epoch 31/75
1/1 - 0s - loss: 0.7574 - accuracy: 0.7870 - val_loss: 0.8524 - val_accuracy:
0.9167
Epoch 32/75
1/1 - 0s - loss: 0.6875 - accuracy: 0.7731 - val_loss: 1.0448 - val_accuracy:
0.9167
Epoch 33/75
1/1 - 0s - loss: 0.5983 - accuracy: 0.8148 - val_loss: 0.8568 - val_accuracy:
0.9167
Epoch 34/75
1/1 - 0s - loss: 0.4951 - accuracy: 0.8519 - val_loss: 0.4828 - val_accuracy:
0.9167
Epoch 35/75
1/1 - 0s - loss: 0.5055 - accuracy: 0.8333 - val_loss: 0.4203 - val_accuracy:
0.9583
Epoch 36/75
1/1 - 0s - loss: 0.4399 - accuracy: 0.8750 - val_loss: 0.2744 - val_accuracy:
0.9583
Epoch 37/75
1/1 - 0s - loss: 0.2871 - accuracy: 0.9120 - val_loss: 0.2323 - val_accuracy:
0.9167
Epoch 38/75
1/1 - 0s - loss: 0.2564 - accuracy: 0.9398 - val_loss: 0.2841 - val_accuracy:
0.9167
Epoch 39/75
1/1 - 0s - loss: 0.2188 - accuracy: 0.9398 - val_loss: 0.1651 - val_accuracy:
0.9583
Epoch 40/75
```

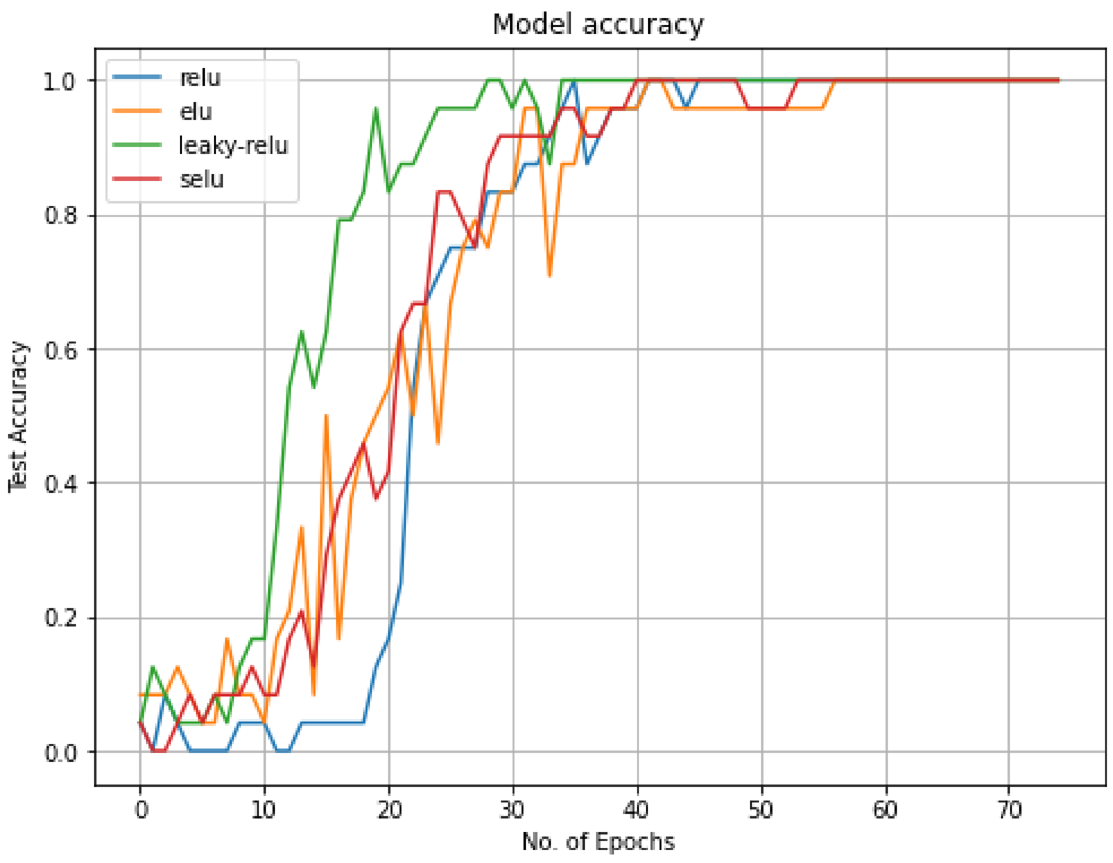
```
1/1 - 0s - loss: 0.2063 - accuracy: 0.9398 - val_loss: 0.2095 - val_accuracy:
0.9583
Epoch 41/75
1/1 - 0s - loss: 0.1482 - accuracy: 0.9537 - val_loss: 0.0215 - val_accuracy:
1.0000
Epoch 42/75
1/1 - 0s - loss: 0.1434 - accuracy: 0.9676 - val_loss: 1.5513e-04 -
val_accuracy: 1.0000
Epoch 43/75
1/1 - 0s - loss: 0.1292 - accuracy: 0.9630 - val_loss: 1.2865e-06 -
val_accuracy: 1.0000
Epoch 44/75
1/1 - 0s - loss: 0.0965 - accuracy: 0.9815 - val_loss: 9.4374e-08 -
val_accuracy: 1.0000
Epoch 45/75
1/1 - 0s - loss: 0.0777 - accuracy: 0.9769 - val_loss: 2.4835e-08 -
val_accuracy: 1.0000
Epoch 46/75
1/1 - 0s - loss: 0.0654 - accuracy: 0.9861 - val_loss: 3.4769e-08 -
val_accuracy: 1.0000
Epoch 47/75
1/1 - 0s - loss: 0.0620 - accuracy: 0.9815 - val_loss: 6.9539e-08 -
val_accuracy: 1.0000
Epoch 48/75
1/1 - 0s - loss: 0.0607 - accuracy: 0.9907 - val_loss: 3.3775e-06 -
val_accuracy: 1.0000
Epoch 49/75
1/1 - 0s - loss: 0.0548 - accuracy: 0.9907 - val_loss: 5.6207e-04 -
val_accuracy: 1.0000
Epoch 50/75
1/1 - 0s - loss: 0.0542 - accuracy: 0.9815 - val_loss: 0.0416 - val_accuracy:
0.9583
Epoch 51/75
1/1 - 0s - loss: 0.0445 - accuracy: 0.9861 - val_loss: 0.0888 - val_accuracy:
0.9583
Epoch 52/75
1/1 - 0s - loss: 0.0359 - accuracy: 0.9907 - val_loss: 0.0944 - val_accuracy:
0.9583
Epoch 53/75
1/1 - 0s - loss: 0.0216 - accuracy: 1.0000 - val_loss: 0.0636 - val_accuracy:
0.9583
Epoch 54/75
1/1 - 0s - loss: 0.0437 - accuracy: 0.9861 - val_loss: 0.0120 - val_accuracy:
1.0000
Epoch 55/75
1/1 - 0s - loss: 0.0427 - accuracy: 0.9815 - val_loss: 1.6049e-04 -
val_accuracy: 1.0000
Epoch 56/75
```

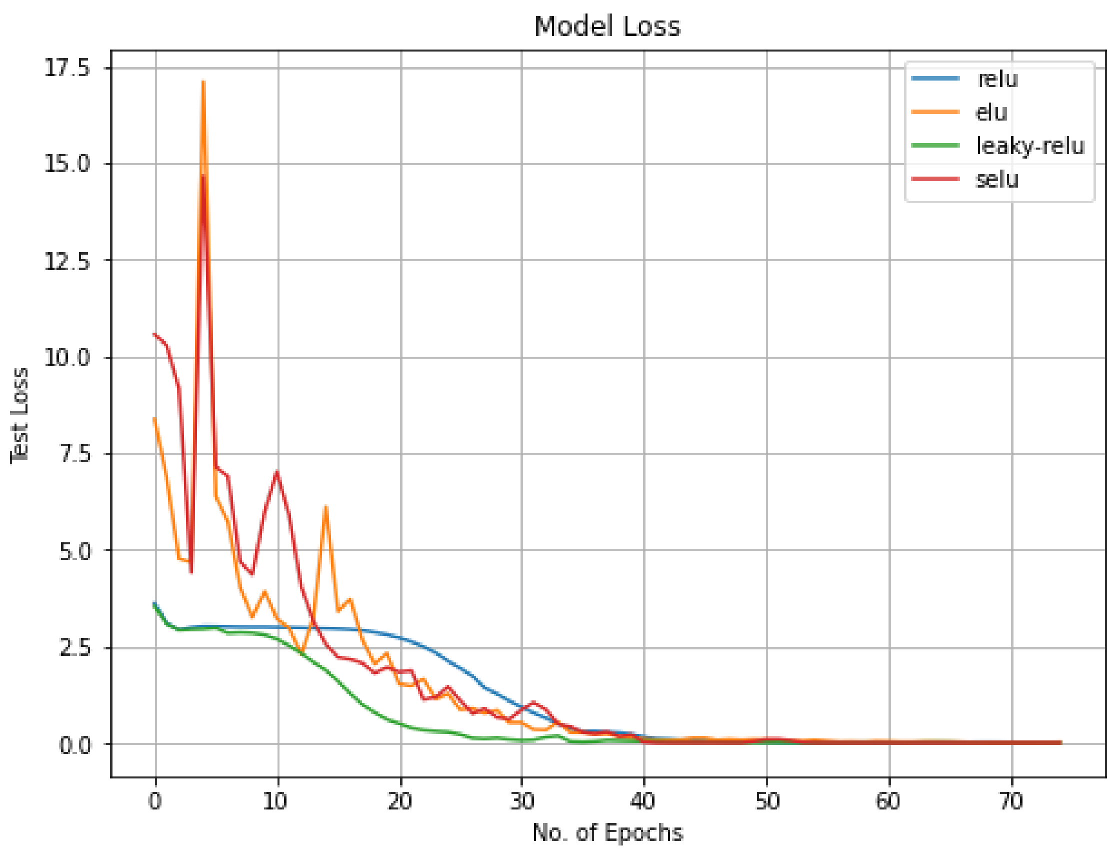
```
1/1 - 0s - loss: 0.0193 - accuracy: 0.9954 - val_loss: 6.1239e-06 -
val_accuracy: 1.0000
Epoch 57/75
1/1 - 0s - loss: 0.0284 - accuracy: 0.9954 - val_loss: 3.9736e-07 -
val_accuracy: 1.0000
Epoch 58/75
1/1 - 0s - loss: 0.0197 - accuracy: 0.9954 - val_loss: 2.9802e-08 -
val_accuracy: 1.0000
Epoch 59/75
1/1 - 0s - loss: 0.0140 - accuracy: 0.9954 - val_loss: 4.9671e-09 -
val_accuracy: 1.0000
Epoch 60/75
1/1 - 0s - loss: 0.0182 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 61/75
1/1 - 0s - loss: 0.0176 - accuracy: 0.9954 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 62/75
1/1 - 0s - loss: 0.0140 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 63/75
1/1 - 0s - loss: 0.0255 - accuracy: 0.9907 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 64/75
1/1 - 0s - loss: 0.0097 - accuracy: 0.9954 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 65/75
1/1 - 0s - loss: 0.0136 - accuracy: 0.9954 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 66/75
1/1 - 0s - loss: 0.0124 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 67/75
1/1 - 0s - loss: 0.0096 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 68/75
1/1 - 0s - loss: 0.0044 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 69/75
1/1 - 0s - loss: 0.0070 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 70/75
1/1 - 0s - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 71/75
1/1 - 0s - loss: 0.0107 - accuracy: 0.9954 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 72/75
```

```
1/1 - 0s - loss: 0.0080 - accuracy: 0.9954 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 73/75
1/1 - 0s - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 74/75
1/1 - 0s - loss: 0.0066 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
val_accuracy: 1.0000
Epoch 75/75
1/1 - 0s - loss: 0.0094 - accuracy: 1.0000 - val_loss: 9.9341e-09 -
val_accuracy: 1.0000
[<tensorflow.python.keras.callbacks.History object at 0x000002579C7A5550>,
<tensorflow.python.keras.callbacks.History object at 0x000002579FC706D0>,
<tensorflow.python.keras.callbacks.History object at 0x00000257A1383400>,
<tensorflow.python.keras.callbacks.History object at 0x00000257A25910A0>]
```

[31]:
```
# 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()
```

[32]:
```
plot_results(results_new, activation_list)
```





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.

```
[33]: 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 - 0s - loss: 3.0227 - accuracy: 0.0509 - val_loss: 3.3781 - val_accuracy:
0.0000e+00
Epoch 2/75
1/1 - 0s - loss: 3.9394 - accuracy: 0.0417 - val_loss: 2.9603 - val_accuracy:
0.1667
Epoch 3/75
1/1 - 0s - loss: 3.9321 - accuracy: 0.0787 - val_loss: 2.8888 - val_accuracy:
0.0833
Epoch 4/75
1/1 - 0s - loss: 3.2816 - accuracy: 0.0463 - val_loss: 3.0333 - val_accuracy:
0.0417
Epoch 5/75
1/1 - 0s - loss: 2.9845 - accuracy: 0.0880 - val_loss: 2.9694 - val_accuracy:
0.0000e+00
Epoch 6/75
1/1 - 0s - loss: 2.9109 - accuracy: 0.1157 - val_loss: 2.7385 - val_accuracy:
0.1667
Epoch 7/75
1/1 - 0s - loss: 3.5470 - accuracy: 0.0602 - val_loss: 3.0208 - val_accuracy:
0.0833
Epoch 8/75
1/1 - 0s - loss: 2.8829 - accuracy: 0.1389 - val_loss: 3.0583 - val_accuracy:
0.0000e+00
Epoch 9/75
1/1 - 0s - loss: 2.8753 - accuracy: 0.1481 - val_loss: 2.9957 - val_accuracy:
0.0000e+00
Epoch 10/75
1/1 - 0s - loss: 2.7962 - accuracy: 0.1435 - val_loss: 2.9133 - val_accuracy:
0.0417
Epoch 11/75
1/1 - 0s - loss: 2.7330 - accuracy: 0.1667 - val_loss: 2.7998 - val_accuracy:
0.0417
Epoch 12/75
1/1 - 0s - loss: 2.6063 - accuracy: 0.2731 - val_loss: 2.6821 - val_accuracy:
0.3333
Epoch 13/75
1/1 - 0s - loss: 2.4683 - accuracy: 0.3009 - val_loss: 2.5484 - val_accuracy:
0.5000
Epoch 14/75
1/1 - 0s - loss: 2.3334 - accuracy: 0.4074 - val_loss: 2.4161 - val_accuracy:
0.3750
Epoch 15/75
1/1 - 0s - loss: 2.2098 - accuracy: 0.4259 - val_loss: 2.2588 - val_accuracy:
0.4583
Epoch 16/75
1/1 - 0s - loss: 2.0446 - accuracy: 0.4583 - val_loss: 2.0264 - val_accuracy:
0.5417
Epoch 17/75
```

```
1/1 - 0s - loss: 1.7722 - accuracy: 0.6204 - val_loss: 1.7330 - val_accuracy:
0.7500
Epoch 18/75
1/1 - 0s - loss: 1.4859 - accuracy: 0.6574 - val_loss: 1.4182 - val_accuracy:
0.8750
Epoch 19/75
1/1 - 0s - loss: 1.3093 - accuracy: 0.6806 - val_loss: 1.0661 - val_accuracy:
0.8333
Epoch 20/75
1/1 - 0s - loss: 0.9455 - accuracy: 0.7963 - val_loss: 0.7589 - val_accuracy:
0.9167
Epoch 21/75
1/1 - 0s - loss: 0.8177 - accuracy: 0.7500 - val_loss: 0.5728 - val_accuracy:
0.9583
Epoch 22/75
1/1 - 0s - loss: 0.6548 - accuracy: 0.8102 - val_loss: 0.4279 - val_accuracy:
0.9167
Epoch 23/75
1/1 - 0s - loss: 0.5048 - accuracy: 0.8426 - val_loss: 0.3854 - val_accuracy:
0.9167
Epoch 24/75
1/1 - 0s - loss: 0.3736 - accuracy: 0.8843 - val_loss: 0.2579 - val_accuracy:
0.9167
Epoch 25/75
1/1 - 0s - loss: 0.2953 - accuracy: 0.9074 - val_loss: 0.2620 - val_accuracy:
0.9167
Epoch 26/75
1/1 - 0s - loss: 0.2226 - accuracy: 0.9398 - val_loss: 0.1895 - val_accuracy:
0.9583
Epoch 27/75
1/1 - 0s - loss: 0.1620 - accuracy: 0.9722 - val_loss: 0.1399 - val_accuracy:
1.0000
Epoch 28/75
1/1 - 0s - loss: 0.1342 - accuracy: 0.9676 - val_loss: 0.1085 - val_accuracy:
0.9583
Epoch 29/75
1/1 - 0s - loss: 0.1257 - accuracy: 0.9537 - val_loss: 0.1052 - val_accuracy:
0.9583
Epoch 30/75
1/1 - 0s - loss: 0.0834 - accuracy: 0.9769 - val_loss: 0.0618 - val_accuracy:
1.0000
Epoch 31/75
1/1 - 0s - loss: 0.0467 - accuracy: 0.9815 - val_loss: 0.0922 - val_accuracy:
0.9583
Epoch 32/75
1/1 - 0s - loss: 0.1059 - accuracy: 0.9676 - val_loss: 0.0412 - val_accuracy:
1.0000
Epoch 33/75
```

```
1/1 - 0s - loss: 0.0445 - accuracy: 0.9769 - val_loss: 0.0594 - val_accuracy:
1.0000
Epoch 34/75
1/1 - 0s - loss: 0.0344 - accuracy: 0.9954 - val_loss: 0.0868 - val_accuracy:
0.9583
Epoch 35/75
1/1 - 0s - loss: 0.0304 - accuracy: 0.9907 - val_loss: 0.0745 - val_accuracy:
0.9583
Epoch 36/75
1/1 - 0s - loss: 0.0324 - accuracy: 0.9769 - val_loss: 0.0302 - val_accuracy:
1.0000
Epoch 37/75
1/1 - 0s - loss: 0.0241 - accuracy: 0.9907 - val_loss: 0.0165 - val_accuracy:
1.0000
Epoch 38/75
1/1 - 0s - loss: 0.0142 - accuracy: 0.9954 - val_loss: 0.0117 - val_accuracy:
1.0000
Epoch 39/75
1/1 - 0s - loss: 0.0191 - accuracy: 0.9861 - val_loss: 0.0140 - val_accuracy:
1.0000
Epoch 40/75
1/1 - 0s - loss: 0.0187 - accuracy: 0.9907 - val_loss: 0.0305 - val_accuracy:
1.0000
Epoch 41/75
1/1 - 0s - loss: 0.0087 - accuracy: 1.0000 - val_loss: 0.0316 - val_accuracy:
1.0000
Epoch 42/75
1/1 - 0s - loss: 0.0158 - accuracy: 0.9907 - val_loss: 0.0333 - val_accuracy:
1.0000
Epoch 43/75
1/1 - 0s - loss: 0.0249 - accuracy: 0.9907 - val_loss: 0.0113 - val_accuracy:
1.0000
Epoch 44/75
1/1 - 0s - loss: 0.0079 - accuracy: 1.0000 - val_loss: 0.0031 - val_accuracy:
1.0000
Epoch 45/75
1/1 - 0s - loss: 0.0059 - accuracy: 1.0000 - val_loss: 0.0042 - val_accuracy:
1.0000
Epoch 46/75
1/1 - 0s - loss: 0.0070 - accuracy: 1.0000 - val_loss: 0.0052 - val_accuracy:
1.0000
Epoch 47/75
1/1 - 0s - loss: 0.0058 - accuracy: 1.0000 - val_loss: 0.0035 - val_accuracy:
1.0000
Epoch 48/75
1/1 - 0s - loss: 0.0059 - accuracy: 1.0000 - val_loss: 0.0037 - val_accuracy:
1.0000
Epoch 49/75
```

```
1/1 - 0s - loss: 0.0081 - accuracy: 1.0000 - val_loss: 0.0062 - val_accuracy:
1.0000
Epoch 50/75
1/1 - 0s - loss: 0.0046 - accuracy: 0.9954 - val_loss: 0.0173 - val_accuracy:
1.0000
Epoch 51/75
1/1 - 0s - loss: 0.0140 - accuracy: 0.9954 - val_loss: 0.0177 - val_accuracy:
1.0000
Epoch 52/75
1/1 - 0s - loss: 0.0107 - accuracy: 0.9954 - val_loss: 0.0100 - val_accuracy:
1.0000
Epoch 53/75
1/1 - 0s - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.0102 - val_accuracy:
1.0000
Epoch 54/75
1/1 - 0s - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.0111 - val_accuracy:
1.0000
Epoch 55/75
1/1 - 0s - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.0086 - val_accuracy:
1.0000
Epoch 56/75
1/1 - 0s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0061 - val_accuracy:
1.0000
Epoch 57/75
1/1 - 0s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0051 - val_accuracy:
1.0000
Epoch 58/75
1/1 - 0s - loss: 0.0066 - accuracy: 1.0000 - val_loss: 0.0128 - val_accuracy:
1.0000
Epoch 59/75
1/1 - 0s - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.0339 - val_accuracy:
1.0000
Epoch 60/75
1/1 - 0s - loss: 0.0130 - accuracy: 0.9954 - val_loss: 0.0288 - val_accuracy:
1.0000
Epoch 61/75
1/1 - 0s - loss: 0.0052 - accuracy: 1.0000 - val_loss: 0.0127 - val_accuracy:
1.0000
Epoch 62/75
1/1 - 0s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0049 - val_accuracy:
1.0000
Epoch 63/75
1/1 - 0s - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy:
1.0000
Epoch 64/75
1/1 - 0s - loss: 7.7043e-04 - accuracy: 1.0000 - val_loss: 8.6664e-04 -
val_accuracy: 1.0000
Epoch 65/75
```

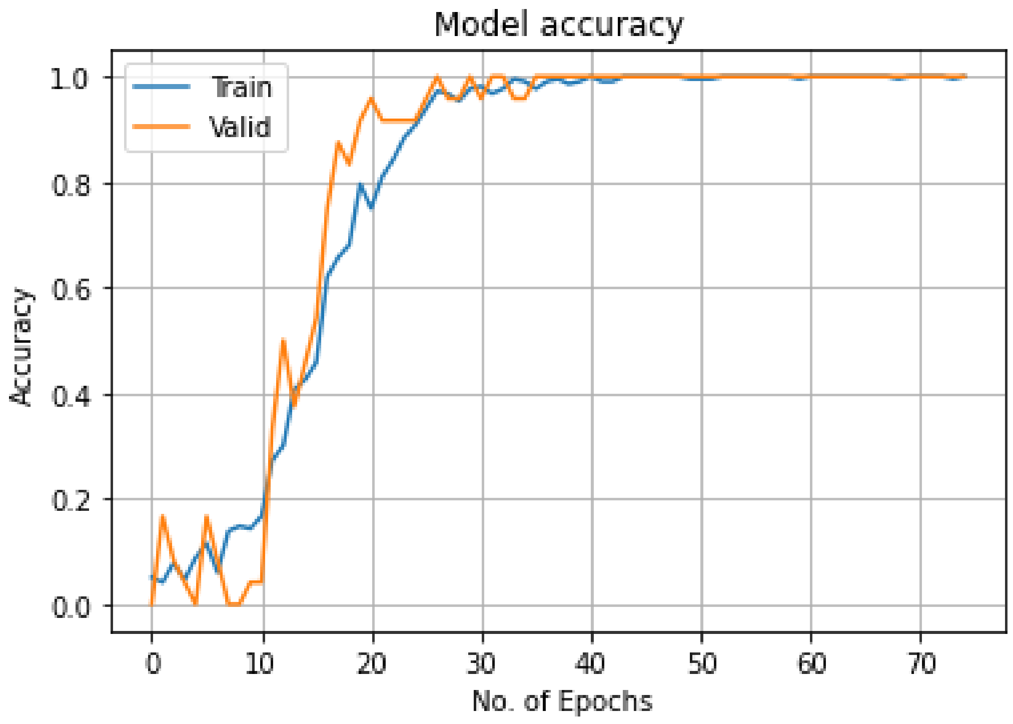
```
1/1 - 0s - loss: 0.0021 - accuracy: 1.0000 - val_loss: 6.7135e-04 -
val_accuracy: 1.0000
Epoch 66/75
1/1 - 0s - loss: 7.4025e-04 - accuracy: 1.0000 - val_loss: 7.5006e-04 -
val_accuracy: 1.0000
Epoch 67/75
1/1 - 0s - loss: 0.0015 - accuracy: 1.0000 - val_loss: 9.4125e-04 -
val_accuracy: 1.0000
Epoch 68/75
1/1 - 0s - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy:
1.0000
Epoch 69/75
1/1 - 0s - loss: 0.0191 - accuracy: 0.9954 - val_loss: 0.0010 - val_accuracy:
1.0000
Epoch 70/75
1/1 - 0s - loss: 2.1388e-04 - accuracy: 1.0000 - val_loss: 0.0011 -
val_accuracy: 1.0000
Epoch 71/75
1/1 - 0s - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.0014 - val_accuracy:
1.0000
Epoch 72/75
1/1 - 0s - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy:
1.0000
Epoch 73/75
1/1 - 0s - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0022 - val_accuracy:
1.0000
Epoch 74/75
1/1 - 0s - loss: 0.0059 - accuracy: 0.9954 - val_loss: 0.0021 - val_accuracy:
1.0000
Epoch 75/75
1/1 - 0s - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0019 - val_accuracy:
1.0000
```

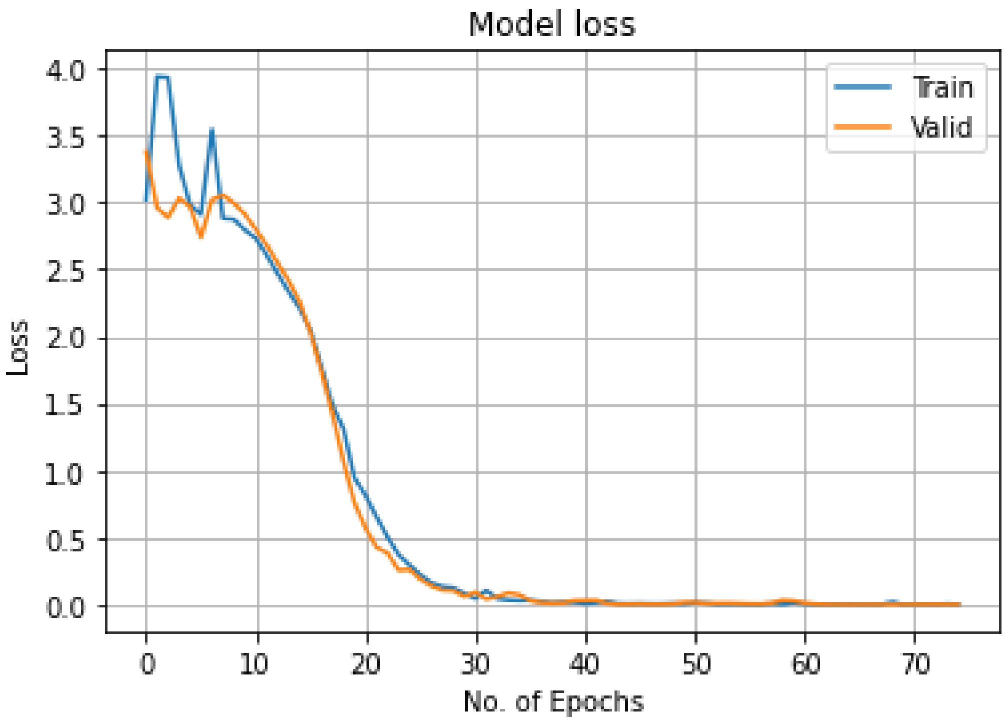
[36]:
```
result_score = model_final.evaluate(np.array(x_test),np.array(y_test),verbose=0)

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

```
Test Loss 0.2434
Test Accuracy 0.9375
```

[37]:
```
# 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%