Nama : Andrew

NIM : 2540119601

Kelas : LA09

Mata Kuliah : Deep Learning

Jurusan : Data Science

2A. [LO 3, 5 poin] Lakukan eksplorasi terhadap data tersebut dengan melihat histogram warnanya dan lakukan proses augmentasi data jika diperlukan. Dan kemudian lakukan resize resolusi gambar menjadi 64 x 64.

Library

```
# Library for load
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import random
```

Load Data

```
datas='/kaggle/input/dataset2c/Dataset2C'
class_data=os.listdir(datas)
for root,dirs,imgs in os.walk(datas):
    print(len(root),len(dirs),len(imgs))
```

```
33 5 0
50 0 50
45 0 50
46 0 50
48 0 50
51 0 46
```

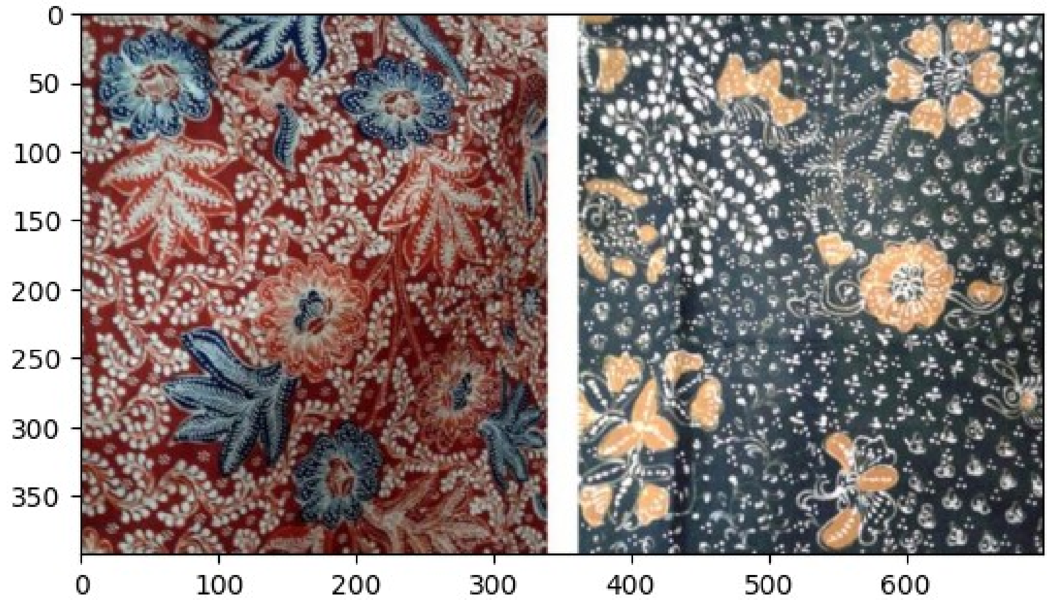
```
def random_choice(img_check = None):
    if img_check== None:
        class_name=random.choice(class_data)
        locate=os.path.join(datas,class_name)
        imgs=os.listdir(locate)
        rndm_img_name=random.choice(imgs)
        img_path=os.path.join(locate,rndm_img_name)
        img=plt.imread(img_path)
    elif img_check in class_data:
        class_name=img_check
        locate=os.path.join(datas,class_name)
        imgs=os.listdir(locate)
        rndm_img_name=random.choice(imgs)
        img_path=os.path.join(locate,rndm_img_name)
        img=plt.imread(img_path)
```

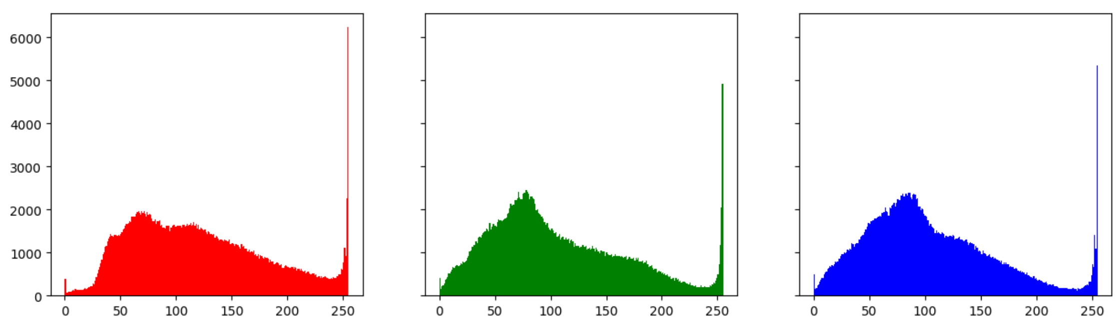
```
    else:
        print("Silakan pilih kelas yang benar !")
    print(f"{class_name} {img.shape}")
    return img

def show_img(img):
    plt.imshow(img)
    plt.axis(False)
    plt.show()

show_img(random_choice())
```

```
batik-priangan (464, 800, 3)
```

```
def build_histogram(image, bins=256):
    rgb_image = image
    plt.imshow(rgb_image)
    image_vector = rgb_image.reshape(1, -1, 3)
    # break into given number of bins
    div = 256 / bins
    bins_vector = (image_vector / div).astype(int)
    # get the red, green, and blue channels
    red = bins_vector[0, :, 0]
    green = bins_vector[0, :, 1]
    blue = bins_vector[0, :, 2]
    # build the histograms and display
    fig, axs = plt.subplots(1, 3, figsize=(15, 4), sharey=True)
    axs[0].hist(red, bins=bins, color='r')
    axs[1].hist(green, bins=bins, color='g')
    axs[2].hist(blue, bins=bins, color='b')
    plt.show()

build_histogram(random_choice(), bins=256)
```

```
batik-lasem (393, 700, 3)
```






```
# how much data in each class
for i in class_data:
    locate = os.path.join(datas,i)
    image = os.listdir(locate)
    print(f"Ada {len(image)} foto {i}")
```

```
Ada 50 foto batik-pekalongan
Ada 50 foto batik-lasem
Ada 50 foto batik-parang
Ada 50 foto batik-priangan
Ada 46 foto batik-megamendung
```

Image data yang didapatkan cukup kecil untuk dilakukan training data, maka dari itu diperlukan data augmentasi.

```
# libary for augmention data
from tqdm import tqdm
from skimage.transform import rotate, AffineTransform, warp
from skimage.util import random_noise
from skimage.filters import gaussian
```

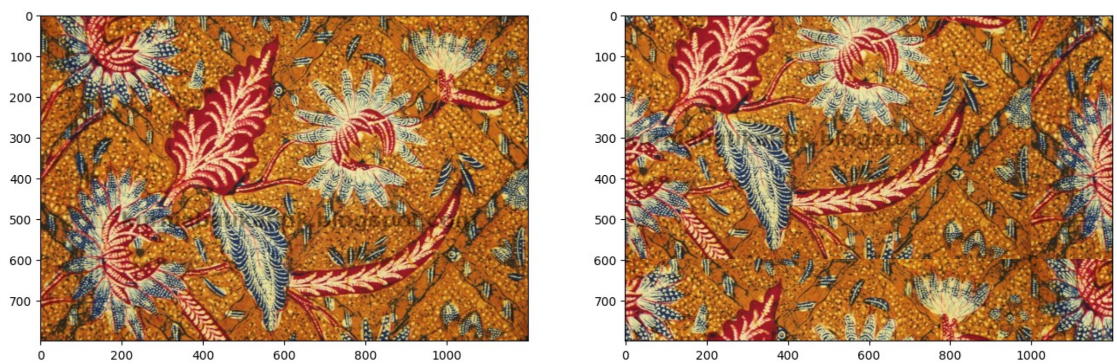
```python
import tensorflow as tf
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

```python
# Membuat dalam function agar mudah visualisasi
def visual(x,y):
    fig = plt.figure(figsize=(16,9))
    plt.subplot(1,2,1)
    plt.imshow(x)
    plt.subplot(1,2,2)
    plt.imshow(y)
```

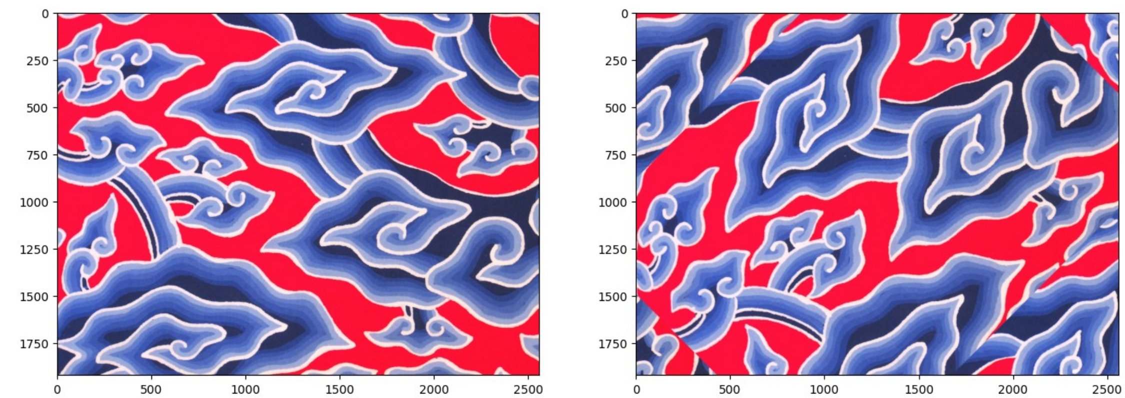
```python
# wrap
x = random_choice()
transform = AffineTransform(translation=(200,200))
wraps = warp(x,transform,mode='wrap')
visual(x,wraps)
```

```
batik-lasem (800, 1200, 3)
```



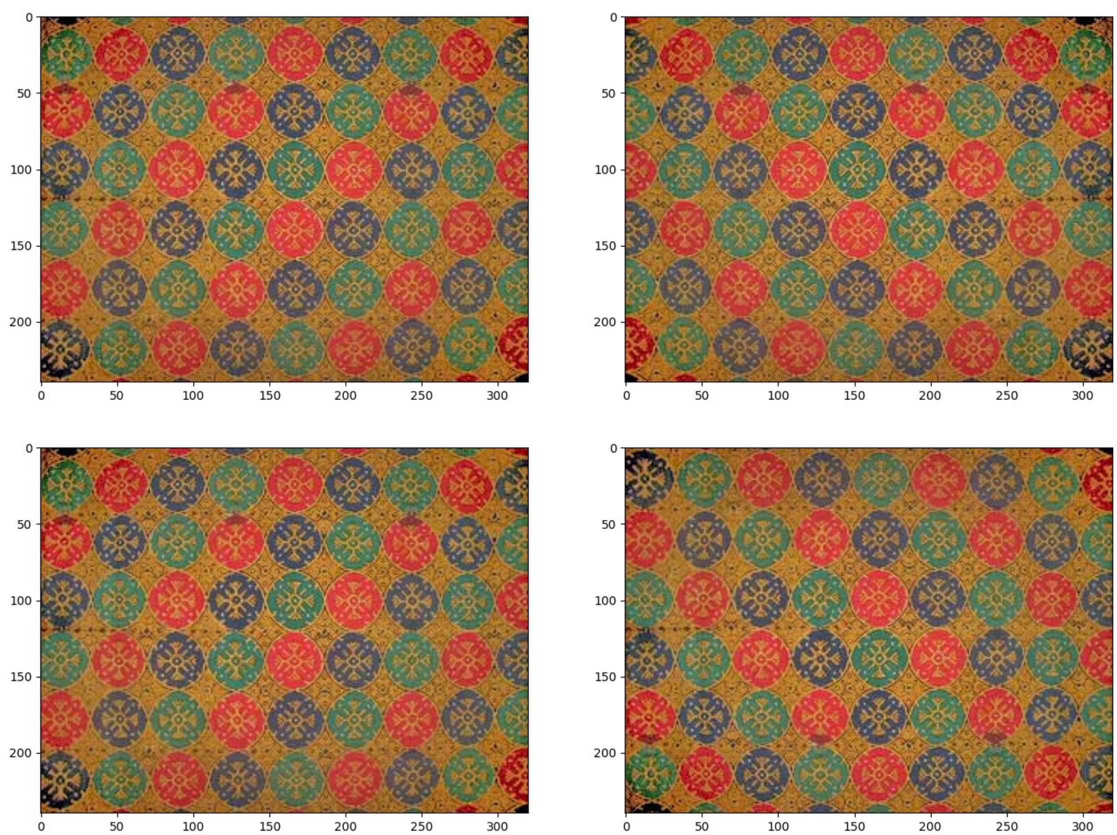
```python
# rotate
x = random_choice()
muter = rotate(x, angle=45, mode = 'wrap')
visual(x,muter)
```

```
batik-megamendung (1920, 2560, 3)
```




```
# flip
x = random_choice()

flip_kiri_kanan = tf.image.flip_left_right(x)
visual(x,flip_kiri_kanan)

flip_atas_bawah = tf.image.flip_up_down(x)
visual(x, flip_atas_bawah )
```

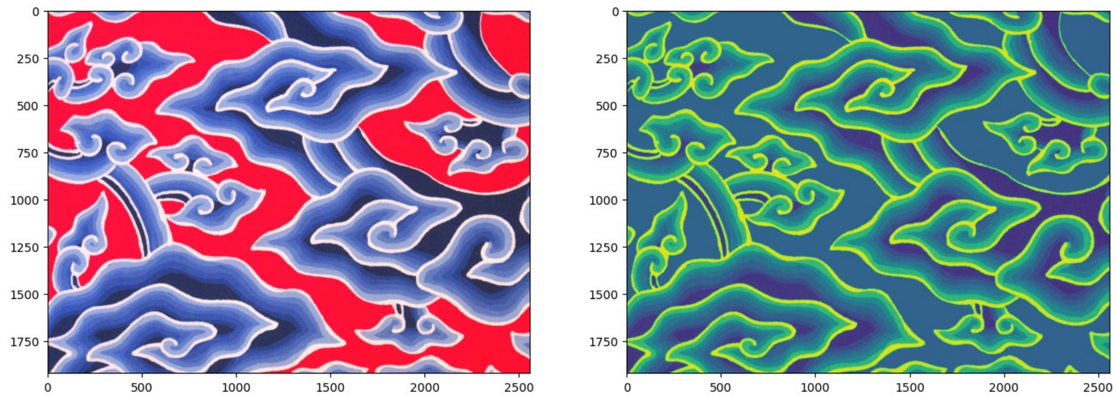
```
batik-pekalongan (240, 320, 3)
```




```
# greyscale
x = random_choice()
```

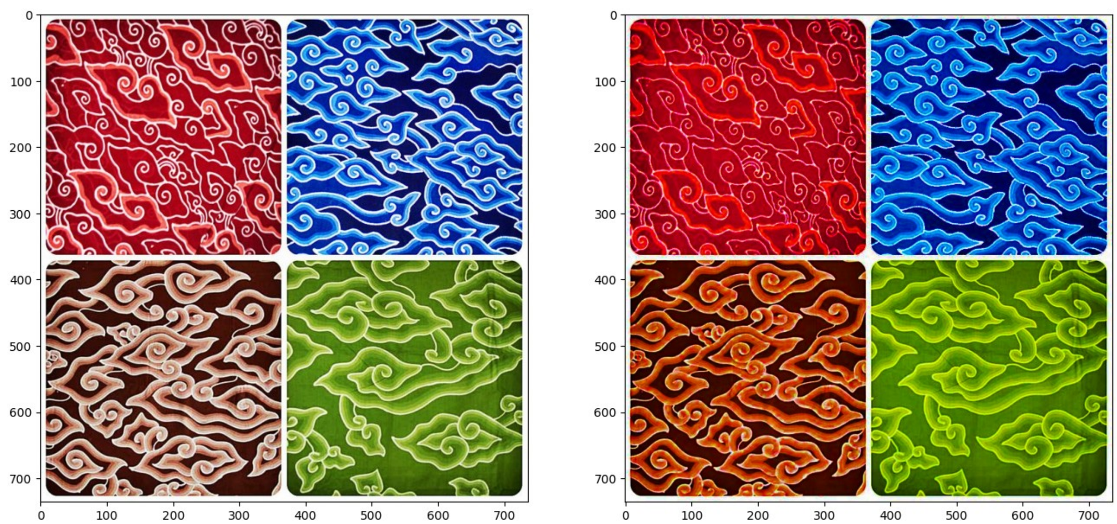
```
grey_image = tf.image.rgb_to_grayscale(x)
visual(x,grey_image)
```

```
batik-megamendung (1920, 2560, 3)
```



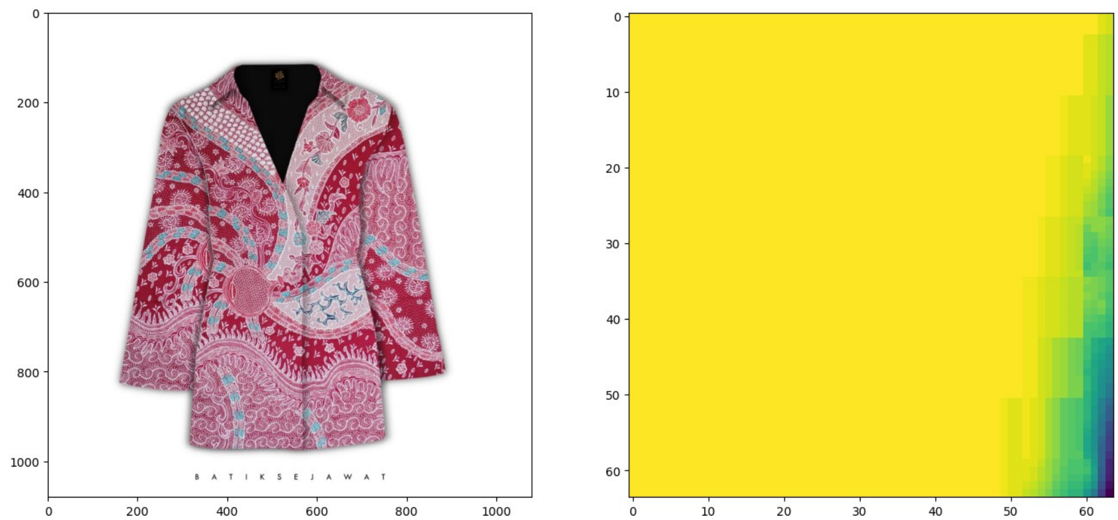
```
# saturation
x = random_choice()
satur = tf.image.adjust_saturation(x, 5)
visual(x,satur)
```

```
batik-megamendung (736, 736, 3)
```



```
# random crop
x = random_choice()
crops = tf.image.random_crop(x, size=[64, 64, 1])
visual(x,crops)
```

```
batik-priangan (1080, 1080, 3)
```

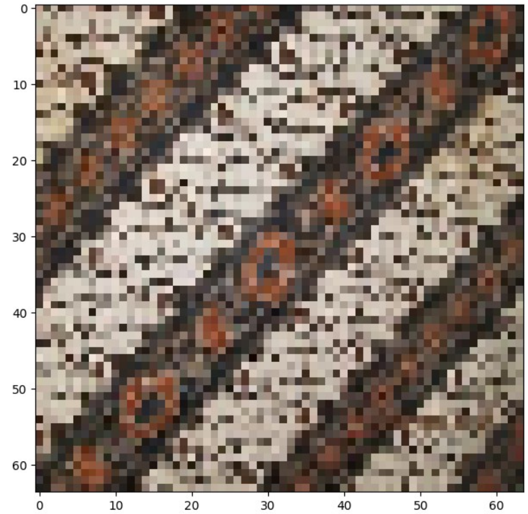


```python
# noise
x = random_choice()
noise = random_noise(x,var=0.22*0.5)
visual(x,noise)
```

```
batik-parang (1083, 750, 3)
```

```python
# See the image size in 64x64 to apply change in train data
size = 64
image = random_choice()
new_size = cv2.resize(image, (size,size))
visual(image,new_size)
```

```
batik-parang (225, 300, 3)
```



Cara augmentation tersebut tidak akan dipanggil lagi, karena keras memiliki function yang langsung memanggil augmentation.

2B. [LO 3, 5 poin] Pisahkan dataset menjadi 80% training set, 10% validation set dan 10% test set.

Menggunakan library dari split folder untuk memudahkan proses pembagian

```
!pip install split-folders
```

```
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
```

```
import splitfolders
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Membuat path folder output yang berisikan train, validation, dan test

```
os.makedirs('./output')
os.makedirs('./output/train')
os.makedirs('./output/val')
os.makedirs('./output/test')
os.listdir('./output')
```

```
['train', 'val', 'test']
```

```
splitfolders.ratio(datas, output="output", seed=1337, ratio=(.8, .1,
.1), group_prefix=None)
```

```
Copying files: 246 files [00:02, 101.63 files/s]
```

Data telah tercopy semua dan masuk kedalam folder, tetapi data masih belum di dalam suatu variabel. Maka dari itu diperlukan pembuatann variabel masing-masing train,val, dan test dengan bantuan image data generator

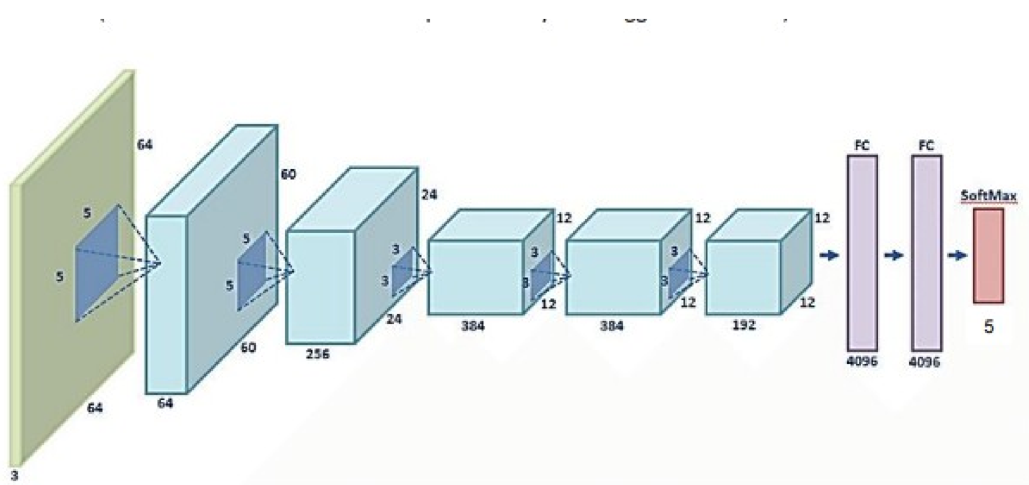
```
train_datagen=ImageDataGenerator(rescale=1./255,rotation_range=30,zoom
_range=0.2,brightness_range=[0.4,1.5],vertical_flip=True)
val_datagen=ImageDataGenerator(rescale=1./255)
test_datagen=ImageDataGenerator(rescale=1./255)

train=train_datagen.flow_from_directory('/kaggle/working/output/train'
, target_size=(64, 64), batch_size=32, class_mode='categorical',
seed=42)
val=val_datagen.flow_from_directory('/kaggle/working/output/val',targe
t_size=(64, 64),batch_size=32, class_mode='categorical', seed=42)
test=test_datagen.flow_from_directory('/kaggle/working/output/test',
target_size=(64, 64),batch_size=32, class_mode='categorical', seed=42)
```

```
Found 196 images belonging to 5 classes.
Found 24 images belonging to 5 classes.
Found 26 images belonging to 5 classes.
```

Dengan begitu image telah terbentuk dengan 196 train, 24 validation, dan 26 test.

Selanjutnya akan ke pembuatan alexnet arsitektur

2C. [LO 3, 15 poin] Buatlah arsitektur baseline sesuai dengan gambar arsitektur AlexNet berikut ini: (Catatan: Activation function tiap hidden layer menggunakan ReLU)



Dari arsiktektur ini terdapat 5 kali colvusion yang setelah ini baru di flaten dan masuk ke neural network. Pada akhirnya akan terdapat 5 layer output dengan activation fuction softmax.

```
# import library

import tensorflow as tf
```

```python
from tensorflow import keras
from tensorflow.keras import layers
from keras.utils.np_utils import to_categorical

# model building
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=(5,5), strides=(1,1),
activation='relu', input_shape=(64,64,3),padding="valid"),
    keras.layers.MaxPool2D(pool_size=(14,14), strides=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.MaxPool2D(pool_size=(2,2), strides=(2,2)),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
activation='relu',padding="same"),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.Conv2D(filters=192, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dense(5, activation='softmax')
])

model.compile(loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 60, 60, 64)        4864      
                                                                 
 max_pooling2d (MaxPooling2D  (None, 24, 24, 64)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 24, 24, 256)       147712    
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 12, 12, 256)      0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 12, 12, 384)       885120    
                                                                 
 conv2d_3 (Conv2D)           (None, 12, 12, 384)       1327488   
                                                                 
 conv2d_4 (Conv2D)           (None, 12, 12, 192)       663744    
                                                                 
 flatten (Flatten)           (None, 27648)             0         
                                                                 
 dense (Dense)               (None, 4096)              113250304 
```

```
 dense_1 (Dense)             (None, 4096)              16781312

 dense_2 (Dense)             (None, 5)                 20485

=================================================================
Total params: 133,081,029
Trainable params: 133,081,029
Non-trainable params: 0
_________________________________________________________________
```

```python
import time
root_logdir = os.path.join(os.curdir, "logs\\fit\\")

def get_run_logdir():
    run_id = time.strftime("run_%Y_%m_%d-%H_%M_%S")
    return os.path.join(root_logdir, run_id)

run_logdir = get_run_logdir()
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)

Epochs = 50
base_model=model.fit(train, epochs=Epochs,
                     validation_data=val,
                     validation_freq=1,callbacks=[tensorboard_cb])
```

```
Epoch 1/50
7/7 [==============================] - 10s 487ms/step - loss: 5.8119 -
accuracy: 0.1888 - val_loss: 1.6089 - val_accuracy: 0.2083
Epoch 2/50
7/7 [==============================] - 3s 421ms/step - loss: 1.6121 -
accuracy: 0.2245 - val_loss: 1.6108 - val_accuracy: 0.2083
Epoch 3/50
7/7 [==============================] - 3s 418ms/step - loss: 1.6117 -
accuracy: 0.1939 - val_loss: 1.6125 - val_accuracy: 0.2083
Epoch 4/50
7/7 [==============================] - 3s 457ms/step - loss: 1.6122 -
accuracy: 0.2041 - val_loss: 1.6089 - val_accuracy: 0.2083
Epoch 5/50
7/7 [==============================] - 3s 416ms/step - loss: 1.6195 -
accuracy: 0.1888 - val_loss: 1.6063 - val_accuracy: 0.2083
Epoch 6/50
7/7 [==============================] - 3s 404ms/step - loss: 1.6121 -
accuracy: 0.1684 - val_loss: 1.5852 - val_accuracy: 0.3333
Epoch 7/50
7/7 [==============================] - 3s 435ms/step - loss: 1.6164 -
accuracy: 0.2704 - val_loss: 1.5429 - val_accuracy: 0.3750
Epoch 8/50
7/7 [==============================] - 3s 484ms/step - loss: 1.6890 -
accuracy: 0.2296 - val_loss: 1.6062 - val_accuracy: 0.2083
Epoch 9/50
7/7 [==============================] - 3s 535ms/step - loss: 1.6088 -
```

```
accuracy: 0.2041 - val_loss: 1.5724 - val_accuracy: 0.2083
Epoch 10/50
7/7 [==============================] - 3s 459ms/step - loss: 1.5956 -
accuracy: 0.1888 - val_loss: 1.5384 - val_accuracy: 0.2500
Epoch 11/50
7/7 [==============================] - 3s 431ms/step - loss: 1.7979 -
accuracy: 0.2500 - val_loss: 1.5876 - val_accuracy: 0.2917
Epoch 12/50
7/7 [==============================] - 3s 414ms/step - loss: 2.2781 -
accuracy: 0.2500 - val_loss: 1.6330 - val_accuracy: 0.2083
Epoch 13/50
7/7 [==============================] - 3s 425ms/step - loss: 1.6104 -
accuracy: 0.2041 - val_loss: 1.5883 - val_accuracy: 0.2083
Epoch 14/50
7/7 [==============================] - 3s 399ms/step - loss: 1.6126 -
accuracy: 0.2194 - val_loss: 1.5470 - val_accuracy: 0.3333
Epoch 15/50
7/7 [==============================] - 3s 454ms/step - loss: 1.5648 -
accuracy: 0.2704 - val_loss: 1.4648 - val_accuracy: 0.3333
Epoch 16/50
7/7 [==============================] - 3s 425ms/step - loss: 1.6658 -
accuracy: 0.3061 - val_loss: 1.5798 - val_accuracy: 0.2083
Epoch 17/50
7/7 [==============================] - 3s 442ms/step - loss: 1.5906 -
accuracy: 0.2245 - val_loss: 1.5245 - val_accuracy: 0.3333
Epoch 18/50
7/7 [==============================] - 3s 426ms/step - loss: 1.5177 -
accuracy: 0.3010 - val_loss: 1.5181 - val_accuracy: 0.2917
Epoch 19/50
7/7 [==============================] - 3s 504ms/step - loss: 1.5488 -
accuracy: 0.2704 - val_loss: 1.4733 - val_accuracy: 0.2917
Epoch 20/50
7/7 [==============================] - 4s 479ms/step - loss: 1.4710 -
accuracy: 0.2857 - val_loss: 1.3283 - val_accuracy: 0.3750
Epoch 21/50
7/7 [==============================] - 3s 418ms/step - loss: 1.3367 -
accuracy: 0.3163 - val_loss: 1.4139 - val_accuracy: 0.3333
Epoch 22/50
7/7 [==============================] - 3s 469ms/step - loss: 2.5183 -
accuracy: 0.2551 - val_loss: 1.5801 - val_accuracy: 0.2083
Epoch 23/50
7/7 [==============================] - 3s 398ms/step - loss: 1.5230 -
accuracy: 0.2551 - val_loss: 1.4478 - val_accuracy: 0.3333
Epoch 24/50
7/7 [==============================] - 3s 446ms/step - loss: 1.4295 -
accuracy: 0.3418 - val_loss: 1.6336 - val_accuracy: 0.2083
Epoch 25/50
7/7 [==============================] - 3s 430ms/step - loss: 1.3712 -
accuracy: 0.3112 - val_loss: 1.5616 - val_accuracy: 0.3333
Epoch 26/50
```

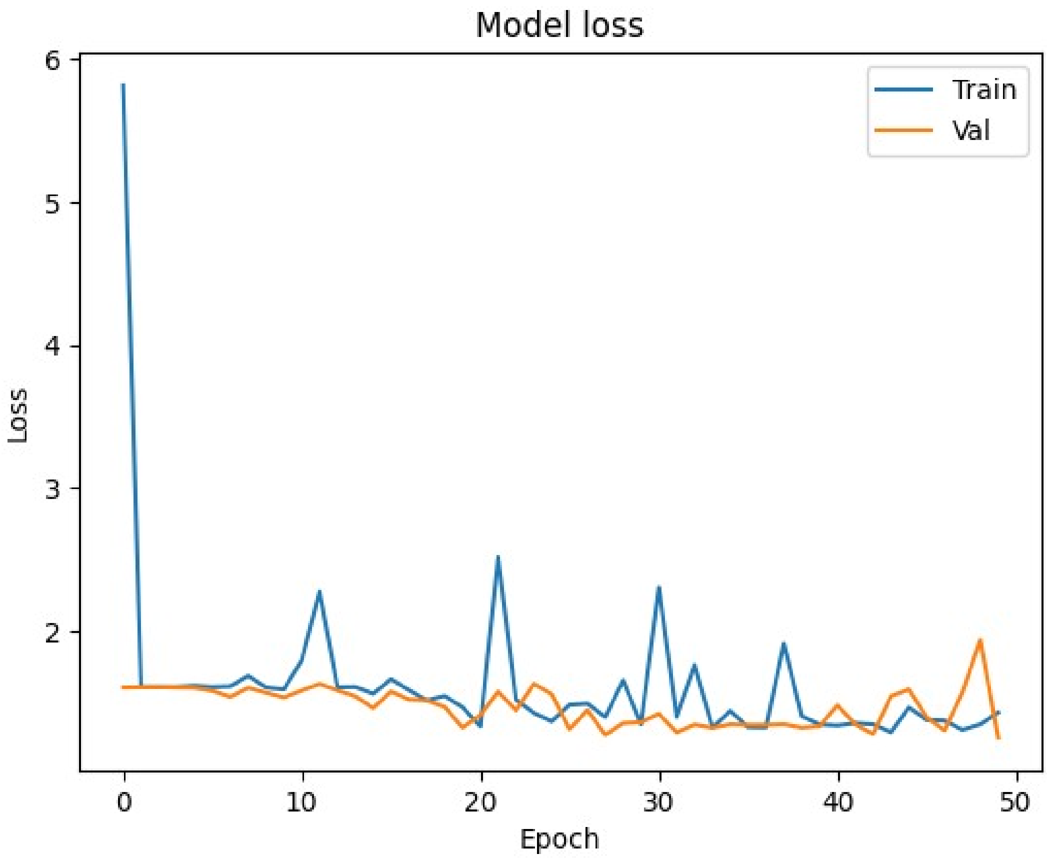
```
7/7 [==============================] - 3s 419ms/step - loss: 1.4874 -
accuracy: 0.2806 - val_loss: 1.3175 - val_accuracy: 0.3750
Epoch 27/50
7/7 [==============================] - 3s 400ms/step - loss: 1.4960 -
accuracy: 0.2449 - val_loss: 1.4498 - val_accuracy: 0.3333
Epoch 28/50
7/7 [==============================] - 3s 428ms/step - loss: 1.4014 -
accuracy: 0.3214 - val_loss: 1.2771 - val_accuracy: 0.4583
Epoch 29/50
7/7 [==============================] - 3s 445ms/step - loss: 1.6582 -
accuracy: 0.2857 - val_loss: 1.3598 - val_accuracy: 0.3333
Epoch 30/50
7/7 [==============================] - 3s 516ms/step - loss: 1.3510 -
accuracy: 0.3469 - val_loss: 1.3658 - val_accuracy: 0.3750
Epoch 31/50
7/7 [==============================] - 3s 424ms/step - loss: 2.3076 -
accuracy: 0.3367 - val_loss: 1.4236 - val_accuracy: 0.3750
Epoch 32/50
7/7 [==============================] - 3s 463ms/step - loss: 1.4006 -
accuracy: 0.3469 - val_loss: 1.2938 - val_accuracy: 0.3333
Epoch 33/50
7/7 [==============================] - 3s 493ms/step - loss: 1.7664 -
accuracy: 0.3214 - val_loss: 1.3496 - val_accuracy: 0.3750
Epoch 34/50
7/7 [==============================] - 3s 428ms/step - loss: 1.3268 -
accuracy: 0.3520 - val_loss: 1.3264 - val_accuracy: 0.3750
Epoch 35/50
7/7 [==============================] - 3s 426ms/step - loss: 1.4441 -
accuracy: 0.3776 - val_loss: 1.3511 - val_accuracy: 0.2917
Epoch 36/50
7/7 [==============================] - 3s 473ms/step - loss: 1.3306 -
accuracy: 0.3265 - val_loss: 1.3473 - val_accuracy: 0.3333
Epoch 37/50
7/7 [==============================] - 3s 402ms/step - loss: 1.3295 -
accuracy: 0.3469 - val_loss: 1.3474 - val_accuracy: 0.3333
Epoch 38/50
7/7 [==============================] - 3s 430ms/step - loss: 1.9149 -
accuracy: 0.2602 - val_loss: 1.3515 - val_accuracy: 0.2917
Epoch 39/50
7/7 [==============================] - 3s 432ms/step - loss: 1.4077 -
accuracy: 0.3112 - val_loss: 1.3276 - val_accuracy: 0.3750
Epoch 40/50
7/7 [==============================] - 4s 534ms/step - loss: 1.3509 -
accuracy: 0.3469 - val_loss: 1.3381 - val_accuracy: 0.3333
Epoch 41/50
7/7 [==============================] - 3s 450ms/step - loss: 1.3429 -
accuracy: 0.3520 - val_loss: 1.4839 - val_accuracy: 0.3333
Epoch 42/50
7/7 [==============================] - 3s 409ms/step - loss: 1.3596 -
accuracy: 0.3571 - val_loss: 1.3504 - val_accuracy: 0.3750
```

```
Epoch 43/50
7/7 [==============================] - 3s 441ms/step - loss: 1.3518 - 
accuracy: 0.3163 - val_loss: 1.2822 - val_accuracy: 0.4583
Epoch 44/50
7/7 [==============================] - 3s 432ms/step - loss: 1.2951 - 
accuracy: 0.3980 - val_loss: 1.5470 - val_accuracy: 0.4583
Epoch 45/50
7/7 [==============================] - 3s 439ms/step - loss: 1.4690 - 
accuracy: 0.3980 - val_loss: 1.5955 - val_accuracy: 0.3333
Epoch 46/50
7/7 [==============================] - 3s 428ms/step - loss: 1.3817 - 
accuracy: 0.3418 - val_loss: 1.3996 - val_accuracy: 0.2083
Epoch 47/50
7/7 [==============================] - 3s 419ms/step - loss: 1.3772 - 
accuracy: 0.3265 - val_loss: 1.3080 - val_accuracy: 0.5417
Epoch 48/50
7/7 [==============================] - 3s 420ms/step - loss: 1.3125 - 
accuracy: 0.3622 - val_loss: 1.5739 - val_accuracy: 0.4167
Epoch 49/50
7/7 [==============================] - 3s 438ms/step - loss: 1.3508 - 
accuracy: 0.3469 - val_loss: 1.9393 - val_accuracy: 0.2917
Epoch 50/50
7/7 [==============================] - 3s 495ms/step - loss: 1.4331 - 
accuracy: 0.3214 - val_loss: 1.2589 - val_accuracy: 0.3750
```

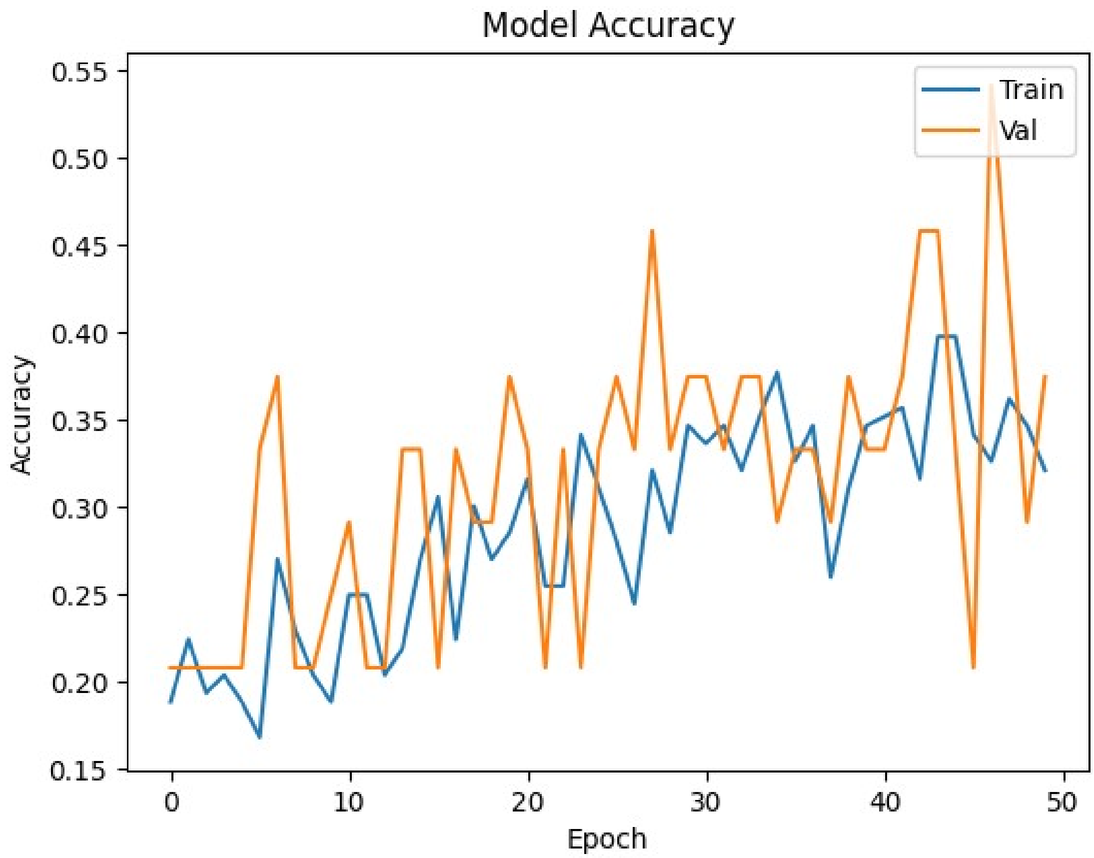
```
base_loss,base_acc=model.evaluate(test)
print("Accuracy of CNN Model: ",base_acc)
print("Loss of CNN Model: ",base_loss)
```

```
1/1 [==============================] - 0s 436ms/step - loss: 1.4387 - 
accuracy: 0.4615
Accuracy of CNN Model:  0.4615384638309479
Loss of CNN Model:  1.4387495517730713
```

```
# plotting model loss
plt.plot(base_model.history['loss'])
plt.plot(base_model.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```




```
# plotting accuracy
plt.plot(base_model.history['accuracy'])
plt.plot(base_model.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Dari sini dapat diketahui bahwa data yang dihasilkan overfit dan akurasi dari model walaupun cukup tinggi, yaitu 43%. Tapi saat di bandingkan dengan validasinya memiliki akurasi 20% begitu juga dengan akurasi dari test 23%nan.

2D. [LO 3, 25 poin] Modifikasi arsitektur AlexNet di atas agar mendapatkan hasil klasifikasi yang optimal. Kalian dapat menambahkan atau mengurangi arsitektur tersebutdan melakukan mengubah arsitektur pada nomor 2c dengan menggunakan dropout, batch normalization dan lain-lainnya. Dan selanjutnya lakukan proses tuning hyperparameter agar akurasi klasifikasinya meningkat. Berikan alasan mengapa modifikasi arsitektur dan metode tuning hyperparameter kalian lebih baik.

Mengganti epoch menjadi 100

Petama menggunakan struktur model yang sama lalu menambahkan dropout dan batch normalization

```
model_2a = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=(5,5), strides=(1,1),
activation='relu', input_shape=(64,64,3)),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(3,3),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
```

```
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(3,3),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=192, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(5, activation='softmax')
])
```

Tidak lupa juga menggunkan optimizer dari sgd

```
model_2a.compile(loss='categorical_crossentropy',optimizer=tf.optimize
rs.SGD(lr=0.001),metrics=['accuracy'])
model_2a.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_5 (Conv2D)           (None, 60, 60, 64)        4864      
                                                                 
 batch_normalization (BatchN  (None, 60, 60, 64)       256       
 ormalization)                                                   
                                                                 
 conv2d_6 (Conv2D)           (None, 20, 20, 256)       147712    
                                                                 
 batch_normalization_1 (Batc  (None, 20, 20, 256)      1024      
 hNormalization)                                                 
                                                                 
 conv2d_7 (Conv2D)           (None, 7, 7, 384)         885120    
                                                                 
 batch_normalization_2 (Batc  (None, 7, 7, 384)        1536      
 hNormalization)                                                 
                                                                 
 conv2d_8 (Conv2D)           (None, 7, 7, 384)         1327488   
                                                                 
 batch_normalization_3 (Batc  (None, 7, 7, 384)        1536      
 hNormalization)                                                 
                                                                 
 conv2d_9 (Conv2D)           (None, 7, 7, 192)         663744    
                                                                 
 batch_normalization_4 (Batc  (None, 7, 7, 192)        768       
 hNormalization)                                                 
```

```
 flatten_1 (Flatten)         (None, 9408)              0

 dense_3 (Dense)             (None, 4096)              38539264

 dropout (Dropout)           (None, 4096)              0

 dense_4 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense_5 (Dense)             (None, 5)                 20485

=================================================================
Total params: 58,375,109
Trainable params: 58,372,549
Non-trainable params: 2,560
_________________________________________________________________
```

```
Epochs = 100
```

Melihat hasil ouput dengan epoch 100

```
result1=model_2a.fit(train, epochs=Epochs,
                     validation_data=val,
                     validation_freq=1,callbacks=[tensorboard_cb])
```

```
Epoch 1/100
7/7 [==============================] - 5s 438ms/step - loss: 3.1543 - accuracy: 0.1888 - val_loss: 1.6269 - val_accuracy: 0.2083
Epoch 2/100
7/7 [==============================] - 3s 447ms/step - loss: 3.7248 - accuracy: 0.2704 - val_loss: 1.7120 - val_accuracy: 0.1667
Epoch 3/100
7/7 [==============================] - 3s 489ms/step - loss: 4.7848 - accuracy: 0.2449 - val_loss: 1.6086 - val_accuracy: 0.1250
Epoch 4/100
7/7 [==============================] - 3s 418ms/step - loss: 2.8737 - accuracy: 0.3214 - val_loss: 1.6495 - val_accuracy: 0.2917
Epoch 5/100
7/7 [==============================] - 3s 412ms/step - loss: 3.2475 - accuracy: 0.2500 - val_loss: 1.7229 - val_accuracy: 0.2083
Epoch 6/100
7/7 [==============================] - 3s 441ms/step - loss: 3.4391 - accuracy: 0.3061 - val_loss: 1.6066 - val_accuracy: 0.1667
Epoch 7/100
7/7 [==============================] - 3s 439ms/step - loss: 2.8106 - accuracy: 0.3163 - val_loss: 1.6350 - val_accuracy: 0.1667
Epoch 8/100
7/7 [==============================] - 3s 425ms/step - loss: 2.2747 - accuracy: 0.3827 - val_loss: 1.7108 - val_accuracy: 0.2500
```

```
Epoch 9/100
7/7 [==============================] - 4s 588ms/step - loss: 3.5572 -
accuracy: 0.2653 - val_loss: 1.6925 - val_accuracy: 0.2083
Epoch 10/100
7/7 [==============================] - 3s 499ms/step - loss: 2.5029 -
accuracy: 0.3010 - val_loss: 1.6992 - val_accuracy: 0.2500
Epoch 11/100
7/7 [==============================] - 3s 422ms/step - loss: 2.4912 -
accuracy: 0.3622 - val_loss: 1.7178 - val_accuracy: 0.2083
Epoch 12/100
7/7 [==============================] - 3s 498ms/step - loss: 2.4124 -
accuracy: 0.2704 - val_loss: 1.7214 - val_accuracy: 0.2083
Epoch 13/100
7/7 [==============================] - 3s 461ms/step - loss: 1.7198 -
accuracy: 0.4133 - val_loss: 1.6018 - val_accuracy: 0.2083
Epoch 14/100
7/7 [==============================] - 3s 509ms/step - loss: 2.1395 -
accuracy: 0.3010 - val_loss: 1.6575 - val_accuracy: 0.2083
Epoch 15/100
7/7 [==============================] - 3s 472ms/step - loss: 2.1863 -
accuracy: 0.3214 - val_loss: 1.6022 - val_accuracy: 0.2500
Epoch 16/100
7/7 [==============================] - 3s 522ms/step - loss: 2.1394 -
accuracy: 0.3418 - val_loss: 1.6610 - val_accuracy: 0.2083
Epoch 17/100
7/7 [==============================] - 3s 434ms/step - loss: 2.3769 -
accuracy: 0.3010 - val_loss: 1.6547 - val_accuracy: 0.2083
Epoch 18/100
7/7 [==============================] - 3s 477ms/step - loss: 1.8992 -
accuracy: 0.3980 - val_loss: 1.5834 - val_accuracy: 0.3333
Epoch 19/100
7/7 [==============================] - 4s 532ms/step - loss: 1.7989 -
accuracy: 0.3878 - val_loss: 1.7200 - val_accuracy: 0.2083
Epoch 20/100
7/7 [==============================] - 3s 478ms/step - loss: 2.3067 -
accuracy: 0.3418 - val_loss: 1.6142 - val_accuracy: 0.2500
Epoch 21/100
7/7 [==============================] - 3s 443ms/step - loss: 1.9658 -
accuracy: 0.3724 - val_loss: 1.7132 - val_accuracy: 0.2500
Epoch 22/100
7/7 [==============================] - 3s 441ms/step - loss: 2.1853 -
accuracy: 0.3622 - val_loss: 1.8422 - val_accuracy: 0.1667
Epoch 23/100
7/7 [==============================] - 3s 537ms/step - loss: 1.7294 -
accuracy: 0.3776 - val_loss: 1.7154 - val_accuracy: 0.2083
Epoch 24/100
7/7 [==============================] - 3s 418ms/step - loss: 1.7534 -
accuracy: 0.4031 - val_loss: 1.7372 - val_accuracy: 0.2083
Epoch 25/100
7/7 [==============================] - 3s 504ms/step - loss: 1.7160 -
```

```
accuracy: 0.4031 - val_loss: 1.8148 - val_accuracy: 0.2083
Epoch 26/100
7/7 [==============================] - 3s 471ms/step - loss: 2.4169 -
accuracy: 0.3520 - val_loss: 2.0780 - val_accuracy: 0.2083
Epoch 27/100
7/7 [==============================] - 3s 483ms/step - loss: 2.0772 -
accuracy: 0.3980 - val_loss: 1.7902 - val_accuracy: 0.2083
Epoch 28/100
7/7 [==============================] - 3s 495ms/step - loss: 1.6656 -
accuracy: 0.3980 - val_loss: 2.3876 - val_accuracy: 0.2083
Epoch 29/100
7/7 [==============================] - 4s 584ms/step - loss: 1.8478 -
accuracy: 0.4235 - val_loss: 1.7952 - val_accuracy: 0.2083
Epoch 30/100
7/7 [==============================] - 3s 458ms/step - loss: 1.6532 -
accuracy: 0.4082 - val_loss: 2.0308 - val_accuracy: 0.2083
Epoch 31/100
7/7 [==============================] - 3s 455ms/step - loss: 1.5515 -
accuracy: 0.3929 - val_loss: 3.3186 - val_accuracy: 0.2083
Epoch 32/100
7/7 [==============================] - 3s 461ms/step - loss: 1.9212 -
accuracy: 0.3520 - val_loss: 1.9222 - val_accuracy: 0.1667
Epoch 33/100
7/7 [==============================] - 3s 493ms/step - loss: 1.7779 -
accuracy: 0.4337 - val_loss: 1.8462 - val_accuracy: 0.2083
Epoch 34/100
7/7 [==============================] - 3s 530ms/step - loss: 1.7512 -
accuracy: 0.4439 - val_loss: 2.2396 - val_accuracy: 0.2083
Epoch 35/100
7/7 [==============================] - 3s 458ms/step - loss: 1.8238 -
accuracy: 0.3827 - val_loss: 1.9774 - val_accuracy: 0.1667
Epoch 36/100
7/7 [==============================] - 3s 471ms/step - loss: 1.3847 -
accuracy: 0.4694 - val_loss: 1.9299 - val_accuracy: 0.2500
Epoch 37/100
7/7 [==============================] - 3s 467ms/step - loss: 1.4673 -
accuracy: 0.4439 - val_loss: 1.9868 - val_accuracy: 0.2917
Epoch 38/100
7/7 [==============================] - 4s 518ms/step - loss: 1.3645 -
accuracy: 0.4643 - val_loss: 1.6893 - val_accuracy: 0.2500
Epoch 39/100
7/7 [==============================] - 3s 492ms/step - loss: 1.6211 -
accuracy: 0.4184 - val_loss: 2.2585 - val_accuracy: 0.2917
Epoch 40/100
7/7 [==============================] - 3s 465ms/step - loss: 1.7617 -
accuracy: 0.3878 - val_loss: 1.5877 - val_accuracy: 0.2917
Epoch 41/100
7/7 [==============================] - 3s 446ms/step - loss: 1.3110 -
accuracy: 0.4898 - val_loss: 2.1687 - val_accuracy: 0.2083
Epoch 42/100
```

```
7/7 [==============================] - 3s 457ms/step - loss: 1.6399 -
accuracy: 0.3878 - val_loss: 2.0065 - val_accuracy: 0.2917
Epoch 43/100
7/7 [==============================] - 3s 470ms/step - loss: 1.9832 -
accuracy: 0.4388 - val_loss: 1.8937 - val_accuracy: 0.2917
Epoch 44/100
7/7 [==============================] - 3s 449ms/step - loss: 1.6482 -
accuracy: 0.3878 - val_loss: 1.7521 - val_accuracy: 0.3333
Epoch 45/100
7/7 [==============================] - 3s 452ms/step - loss: 1.4532 -
accuracy: 0.4235 - val_loss: 1.6015 - val_accuracy: 0.3333
Epoch 46/100
7/7 [==============================] - 3s 452ms/step - loss: 1.4157 -
accuracy: 0.4643 - val_loss: 1.4548 - val_accuracy: 0.2917
Epoch 47/100
7/7 [==============================] - 3s 485ms/step - loss: 1.3707 -
accuracy: 0.4796 - val_loss: 1.4855 - val_accuracy: 0.3750
Epoch 48/100
7/7 [==============================] - 4s 512ms/step - loss: 1.2611 -
accuracy: 0.5357 - val_loss: 1.6431 - val_accuracy: 0.2917
Epoch 49/100
7/7 [==============================] - 3s 489ms/step - loss: 1.8451 -
accuracy: 0.4082 - val_loss: 1.7997 - val_accuracy: 0.3333
Epoch 50/100
7/7 [==============================] - 3s 443ms/step - loss: 1.4387 -
accuracy: 0.4694 - val_loss: 1.5825 - val_accuracy: 0.4167
Epoch 51/100
7/7 [==============================] - 3s 423ms/step - loss: 1.3659 -
accuracy: 0.4592 - val_loss: 1.4555 - val_accuracy: 0.4583
Epoch 52/100
7/7 [==============================] - 3s 476ms/step - loss: 1.4158 -
accuracy: 0.4694 - val_loss: 1.6162 - val_accuracy: 0.3750
Epoch 53/100
7/7 [==============================] - 3s 484ms/step - loss: 1.3876 -
accuracy: 0.5000 - val_loss: 1.6227 - val_accuracy: 0.4167
Epoch 54/100
7/7 [==============================] - 3s 451ms/step - loss: 1.3025 -
accuracy: 0.5306 - val_loss: 1.5654 - val_accuracy: 0.4167
Epoch 55/100
7/7 [==============================] - 3s 462ms/step - loss: 1.4077 -
accuracy: 0.5255 - val_loss: 1.5654 - val_accuracy: 0.4583
Epoch 56/100
7/7 [==============================] - 3s 422ms/step - loss: 1.2644 -
accuracy: 0.4694 - val_loss: 1.8652 - val_accuracy: 0.4167
Epoch 57/100
7/7 [==============================] - 3s 468ms/step - loss: 1.5009 -
accuracy: 0.4541 - val_loss: 1.7626 - val_accuracy: 0.2500
Epoch 58/100
7/7 [==============================] - 4s 521ms/step - loss: 1.3030 -
accuracy: 0.5153 - val_loss: 2.3015 - val_accuracy: 0.2500
```

```
Epoch 59/100
7/7 [==============================] - 3s 477ms/step - loss: 1.2483 -
accuracy: 0.5255 - val_loss: 1.9992 - val_accuracy: 0.2500
Epoch 60/100
7/7 [==============================] - 3s 437ms/step - loss: 1.3376 -
accuracy: 0.5000 - val_loss: 1.5540 - val_accuracy: 0.3333
Epoch 61/100
7/7 [==============================] - 3s 429ms/step - loss: 1.2921 -
accuracy: 0.5612 - val_loss: 1.5445 - val_accuracy: 0.2083
Epoch 62/100
7/7 [==============================] - 3s 427ms/step - loss: 1.3463 -
accuracy: 0.4745 - val_loss: 1.5787 - val_accuracy: 0.4167
Epoch 63/100
7/7 [==============================] - 3s 467ms/step - loss: 1.2937 -
accuracy: 0.4898 - val_loss: 1.5895 - val_accuracy: 0.3333
Epoch 64/100
7/7 [==============================] - 3s 469ms/step - loss: 1.1335 -
accuracy: 0.5612 - val_loss: 2.1612 - val_accuracy: 0.2500
Epoch 65/100
7/7 [==============================] - 3s 463ms/step - loss: 1.4004 -
accuracy: 0.4694 - val_loss: 1.7597 - val_accuracy: 0.4167
Epoch 66/100
7/7 [==============================] - 3s 456ms/step - loss: 1.1327 -
accuracy: 0.5663 - val_loss: 1.8021 - val_accuracy: 0.3750
Epoch 67/100
7/7 [==============================] - 3s 456ms/step - loss: 1.1973 -
accuracy: 0.5153 - val_loss: 1.7658 - val_accuracy: 0.2500
Epoch 68/100
7/7 [==============================] - 4s 519ms/step - loss: 1.2873 -
accuracy: 0.4949 - val_loss: 1.5708 - val_accuracy: 0.3750
Epoch 69/100
7/7 [==============================] - 3s 431ms/step - loss: 1.1137 -
accuracy: 0.5663 - val_loss: 2.2014 - val_accuracy: 0.1667
Epoch 70/100
7/7 [==============================] - 3s 450ms/step - loss: 1.6000 -
accuracy: 0.4388 - val_loss: 2.0689 - val_accuracy: 0.2083
Epoch 71/100
7/7 [==============================] - 3s 423ms/step - loss: 1.1620 -
accuracy: 0.5306 - val_loss: 2.1811 - val_accuracy: 0.2917
Epoch 72/100
7/7 [==============================] - 3s 455ms/step - loss: 1.3460 -
accuracy: 0.4745 - val_loss: 1.5752 - val_accuracy: 0.3333
Epoch 73/100
7/7 [==============================] - 3s 442ms/step - loss: 1.0981 -
accuracy: 0.5306 - val_loss: 1.9738 - val_accuracy: 0.3750
Epoch 74/100
7/7 [==============================] - 3s 468ms/step - loss: 1.2465 -
accuracy: 0.5051 - val_loss: 1.6216 - val_accuracy: 0.2917
Epoch 75/100
7/7 [==============================] - 3s 431ms/step - loss: 1.2177 -
```

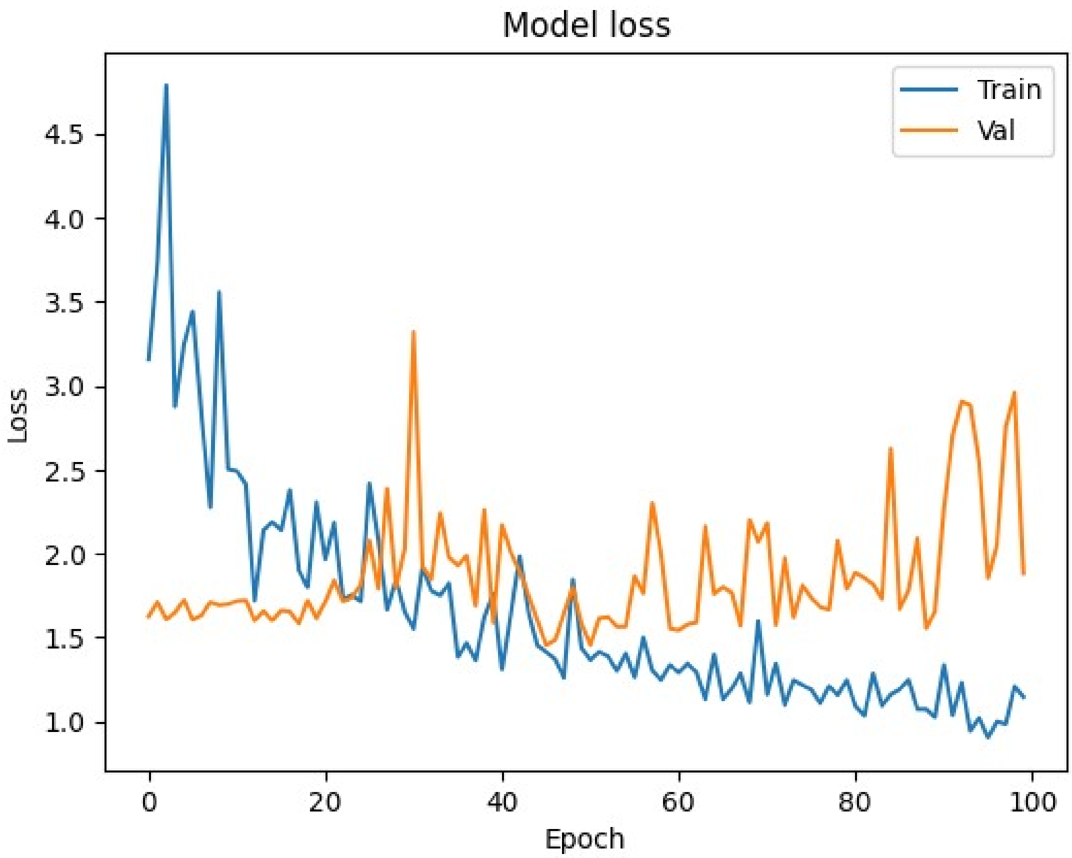
```
accuracy: 0.5000 - val_loss: 1.8123 - val_accuracy: 0.4167
Epoch 76/100
7/7 [==============================] - 3s 394ms/step - loss: 1.1909 -
accuracy: 0.5408 - val_loss: 1.7372 - val_accuracy: 0.3750
Epoch 77/100
7/7 [==============================] - 3s 411ms/step - loss: 1.1089 -
accuracy: 0.5306 - val_loss: 1.6813 - val_accuracy: 0.3333
Epoch 78/100
7/7 [==============================] - 3s 493ms/step - loss: 1.2111 -
accuracy: 0.4949 - val_loss: 1.6666 - val_accuracy: 0.3333
Epoch 79/100
7/7 [==============================] - 3s 408ms/step - loss: 1.1582 -
accuracy: 0.5765 - val_loss: 2.0764 - val_accuracy: 0.2500
Epoch 80/100
7/7 [==============================] - 3s 444ms/step - loss: 1.2471 -
accuracy: 0.5612 - val_loss: 1.7894 - val_accuracy: 0.3333
Epoch 81/100
7/7 [==============================] - 3s 511ms/step - loss: 1.0927 -
accuracy: 0.5459 - val_loss: 1.8864 - val_accuracy: 0.3333
Epoch 82/100
7/7 [==============================] - 3s 500ms/step - loss: 1.0362 -
accuracy: 0.5510 - val_loss: 1.8545 - val_accuracy: 0.3333
Epoch 83/100
7/7 [==============================] - 3s 439ms/step - loss: 1.2863 -
accuracy: 0.5612 - val_loss: 1.8190 - val_accuracy: 0.2917
Epoch 84/100
7/7 [==============================] - 3s 462ms/step - loss: 1.0938 -
accuracy: 0.6122 - val_loss: 1.7293 - val_accuracy: 0.3750
Epoch 85/100
7/7 [==============================] - 3s 422ms/step - loss: 1.1626 -
accuracy: 0.5969 - val_loss: 2.6240 - val_accuracy: 0.1667
Epoch 86/100
7/7 [==============================] - 3s 525ms/step - loss: 1.1928 -
accuracy: 0.5510 - val_loss: 1.6691 - val_accuracy: 0.3333
Epoch 87/100
7/7 [==============================] - 3s 458ms/step - loss: 1.2487 -
accuracy: 0.4796 - val_loss: 1.7804 - val_accuracy: 0.4583
Epoch 88/100
7/7 [==============================] - 4s 562ms/step - loss: 1.0748 -
accuracy: 0.5357 - val_loss: 2.0931 - val_accuracy: 0.4583
Epoch 89/100
7/7 [==============================] - 3s 510ms/step - loss: 1.0740 -
accuracy: 0.5969 - val_loss: 1.5561 - val_accuracy: 0.4167
Epoch 90/100
7/7 [==============================] - 3s 460ms/step - loss: 1.0262 -
accuracy: 0.6173 - val_loss: 1.6550 - val_accuracy: 0.4583
Epoch 91/100
7/7 [==============================] - 3s 452ms/step - loss: 1.3373 -
accuracy: 0.5357 - val_loss: 2.2680 - val_accuracy: 0.3750
Epoch 92/100
```

```
7/7 [==============================] - 3s 469ms/step - loss: 1.0386 -
accuracy: 0.5612 - val_loss: 2.7039 - val_accuracy: 0.2083
Epoch 93/100
7/7 [==============================] - 3s 512ms/step - loss: 1.2325 -
accuracy: 0.5408 - val_loss: 2.9054 - val_accuracy: 0.3750
Epoch 94/100
7/7 [==============================] - 3s 435ms/step - loss: 0.9458 -
accuracy: 0.6276 - val_loss: 2.8836 - val_accuracy: 0.3750
Epoch 95/100
7/7 [==============================] - 3s 479ms/step - loss: 1.0231 -
accuracy: 0.5714 - val_loss: 2.5392 - val_accuracy: 0.2917
Epoch 96/100
7/7 [==============================] - 3s 431ms/step - loss: 0.9059 -
accuracy: 0.6480 - val_loss: 1.8546 - val_accuracy: 0.4583
Epoch 97/100
7/7 [==============================] - 3s 418ms/step - loss: 1.0006 -
accuracy: 0.6327 - val_loss: 2.0512 - val_accuracy: 0.4583
Epoch 98/100
7/7 [==============================] - 4s 513ms/step - loss: 0.9857 -
accuracy: 0.6020 - val_loss: 2.7524 - val_accuracy: 0.2917
Epoch 99/100
7/7 [==============================] - 3s 446ms/step - loss: 1.2107 -
accuracy: 0.6020 - val_loss: 2.9587 - val_accuracy: 0.3333
Epoch 100/100
7/7 [==============================] - 3s 434ms/step - loss: 1.1479 -
accuracy: 0.6122 - val_loss: 1.8818 - val_accuracy: 0.4167
```

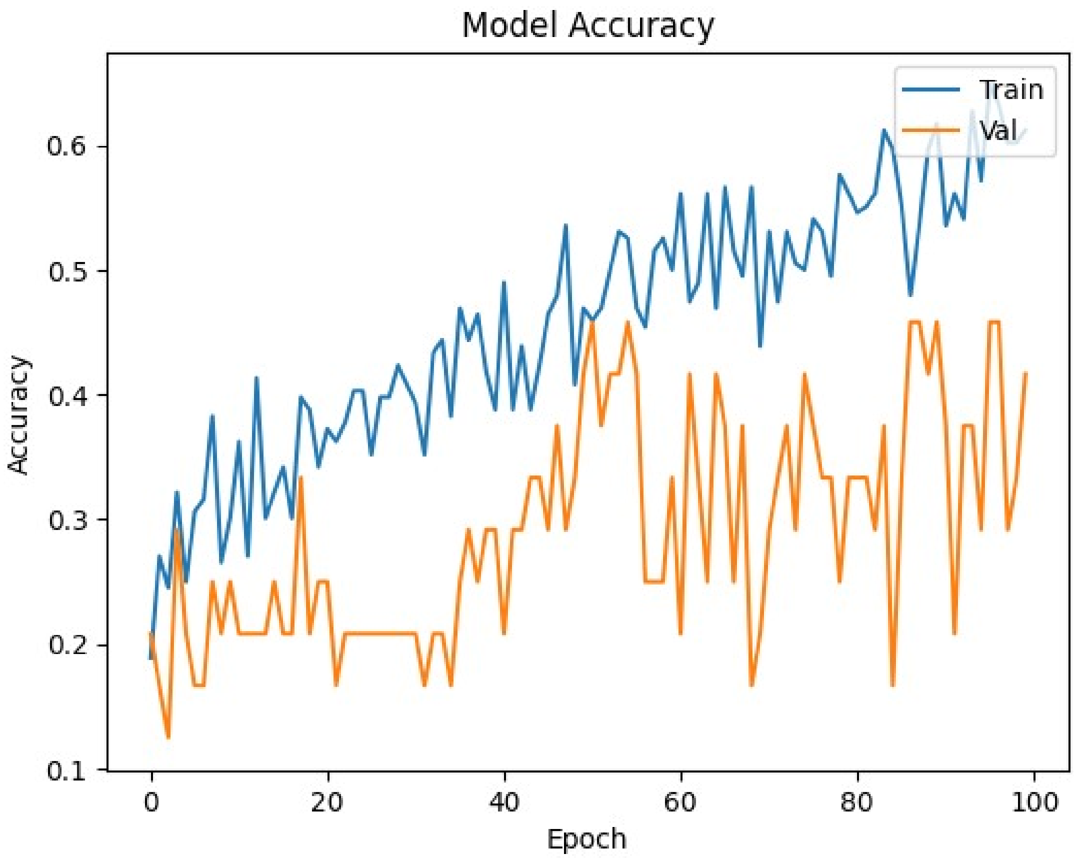
```python
loss1,acc1=model_2a.evaluate(test)
print("Accuracy of CNN Model: ",acc1)
print("Loss of CNN Model: ",loss1)
```

```
1/1 [==============================] - 0s 338ms/step - loss: 2.9222 -
accuracy: 0.3077
Accuracy of CNN Model:  0.3076923191547394
Loss of CNN Model:  2.9221997261047363
```

```python
# plotting model loss
plt.plot(result1.history['loss'])
plt.plot(result1.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```




```
# plotting accuracy
plt.plot(result1.history['accuracy'])
plt.plot(result1.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Dari sini dapat dilihat peningkatan terdapat model yang didapatkan adalah 62 persen. Untuk akurasi validasi dengan akurasi dengan test juga mengalami penningkatan, yaitu 41% untuk validasi dan 30% untuk test.

Untuk plot dar.i model loss masih menunjukan hasil yang tidak baik, tetapi sudah mulai terlihat garis train hampir menuju 0.

Model ke 2, didalam model ini akan ada penambahan max pool. Dengan menggunkanan max pool dapat membuat fitur yang ada didalam image lebih sharper. max pool akan di taruh stelah convulsion 1 dan akhir. Sehingga akan terdapat 2 max pool. Dan juga menambahkan learning rate.

```
model_2b = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=(5,5), strides=(1,1), 
activation='relu', input_shape=(64,64,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(3,3), 
activation='relu', padding="valid"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(3,3), 
activation='relu', padding="valid"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
```

```
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=192, kernel_size=(3,3), strides=(1,1),
activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(5, activation='softmax')
])

model_2b.compile(loss='categorical_crossentropy',optimizer=tf.optimize
rs.Adam(lr=0.01),metrics=['accuracy'])
model_2b.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 60, 60, 64)        4864

 batch_normalization_5 (Batc  (None, 60, 60, 64)       256
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 29, 29, 64)       0
 2D)

 conv2d_11 (Conv2D)          (None, 9, 9, 256)         147712

 batch_normalization_6 (Batc  (None, 9, 9, 256)        1024
 hNormalization)

 conv2d_12 (Conv2D)          (None, 3, 3, 384)         885120

 batch_normalization_7 (Batc  (None, 3, 3, 384)        1536
 hNormalization)

 conv2d_13 (Conv2D)          (None, 3, 3, 384)         1327488

 batch_normalization_8 (Batc  (None, 3, 3, 384)        1536
 hNormalization)

 conv2d_14 (Conv2D)          (None, 3, 3, 192)         663744

 batch_normalization_9 (Batc  (None, 3, 3, 192)        768
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 192)        0
```

```
 2D)

 flatten_2 (Flatten)         (None, 192)               0

 dense_6 (Dense)             (None, 4096)              790528

 dropout_2 (Dropout)         (None, 4096)              0

 dense_7 (Dense)             (None, 4096)              16781312

 dropout_3 (Dropout)         (None, 4096)              0

 dense_8 (Dense)             (None, 5)                 20485

=================================================================
Total params: 20,626,373
Trainable params: 20,623,813
Non-trainable params: 2,560
_________________________________________________________________
```

```
result2=model_2b.fit(train, epochs=Epochs,
                     validation_data=val,
                     validation_freq=1,callbacks=[tensorboard_cb])
```

```
Epoch 1/100
7/7 [==============================] - 8s 437ms/step - loss: 5.1274 -
accuracy: 0.2296 - val_loss: 1.8445 - val_accuracy: 0.2083
Epoch 2/100
7/7 [==============================] - 3s 448ms/step - loss: 2.3877 -
accuracy: 0.2500 - val_loss: 2.7075 - val_accuracy: 0.2083
Epoch 3/100
7/7 [==============================] - 3s 429ms/step - loss: 1.8662 -
accuracy: 0.3265 - val_loss: 2.0325 - val_accuracy: 0.2083
Epoch 4/100
7/7 [==============================] - 3s 437ms/step - loss: 1.8300 -
accuracy: 0.3469 - val_loss: 1.7785 - val_accuracy: 0.2083
Epoch 5/100
7/7 [==============================] - 3s 468ms/step - loss: 1.6497 -
accuracy: 0.3163 - val_loss: 1.6043 - val_accuracy: 0.1667
Epoch 6/100
7/7 [==============================] - 3s 386ms/step - loss: 1.8672 -
accuracy: 0.2551 - val_loss: 1.6088 - val_accuracy: 0.2500
Epoch 7/100
7/7 [==============================] - 3s 445ms/step - loss: 1.5813 -
accuracy: 0.3827 - val_loss: 1.6098 - val_accuracy: 0.1667
Epoch 8/100
7/7 [==============================] - 3s 453ms/step - loss: 1.5138 -
accuracy: 0.3571 - val_loss: 1.6024 - val_accuracy: 0.1667
Epoch 9/100
7/7 [==============================] - 3s 434ms/step - loss: 1.4789 -
accuracy: 0.3316 - val_loss: 1.6283 - val_accuracy: 0.2083
```

```
Epoch 10/100
7/7 [==============================] - 3s 479ms/step - loss: 1.4634 -
accuracy: 0.3827 - val_loss: 1.6385 - val_accuracy: 0.1250
Epoch 11/100
7/7 [==============================] - 3s 479ms/step - loss: 1.4241 -
accuracy: 0.3776 - val_loss: 1.5475 - val_accuracy: 0.2500
Epoch 12/100
7/7 [==============================] - 3s 432ms/step - loss: 1.4102 -
accuracy: 0.3878 - val_loss: 1.6588 - val_accuracy: 0.2083
Epoch 13/100
7/7 [==============================] - 3s 459ms/step - loss: 1.3964 -
accuracy: 0.3724 - val_loss: 1.8706 - val_accuracy: 0.2083
Epoch 14/100
7/7 [==============================] - 3s 426ms/step - loss: 1.3880 -
accuracy: 0.3827 - val_loss: 1.5961 - val_accuracy: 0.2083
Epoch 15/100
7/7 [==============================] - 4s 526ms/step - loss: 1.3918 -
accuracy: 0.4031 - val_loss: 1.7306 - val_accuracy: 0.2083
Epoch 16/100
7/7 [==============================] - 3s 460ms/step - loss: 1.5355 -
accuracy: 0.3724 - val_loss: 1.6376 - val_accuracy: 0.2917
Epoch 17/100
7/7 [==============================] - 3s 446ms/step - loss: 1.5234 -
accuracy: 0.3673 - val_loss: 1.8174 - val_accuracy: 0.2083
Epoch 18/100
7/7 [==============================] - 3s 425ms/step - loss: 1.3964 -
accuracy: 0.3929 - val_loss: 1.6060 - val_accuracy: 0.2500
Epoch 19/100
7/7 [==============================] - 3s 414ms/step - loss: 1.3375 -
accuracy: 0.4337 - val_loss: 1.5404 - val_accuracy: 0.3333
Epoch 20/100
7/7 [==============================] - 3s 449ms/step - loss: 1.5788 -
accuracy: 0.3112 - val_loss: 1.4647 - val_accuracy: 0.3333
Epoch 21/100
7/7 [==============================] - 3s 436ms/step - loss: 1.3879 -
accuracy: 0.4388 - val_loss: 1.6271 - val_accuracy: 0.2917
Epoch 22/100
7/7 [==============================] - 3s 442ms/step - loss: 1.4471 -
accuracy: 0.3980 - val_loss: 1.6040 - val_accuracy: 0.2500
Epoch 23/100
7/7 [==============================] - 3s 480ms/step - loss: 1.2645 -
accuracy: 0.4235 - val_loss: 1.6317 - val_accuracy: 0.2500
Epoch 24/100
7/7 [==============================] - 3s 403ms/step - loss: 1.4935 -
accuracy: 0.3724 - val_loss: 1.6108 - val_accuracy: 0.3333
Epoch 25/100
7/7 [==============================] - 3s 461ms/step - loss: 1.3913 -
accuracy: 0.3980 - val_loss: 1.6504 - val_accuracy: 0.3333
Epoch 26/100
7/7 [==============================] - 3s 460ms/step - loss: 1.4179 -
```

```
accuracy: 0.4082 - val_loss: 1.7139 - val_accuracy: 0.2500
Epoch 27/100
7/7 [==============================] - 3s 394ms/step - loss: 1.5094 -
accuracy: 0.3673 - val_loss: 1.8314 - val_accuracy: 0.2083
Epoch 28/100
7/7 [==============================] - 3s 407ms/step - loss: 1.3847 -
accuracy: 0.4184 - val_loss: 1.6993 - val_accuracy: 0.2917
Epoch 29/100
7/7 [==============================] - 3s 431ms/step - loss: 1.3148 -
accuracy: 0.4643 - val_loss: 1.8925 - val_accuracy: 0.2083
Epoch 30/100
7/7 [==============================] - 3s 429ms/step - loss: 1.3684 -
accuracy: 0.4133 - val_loss: 1.8693 - val_accuracy: 0.2083
Epoch 31/100
7/7 [==============================] - 3s 493ms/step - loss: 1.3417 -
accuracy: 0.4337 - val_loss: 1.6490 - val_accuracy: 0.2917
Epoch 32/100
7/7 [==============================] - 3s 453ms/step - loss: 1.3141 -
accuracy: 0.4439 - val_loss: 1.6034 - val_accuracy: 0.3333
Epoch 33/100
7/7 [==============================] - 3s 466ms/step - loss: 1.3356 -
accuracy: 0.4541 - val_loss: 1.6320 - val_accuracy: 0.4167
Epoch 34/100
7/7 [==============================] - 3s 470ms/step - loss: 1.3401 -
accuracy: 0.3980 - val_loss: 1.5715 - val_accuracy: 0.3333
Epoch 35/100
7/7 [==============================] - 3s 466ms/step - loss: 1.3523 -
accuracy: 0.4541 - val_loss: 1.7345 - val_accuracy: 0.3333
Epoch 36/100
7/7 [==============================] - 3s 391ms/step - loss: 1.3265 -
accuracy: 0.4337 - val_loss: 1.9772 - val_accuracy: 0.3750
Epoch 37/100
7/7 [==============================] - 3s 521ms/step - loss: 1.2940 -
accuracy: 0.5000 - val_loss: 1.7432 - val_accuracy: 0.4167
Epoch 38/100
7/7 [==============================] - 3s 460ms/step - loss: 1.3419 -
accuracy: 0.4592 - val_loss: 1.3999 - val_accuracy: 0.3750
Epoch 39/100
7/7 [==============================] - 3s 504ms/step - loss: 1.2833 -
accuracy: 0.4541 - val_loss: 1.7617 - val_accuracy: 0.2917
Epoch 40/100
7/7 [==============================] - 3s 482ms/step - loss: 1.2394 -
accuracy: 0.4898 - val_loss: 1.5182 - val_accuracy: 0.3333
Epoch 41/100
7/7 [==============================] - 3s 419ms/step - loss: 1.1745 -
accuracy: 0.5255 - val_loss: 1.9352 - val_accuracy: 0.2917
Epoch 42/100
7/7 [==============================] - 3s 460ms/step - loss: 1.1761 -
accuracy: 0.5306 - val_loss: 1.6146 - val_accuracy: 0.2917
Epoch 43/100
```

```
7/7 [==============================] - 3s 458ms/step - loss: 1.1495 -
accuracy: 0.5459 - val_loss: 1.4464 - val_accuracy: 0.2917
Epoch 44/100
7/7 [==============================] - 3s 450ms/step - loss: 1.2530 -
accuracy: 0.5153 - val_loss: 1.6227 - val_accuracy: 0.4583
Epoch 45/100
7/7 [==============================] - 3s 501ms/step - loss: 1.3166 -
accuracy: 0.4490 - val_loss: 1.8359 - val_accuracy: 0.2500
Epoch 46/100
7/7 [==============================] - 3s 455ms/step - loss: 1.2492 -
accuracy: 0.5102 - val_loss: 1.6095 - val_accuracy: 0.3333
Epoch 47/100
7/7 [==============================] - 3s 452ms/step - loss: 1.2796 -
accuracy: 0.4745 - val_loss: 1.4192 - val_accuracy: 0.2917
Epoch 48/100
7/7 [==============================] - 3s 472ms/step - loss: 1.2502 -
accuracy: 0.5000 - val_loss: 1.3390 - val_accuracy: 0.3750
Epoch 49/100
7/7 [==============================] - 3s 478ms/step - loss: 1.2360 -
accuracy: 0.4847 - val_loss: 1.2829 - val_accuracy: 0.4167
Epoch 50/100
7/7 [==============================] - 3s 432ms/step - loss: 1.3229 -
accuracy: 0.4796 - val_loss: 2.7469 - val_accuracy: 0.1667
Epoch 51/100
7/7 [==============================] - 3s 454ms/step - loss: 1.3553 -
accuracy: 0.5000 - val_loss: 2.0301 - val_accuracy: 0.2083
Epoch 52/100
7/7 [==============================] - 3s 477ms/step - loss: 1.2934 -
accuracy: 0.5153 - val_loss: 1.8983 - val_accuracy: 0.2500
Epoch 53/100
7/7 [==============================] - 3s 492ms/step - loss: 1.2089 -
accuracy: 0.4643 - val_loss: 2.6295 - val_accuracy: 0.2500
Epoch 54/100
7/7 [==============================] - 3s 438ms/step - loss: 1.2021 -
accuracy: 0.5255 - val_loss: 2.5020 - val_accuracy: 0.2917
Epoch 55/100
7/7 [==============================] - 3s 524ms/step - loss: 1.1939 -
accuracy: 0.5051 - val_loss: 1.7516 - val_accuracy: 0.3333
Epoch 56/100
7/7 [==============================] - 3s 446ms/step - loss: 1.0821 -
accuracy: 0.5969 - val_loss: 2.1856 - val_accuracy: 0.2917
Epoch 57/100
7/7 [==============================] - 3s 422ms/step - loss: 1.1088 -
accuracy: 0.5153 - val_loss: 1.4743 - val_accuracy: 0.4167
Epoch 58/100
7/7 [==============================] - 3s 442ms/step - loss: 1.1814 -
accuracy: 0.5153 - val_loss: 1.6262 - val_accuracy: 0.2917
Epoch 59/100
7/7 [==============================] - 3s 391ms/step - loss: 1.1440 -
accuracy: 0.5357 - val_loss: 1.4231 - val_accuracy: 0.3750
```

```
Epoch 60/100
7/7 [==============================] - 3s 484ms/step - loss: 1.2198 -
accuracy: 0.5051 - val_loss: 1.8097 - val_accuracy: 0.3750
Epoch 61/100
7/7 [==============================] - 3s 416ms/step - loss: 1.1487 -
accuracy: 0.5969 - val_loss: 1.9682 - val_accuracy: 0.4583
Epoch 62/100
7/7 [==============================] - 3s 422ms/step - loss: 1.1670 -
accuracy: 0.5459 - val_loss: 1.8770 - val_accuracy: 0.3750
Epoch 63/100
7/7 [==============================] - 3s 424ms/step - loss: 1.1937 -
accuracy: 0.5714 - val_loss: 1.6876 - val_accuracy: 0.4167
Epoch 64/100
7/7 [==============================] - 3s 445ms/step - loss: 1.2245 -
accuracy: 0.5204 - val_loss: 1.6421 - val_accuracy: 0.4167
Epoch 65/100
7/7 [==============================] - 4s 517ms/step - loss: 1.0239 -
accuracy: 0.6071 - val_loss: 1.5170 - val_accuracy: 0.4583
Epoch 66/100
7/7 [==============================] - 3s 493ms/step - loss: 1.1926 -
accuracy: 0.5357 - val_loss: 2.5031 - val_accuracy: 0.3750
Epoch 67/100
7/7 [==============================] - 3s 493ms/step - loss: 1.0900 -
accuracy: 0.5459 - val_loss: 1.5830 - val_accuracy: 0.4583
Epoch 68/100
7/7 [==============================] - 3s 424ms/step - loss: 1.0966 -
accuracy: 0.5663 - val_loss: 2.8172 - val_accuracy: 0.3333
Epoch 69/100
7/7 [==============================] - 3s 393ms/step - loss: 1.0685 -
accuracy: 0.5612 - val_loss: 1.8394 - val_accuracy: 0.3750
Epoch 70/100
7/7 [==============================] - 3s 512ms/step - loss: 1.1355 -
accuracy: 0.5408 - val_loss: 1.6480 - val_accuracy: 0.3750
Epoch 71/100
7/7 [==============================] - 3s 412ms/step - loss: 1.1389 -
accuracy: 0.5306 - val_loss: 2.8762 - val_accuracy: 0.1667
Epoch 72/100
7/7 [==============================] - 3s 429ms/step - loss: 1.1646 -
accuracy: 0.5153 - val_loss: 2.2559 - val_accuracy: 0.3750
Epoch 73/100
7/7 [==============================] - 3s 435ms/step - loss: 1.0913 -
accuracy: 0.5255 - val_loss: 1.9107 - val_accuracy: 0.3333
Epoch 74/100
7/7 [==============================] - 3s 431ms/step - loss: 1.0259 -
accuracy: 0.6071 - val_loss: 2.2086 - val_accuracy: 0.4167
Epoch 75/100
7/7 [==============================] - 4s 528ms/step - loss: 0.9945 -
accuracy: 0.6429 - val_loss: 2.4363 - val_accuracy: 0.2500
Epoch 76/100
7/7 [==============================] - 3s 433ms/step - loss: 1.0244 -
```

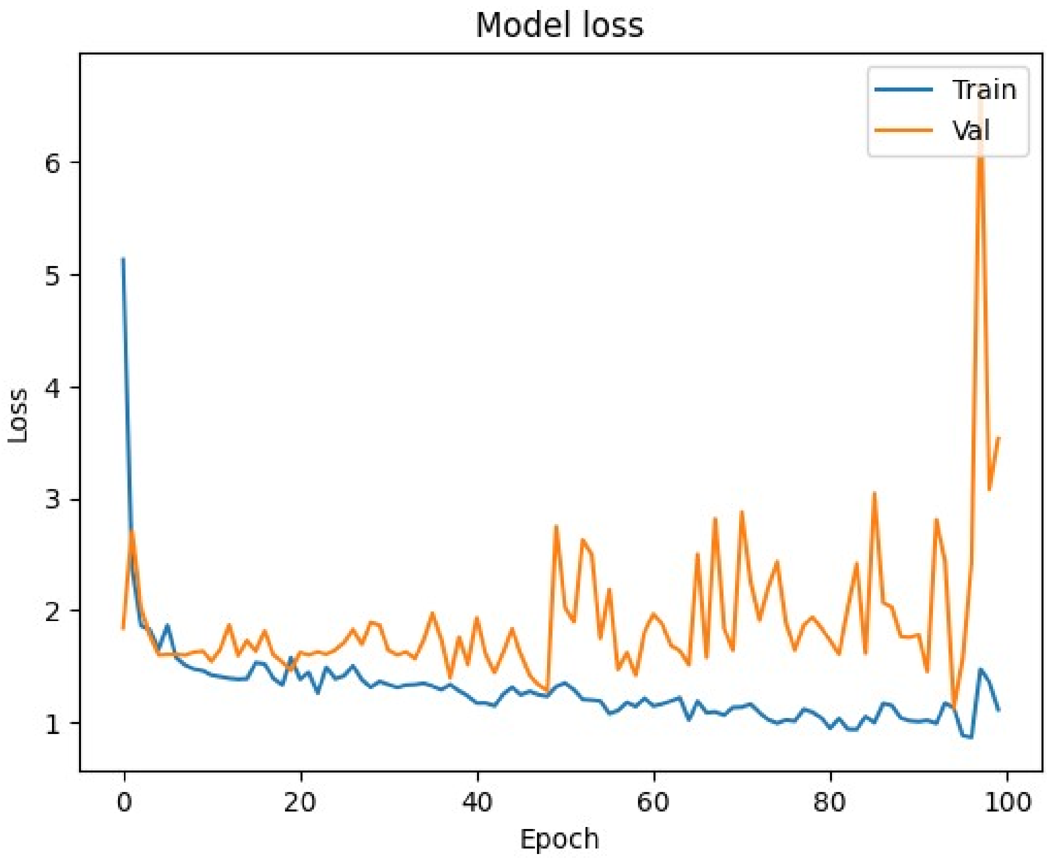
```
accuracy: 0.5867 - val_loss: 1.8853 - val_accuracy: 0.3750
Epoch 77/100
7/7 [==============================] - 3s 425ms/step - loss: 1.0133 -
accuracy: 0.5867 - val_loss: 1.6491 - val_accuracy: 0.3750
Epoch 78/100
7/7 [==============================] - 3s 446ms/step - loss: 1.1196 -
accuracy: 0.5867 - val_loss: 1.8680 - val_accuracy: 0.4167
Epoch 79/100
7/7 [==============================] - 3s 405ms/step - loss: 1.0950 -
accuracy: 0.5612 - val_loss: 1.9402 - val_accuracy: 0.3333
Epoch 80/100
7/7 [==============================] - 3s 473ms/step - loss: 1.0412 -
accuracy: 0.5918 - val_loss: 1.8420 - val_accuracy: 0.3750
Epoch 81/100
7/7 [==============================] - 3s 469ms/step - loss: 0.9513 -
accuracy: 0.6378 - val_loss: 1.7320 - val_accuracy: 0.3750
Epoch 82/100
7/7 [==============================] - 3s 484ms/step - loss: 1.0382 -
accuracy: 0.5714 - val_loss: 1.6094 - val_accuracy: 0.4167
Epoch 83/100
7/7 [==============================] - 3s 413ms/step - loss: 0.9432 -
accuracy: 0.6173 - val_loss: 2.0237 - val_accuracy: 0.3333
Epoch 84/100
7/7 [==============================] - 3s 407ms/step - loss: 0.9410 -
accuracy: 0.6429 - val_loss: 2.4180 - val_accuracy: 0.4167
Epoch 85/100
7/7 [==============================] - 4s 541ms/step - loss: 1.0553 -
accuracy: 0.6276 - val_loss: 1.6183 - val_accuracy: 0.2917
Epoch 86/100
7/7 [==============================] - 3s 453ms/step - loss: 0.9983 -
accuracy: 0.6020 - val_loss: 3.0433 - val_accuracy: 0.3750
Epoch 87/100
7/7 [==============================] - 3s 509ms/step - loss: 1.1714 -
accuracy: 0.5306 - val_loss: 2.0678 - val_accuracy: 0.3333
Epoch 88/100
7/7 [==============================] - 3s 432ms/step - loss: 1.1526 -
accuracy: 0.5408 - val_loss: 2.0278 - val_accuracy: 0.2083
Epoch 89/100
7/7 [==============================] - 3s 415ms/step - loss: 1.0411 -
accuracy: 0.5918 - val_loss: 1.7669 - val_accuracy: 0.2917
Epoch 90/100
7/7 [==============================] - 3s 412ms/step - loss: 1.0164 -
accuracy: 0.6122 - val_loss: 1.7608 - val_accuracy: 0.4583
Epoch 91/100
7/7 [==============================] - 3s 438ms/step - loss: 1.0093 -
accuracy: 0.6020 - val_loss: 1.7836 - val_accuracy: 0.2917
Epoch 92/100
7/7 [==============================] - 3s 429ms/step - loss: 1.0205 -
accuracy: 0.6173 - val_loss: 1.4570 - val_accuracy: 0.5417
Epoch 93/100
```

```
7/7 [==============================] - 3s 405ms/step - loss: 0.9934 -
accuracy: 0.6378 - val_loss: 2.8104 - val_accuracy: 0.2917
Epoch 94/100
7/7 [==============================] - 3s 480ms/step - loss: 1.1746 -
accuracy: 0.5867 - val_loss: 2.4410 - val_accuracy: 0.2917
Epoch 95/100
7/7 [==============================] - 4s 570ms/step - loss: 1.1296 -
accuracy: 0.5255 - val_loss: 1.1270 - val_accuracy: 0.5417
Epoch 96/100
7/7 [==============================] - 3s 436ms/step - loss: 0.8858 -
accuracy: 0.6684 - val_loss: 1.5734 - val_accuracy: 0.5000
Epoch 97/100
7/7 [==============================] - 3s 399ms/step - loss: 0.8685 -
accuracy: 0.6531 - val_loss: 2.4286 - val_accuracy: 0.4583
Epoch 98/100
7/7 [==============================] - 3s 421ms/step - loss: 1.4776 -
accuracy: 0.5612 - val_loss: 6.6788 - val_accuracy: 0.2083
Epoch 99/100
7/7 [==============================] - 3s 441ms/step - loss: 1.3637 -
accuracy: 0.4439 - val_loss: 3.0767 - val_accuracy: 0.2500
Epoch 100/100
7/7 [==============================] - 3s 481ms/step - loss: 1.1173 -
accuracy: 0.5357 - val_loss: 3.5332 - val_accuracy: 0.2500
```

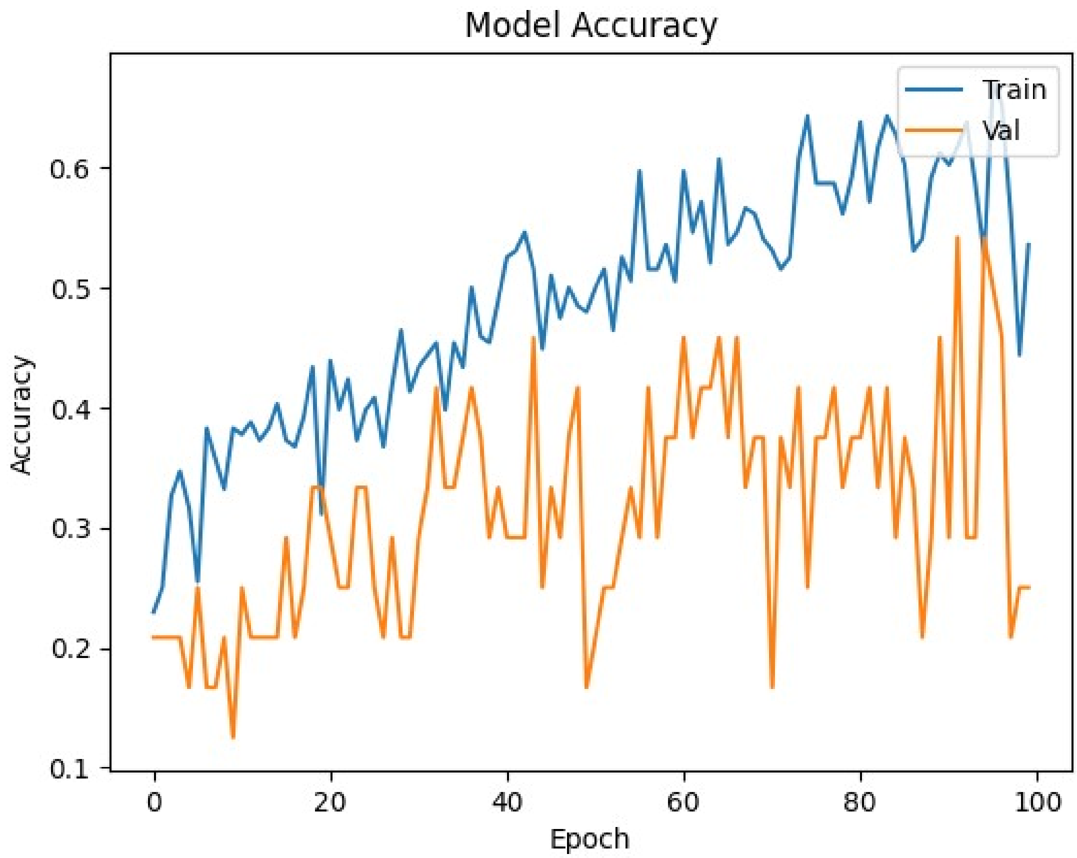
```python
loss2,acc2=model_2b.evaluate(test)
print("Accuracy of CNN Model: ",acc2)
print("Loss of CNN Model: ",loss2)
```

```
1/1 [==============================] - 0s 299ms/step - loss: 4.4970 -
accuracy: 0.2308
Accuracy of CNN Model:  0.23076923191547394
Loss of CNN Model:  4.49703311920166
```

```python
# plotting model loss
plt.plot(result2.history['loss'])
plt.plot(result2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```




```
# plotting accuracy
plt.plot(result2.history['accuracy'])
plt.plot(result2.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Dari hasil yang didapatkan plot model loss sudah memperlihatkan plot yang baik pada train, tetapi validasi masih terlihat tidak baik dan masih jauh dari train. Selain itu untuk akurasi model ini adalah 63% hampir sama dengan yang sebelumnya. Untuk validasi akurasinya 37% dan untuk test 38% lebih stabil daripada sebelumnya.

Untuk model selanjutnya akan mencoba mengganti optimizer dari SGD menjadi adam untuk melihat akurasi yang dihasilkan.

```
from keras import regularizers

model_2c = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=(5,5), strides=(1,1),
activation='relu', input_shape=(64,64,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3),strides=(1,1)),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(2,2),
activation='relu', padding="valid"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2,2), strides=(1,1)),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), strides=(1,1),
activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=64, kernel_size=(3,3), strides=(1,1),
```

```
activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2,2), strides=(2,2)),
    keras.layers.Flatten(),

keras.layers.Dense(1200,kernel_regularizer=regularizers.l2(0.01),activ
ation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(5, activation='softmax')
])

model_2c.compile(loss='categorical_crossentropy',optimizer=tf.optimize
rs.Adam(learning_rate=0.001),metrics=['accuracy'])
model_2c.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_15 (Conv2D)          (None, 60, 60, 64)        4864      
                                                                 
 batch_normalization_10 (Bat  (None, 60, 60, 64)       256       
 chNormalization)                                                
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 58, 58, 64)       0         
 2D)                                                             
                                                                 
 conv2d_16 (Conv2D)          (None, 28, 28, 256)       147712    
                                                                 
 batch_normalization_11 (Bat  (None, 28, 28, 256)      1024      
 chNormalization)                                                
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 27, 27, 256)      0         
 2D)                                                             
                                                                 
 conv2d_17 (Conv2D)          (None, 25, 25, 128)       295040    
                                                                 
 batch_normalization_12 (Bat  (None, 25, 25, 128)      512       
 chNormalization)                                                
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 12, 12, 128)      0         
 2D)                                                             
                                                                 
 conv2d_18 (Conv2D)          (None, 10, 10, 64)        73792     
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             
                                                                 
 flatten_3 (Flatten)         (None, 1600)              0         
                                                                 
 dense_9 (Dense)             (None, 1200)              1921200   
```

```
 dropout_4 (Dropout)         (None, 1200)              0

 dense_10 (Dense)            (None, 5)                 6005

=================================================================
Total params: 2,450,405
Trainable params: 2,449,509
Non-trainable params: 896
_________________________________________________________________
```

```
result3=model_2c.fit(train, epochs=Epochs,
                     validation_data=val,batch_size=15,
                     validation_freq=1,callbacks=[tensorboard_cb])
```

```
Epoch 1/100
7/7 [==============================] - 7s 558ms/step - loss: 17.8969 -
accuracy: 0.2551 - val_loss: 14.3037 - val_accuracy: 0.1667
Epoch 2/100
7/7 [==============================] - 3s 443ms/step - loss: 14.5455 -
accuracy: 0.2908 - val_loss: 13.3002 - val_accuracy: 0.2083
Epoch 3/100
7/7 [==============================] - 3s 470ms/step - loss: 13.4294 -
accuracy: 0.2551 - val_loss: 12.5856 - val_accuracy: 0.2500
Epoch 4/100
7/7 [==============================] - 3s 447ms/step - loss: 12.5595 -
accuracy: 0.3418 - val_loss: 12.4026 - val_accuracy: 0.1667
Epoch 5/100
7/7 [==============================] - 3s 489ms/step - loss: 11.6382 -
accuracy: 0.3214 - val_loss: 11.6556 - val_accuracy: 0.1250
Epoch 6/100
7/7 [==============================] - 4s 553ms/step - loss: 10.9300 -
accuracy: 0.3724 - val_loss: 11.7735 - val_accuracy: 0.2083
Epoch 7/100
7/7 [==============================] - 3s 412ms/step - loss: 10.3371 -
accuracy: 0.3980 - val_loss: 11.0958 - val_accuracy: 0.2083
Epoch 8/100
7/7 [==============================] - 3s 486ms/step - loss: 9.7617 -
accuracy: 0.3673 - val_loss: 11.1006 - val_accuracy: 0.2083
Epoch 9/100
7/7 [==============================] - 3s 475ms/step - loss: 9.2812 -
accuracy: 0.4337 - val_loss: 10.6397 - val_accuracy: 0.2083
Epoch 10/100
7/7 [==============================] - 3s 450ms/step - loss: 8.7499 -
accuracy: 0.4490 - val_loss: 9.8081 - val_accuracy: 0.2500
Epoch 11/100
7/7 [==============================] - 3s 441ms/step - loss: 8.3515 -
accuracy: 0.4337 - val_loss: 9.4234 - val_accuracy: 0.2083
Epoch 12/100
7/7 [==============================] - 3s 400ms/step - loss: 7.9648 -
accuracy: 0.4337 - val_loss: 9.0618 - val_accuracy: 0.1667
```

```
Epoch 13/100
7/7 [==============================] - 3s 432ms/step - loss: 7.6358 -
accuracy: 0.4541 - val_loss: 8.3606 - val_accuracy: 0.2083
Epoch 14/100
7/7 [==============================] - 3s 443ms/step - loss: 7.2157 -
accuracy: 0.4337 - val_loss: 9.0304 - val_accuracy: 0.2083
Epoch 15/100
7/7 [==============================] - 3s 428ms/step - loss: 6.9609 -
accuracy: 0.4490 - val_loss: 8.0112 - val_accuracy: 0.2083
Epoch 16/100
7/7 [==============================] - 3s 483ms/step - loss: 6.6345 -
accuracy: 0.4643 - val_loss: 7.4944 - val_accuracy: 0.1250
Epoch 17/100
7/7 [==============================] - 3s 450ms/step - loss: 6.4591 -
accuracy: 0.4949 - val_loss: 7.1976 - val_accuracy: 0.2917
Epoch 18/100
7/7 [==============================] - 3s 470ms/step - loss: 6.0218 -
accuracy: 0.4847 - val_loss: 7.2382 - val_accuracy: 0.2917
Epoch 19/100
7/7 [==============================] - 3s 416ms/step - loss: 5.7637 -
accuracy: 0.5000 - val_loss: 7.2932 - val_accuracy: 0.2083
Epoch 20/100
7/7 [==============================] - 3s 420ms/step - loss: 5.4749 -
accuracy: 0.5357 - val_loss: 6.8911 - val_accuracy: 0.2500
Epoch 21/100
7/7 [==============================] - 3s 443ms/step - loss: 5.3608 -
accuracy: 0.5102 - val_loss: 6.3243 - val_accuracy: 0.1250
Epoch 22/100
7/7 [==============================] - 3s 405ms/step - loss: 5.1429 -
accuracy: 0.4643 - val_loss: 6.2259 - val_accuracy: 0.2083
Epoch 23/100
7/7 [==============================] - 3s 450ms/step - loss: 4.9156 -
accuracy: 0.5255 - val_loss: 6.3406 - val_accuracy: 0.2917
Epoch 24/100
7/7 [==============================] - 3s 456ms/step - loss: 4.6912 -
accuracy: 0.5663 - val_loss: 7.0926 - val_accuracy: 0.2083
Epoch 25/100
7/7 [==============================] - 3s 461ms/step - loss: 4.6520 -
accuracy: 0.4796 - val_loss: 6.9928 - val_accuracy: 0.2083
Epoch 26/100
7/7 [==============================] - 3s 492ms/step - loss: 4.5172 -
accuracy: 0.4949 - val_loss: 5.2969 - val_accuracy: 0.2917
Epoch 27/100
7/7 [==============================] - 3s 413ms/step - loss: 4.3366 -
accuracy: 0.4898 - val_loss: 6.1383 - val_accuracy: 0.2083
Epoch 28/100
7/7 [==============================] - 3s 466ms/step - loss: 4.2011 -
accuracy: 0.5102 - val_loss: 5.0444 - val_accuracy: 0.2083
Epoch 29/100
7/7 [==============================] - 3s 422ms/step - loss: 4.2098 -
```

```
accuracy: 0.5000 - val_loss: 5.4864 - val_accuracy: 0.2083
Epoch 30/100
7/7 [==============================] - 3s 450ms/step - loss: 3.9172 -
accuracy: 0.4898 - val_loss: 5.1776 - val_accuracy: 0.1667
Epoch 31/100
7/7 [==============================] - 3s 530ms/step - loss: 3.7223 -
accuracy: 0.5714 - val_loss: 4.9585 - val_accuracy: 0.2500
Epoch 32/100
7/7 [==============================] - 3s 441ms/step - loss: 3.7293 -
accuracy: 0.5204 - val_loss: 4.8105 - val_accuracy: 0.1667
Epoch 33/100
7/7 [==============================] - 3s 491ms/step - loss: 3.6404 -
accuracy: 0.5255 - val_loss: 4.6013 - val_accuracy: 0.2917
Epoch 34/100
7/7 [==============================] - 3s 424ms/step - loss: 3.4857 -
accuracy: 0.5510 - val_loss: 4.5387 - val_accuracy: 0.2083
Epoch 35/100
7/7 [==============================] - 3s 385ms/step - loss: 3.3432 -
accuracy: 0.5459 - val_loss: 4.4457 - val_accuracy: 0.1667
Epoch 36/100
7/7 [==============================] - 3s 532ms/step - loss: 3.3637 -
accuracy: 0.5051 - val_loss: 4.2194 - val_accuracy: 0.1250
Epoch 37/100
7/7 [==============================] - 3s 388ms/step - loss: 3.1861 -
accuracy: 0.5663 - val_loss: 5.3766 - val_accuracy: 0.2083
Epoch 38/100
7/7 [==============================] - 3s 398ms/step - loss: 3.0182 -
accuracy: 0.5867 - val_loss: 3.9530 - val_accuracy: 0.2083
Epoch 39/100
7/7 [==============================] - 3s 437ms/step - loss: 3.0065 -
accuracy: 0.5357 - val_loss: 5.1310 - val_accuracy: 0.1667
Epoch 40/100
7/7 [==============================] - 3s 426ms/step - loss: 2.9163 -
accuracy: 0.5714 - val_loss: 4.2808 - val_accuracy: 0.2500
Epoch 41/100
7/7 [==============================] - 3s 499ms/step - loss: 2.8224 -
accuracy: 0.6122 - val_loss: 4.6882 - val_accuracy: 0.2083
Epoch 42/100
7/7 [==============================] - 3s 428ms/step - loss: 2.6419 -
accuracy: 0.6327 - val_loss: 4.9930 - val_accuracy: 0.1667
Epoch 43/100
7/7 [==============================] - 3s 417ms/step - loss: 2.5752 -
accuracy: 0.6327 - val_loss: 4.9526 - val_accuracy: 0.2917
Epoch 44/100
7/7 [==============================] - 3s 391ms/step - loss: 2.3101 -
accuracy: 0.6939 - val_loss: 4.9476 - val_accuracy: 0.2083
Epoch 45/100
7/7 [==============================] - 3s 445ms/step - loss: 2.3621 -
accuracy: 0.6480 - val_loss: 5.0756 - val_accuracy: 0.2083
Epoch 46/100
```

```
7/7 [==============================] - 3s 515ms/step - loss: 2.4080 -
accuracy: 0.6480 - val_loss: 5.9135 - val_accuracy: 0.2500
Epoch 47/100
7/7 [==============================] - 3s 415ms/step - loss: 2.2952 -
accuracy: 0.6735 - val_loss: 5.4682 - val_accuracy: 0.2917
Epoch 48/100
7/7 [==============================] - 3s 435ms/step - loss: 2.3301 -
accuracy: 0.6582 - val_loss: 5.6175 - val_accuracy: 0.1667
Epoch 49/100
7/7 [==============================] - 3s 422ms/step - loss: 2.3124 -
accuracy: 0.6531 - val_loss: 3.3354 - val_accuracy: 0.3750
Epoch 50/100
7/7 [==============================] - 3s 400ms/step - loss: 2.2514 -
accuracy: 0.6378 - val_loss: 3.3038 - val_accuracy: 0.3750
Epoch 51/100
7/7 [==============================] - 3s 434ms/step - loss: 2.1863 -
accuracy: 0.6327 - val_loss: 3.4802 - val_accuracy: 0.3750
Epoch 52/100
7/7 [==============================] - 3s 457ms/step - loss: 2.1282 -
accuracy: 0.6276 - val_loss: 3.3479 - val_accuracy: 0.3750
Epoch 53/100
7/7 [==============================] - 3s 448ms/step - loss: 2.2370 -
accuracy: 0.6684 - val_loss: 4.9376 - val_accuracy: 0.2917
Epoch 54/100
7/7 [==============================] - 3s 427ms/step - loss: 2.1879 -
accuracy: 0.6224 - val_loss: 3.0333 - val_accuracy: 0.2500
Epoch 55/100
7/7 [==============================] - 3s 420ms/step - loss: 2.1551 -
accuracy: 0.6071 - val_loss: 2.9943 - val_accuracy: 0.2500
Epoch 56/100
7/7 [==============================] - 3s 436ms/step - loss: 2.0467 -
accuracy: 0.6735 - val_loss: 4.0561 - val_accuracy: 0.1667
Epoch 57/100
7/7 [==============================] - 3s 498ms/step - loss: 1.8944 -
accuracy: 0.7194 - val_loss: 3.4695 - val_accuracy: 0.4167
Epoch 58/100
7/7 [==============================] - 3s 399ms/step - loss: 1.8754 -
accuracy: 0.6837 - val_loss: 3.2981 - val_accuracy: 0.4583
Epoch 59/100
7/7 [==============================] - 3s 514ms/step - loss: 1.8044 -
accuracy: 0.6990 - val_loss: 3.1477 - val_accuracy: 0.3750
Epoch 60/100
7/7 [==============================] - 3s 412ms/step - loss: 1.7396 -
accuracy: 0.7092 - val_loss: 3.5373 - val_accuracy: 0.2917
Epoch 61/100
7/7 [==============================] - 3s 451ms/step - loss: 1.8995 -
accuracy: 0.6786 - val_loss: 3.5051 - val_accuracy: 0.2917
Epoch 62/100
7/7 [==============================] - 3s 451ms/step - loss: 1.9022 -
accuracy: 0.6378 - val_loss: 4.5949 - val_accuracy: 0.2917
```

```
Epoch 63/100
7/7 [==============================] - 3s 491ms/step - loss: 1.9804 -
accuracy: 0.6582 - val_loss: 3.8927 - val_accuracy: 0.2917
Epoch 64/100
7/7 [==============================] - 3s 506ms/step - loss: 1.8738 -
accuracy: 0.7143 - val_loss: 3.5464 - val_accuracy: 0.3333
Epoch 65/100
7/7 [==============================] - 3s 423ms/step - loss: 1.7249 -
accuracy: 0.7296 - val_loss: 3.2559 - val_accuracy: 0.3333
Epoch 66/100
7/7 [==============================] - 3s 445ms/step - loss: 1.6638 -
accuracy: 0.7551 - val_loss: 3.5111 - val_accuracy: 0.4167
Epoch 67/100
7/7 [==============================] - 4s 523ms/step - loss: 1.7287 -
accuracy: 0.6786 - val_loss: 3.7455 - val_accuracy: 0.4583
Epoch 68/100
7/7 [==============================] - 3s 516ms/step - loss: 1.7303 -
accuracy: 0.7092 - val_loss: 3.3872 - val_accuracy: 0.3333
Epoch 69/100
7/7 [==============================] - 3s 454ms/step - loss: 1.5961 -
accuracy: 0.7449 - val_loss: 3.3272 - val_accuracy: 0.3750
Epoch 70/100
7/7 [==============================] - 3s 474ms/step - loss: 1.6399 -
accuracy: 0.7041 - val_loss: 3.2372 - val_accuracy: 0.2917
Epoch 71/100
7/7 [==============================] - 3s 445ms/step - loss: 1.6658 -
accuracy: 0.6990 - val_loss: 3.1712 - val_accuracy: 0.4167
Epoch 72/100
7/7 [==============================] - 3s 442ms/step - loss: 1.4922 -
accuracy: 0.7755 - val_loss: 3.3547 - val_accuracy: 0.3750
Epoch 73/100
7/7 [==============================] - 3s 465ms/step - loss: 1.7001 -
accuracy: 0.6888 - val_loss: 2.8332 - val_accuracy: 0.2500
Epoch 74/100
7/7 [==============================] - 3s 498ms/step - loss: 1.9013 -
accuracy: 0.6378 - val_loss: 3.1955 - val_accuracy: 0.3333
Epoch 75/100
7/7 [==============================] - 3s 483ms/step - loss: 1.7574 -
accuracy: 0.6378 - val_loss: 3.3500 - val_accuracy: 0.3750
Epoch 76/100
7/7 [==============================] - 3s 403ms/step - loss: 1.5772 -
accuracy: 0.7602 - val_loss: 3.5746 - val_accuracy: 0.2500
Epoch 77/100
7/7 [==============================] - 4s 504ms/step - loss: 1.5225 -
accuracy: 0.7041 - val_loss: 3.2022 - val_accuracy: 0.1667
Epoch 78/100
7/7 [==============================] - 3s 436ms/step - loss: 1.5142 -
accuracy: 0.7194 - val_loss: 4.1503 - val_accuracy: 0.3333
Epoch 79/100
7/7 [==============================] - 3s 381ms/step - loss: 1.5258 -
```

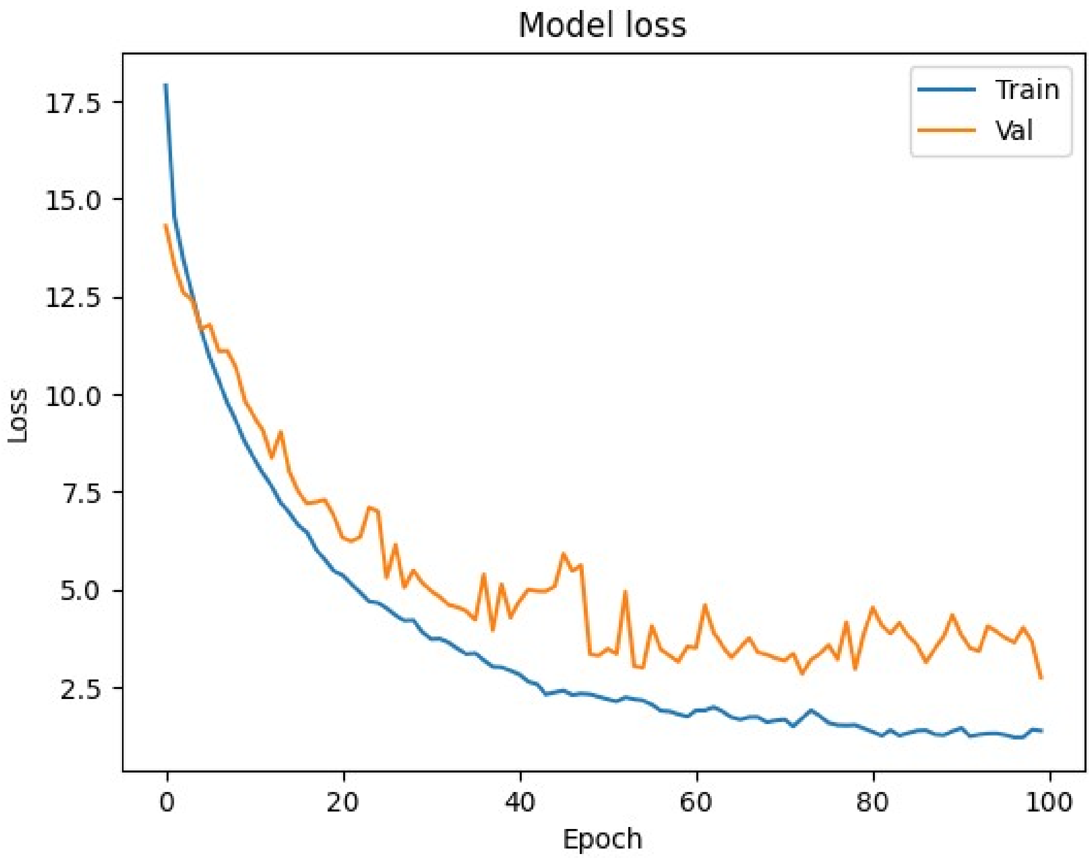
```
accuracy: 0.7194 - val_loss: 2.9558 - val_accuracy: 0.3750
Epoch 80/100
7/7 [==============================] - 3s 407ms/step - loss: 1.4362 -
accuracy: 0.7653 - val_loss: 3.8943 - val_accuracy: 0.2917
Epoch 81/100
7/7 [==============================] - 3s 447ms/step - loss: 1.3542 -
accuracy: 0.8061 - val_loss: 4.5284 - val_accuracy: 0.3750
Epoch 82/100
7/7 [==============================] - 3s 434ms/step - loss: 1.2500 -
accuracy: 0.8163 - val_loss: 4.0703 - val_accuracy: 0.4583
Epoch 83/100
7/7 [==============================] - 3s 419ms/step - loss: 1.3982 -
accuracy: 0.7398 - val_loss: 3.8730 - val_accuracy: 0.3333
Epoch 84/100
7/7 [==============================] - 3s 412ms/step - loss: 1.2510 -
accuracy: 0.8112 - val_loss: 4.1381 - val_accuracy: 0.3333
Epoch 85/100
7/7 [==============================] - 3s 441ms/step - loss: 1.3222 -
accuracy: 0.7959 - val_loss: 3.8048 - val_accuracy: 0.3333
Epoch 86/100
7/7 [==============================] - 3s 421ms/step - loss: 1.3842 -
accuracy: 0.7857 - val_loss: 3.5759 - val_accuracy: 0.2917
Epoch 87/100
7/7 [==============================] - 4s 527ms/step - loss: 1.3946 -
accuracy: 0.7449 - val_loss: 3.1368 - val_accuracy: 0.4167
Epoch 88/100
7/7 [==============================] - 3s 487ms/step - loss: 1.2870 -
accuracy: 0.7857 - val_loss: 3.4786 - val_accuracy: 0.3333
Epoch 89/100
7/7 [==============================] - 3s 440ms/step - loss: 1.2675 -
accuracy: 0.8010 - val_loss: 3.7933 - val_accuracy: 0.2500
Epoch 90/100
7/7 [==============================] - 3s 493ms/step - loss: 1.3677 -
accuracy: 0.7500 - val_loss: 4.3456 - val_accuracy: 0.4167
Epoch 91/100
7/7 [==============================] - 3s 508ms/step - loss: 1.4500 -
accuracy: 0.7092 - val_loss: 3.8227 - val_accuracy: 0.4167
Epoch 92/100
7/7 [==============================] - 3s 421ms/step - loss: 1.2369 -
accuracy: 0.8112 - val_loss: 3.4986 - val_accuracy: 0.3750
Epoch 93/100
7/7 [==============================] - 3s 464ms/step - loss: 1.2829 -
accuracy: 0.7755 - val_loss: 3.4220 - val_accuracy: 0.3333
Epoch 94/100
7/7 [==============================] - 3s 461ms/step - loss: 1.3082 -
accuracy: 0.7857 - val_loss: 4.0535 - val_accuracy: 0.3333
Epoch 95/100
7/7 [==============================] - 3s 443ms/step - loss: 1.3124 -
accuracy: 0.7908 - val_loss: 3.9184 - val_accuracy: 0.4167
Epoch 96/100
```

```
7/7 [==============================] - 3s 411ms/step - loss: 1.2724 -
accuracy: 0.7704 - val_loss: 3.7503 - val_accuracy: 0.2917
Epoch 97/100
7/7 [==============================] - 4s 588ms/step - loss: 1.2071 -
accuracy: 0.7551 - val_loss: 3.6363 - val_accuracy: 0.3333
Epoch 98/100
7/7 [==============================] - 3s 422ms/step - loss: 1.2117 -
accuracy: 0.8061 - val_loss: 4.0169 - val_accuracy: 0.4167
Epoch 99/100
7/7 [==============================] - 3s 419ms/step - loss: 1.4137 -
accuracy: 0.7194 - val_loss: 3.6533 - val_accuracy: 0.3333
Epoch 100/100
7/7 [==============================] - 3s 459ms/step - loss: 1.3839 -
accuracy: 0.7296 - val_loss: 2.7430 - val_accuracy: 0.3333
```

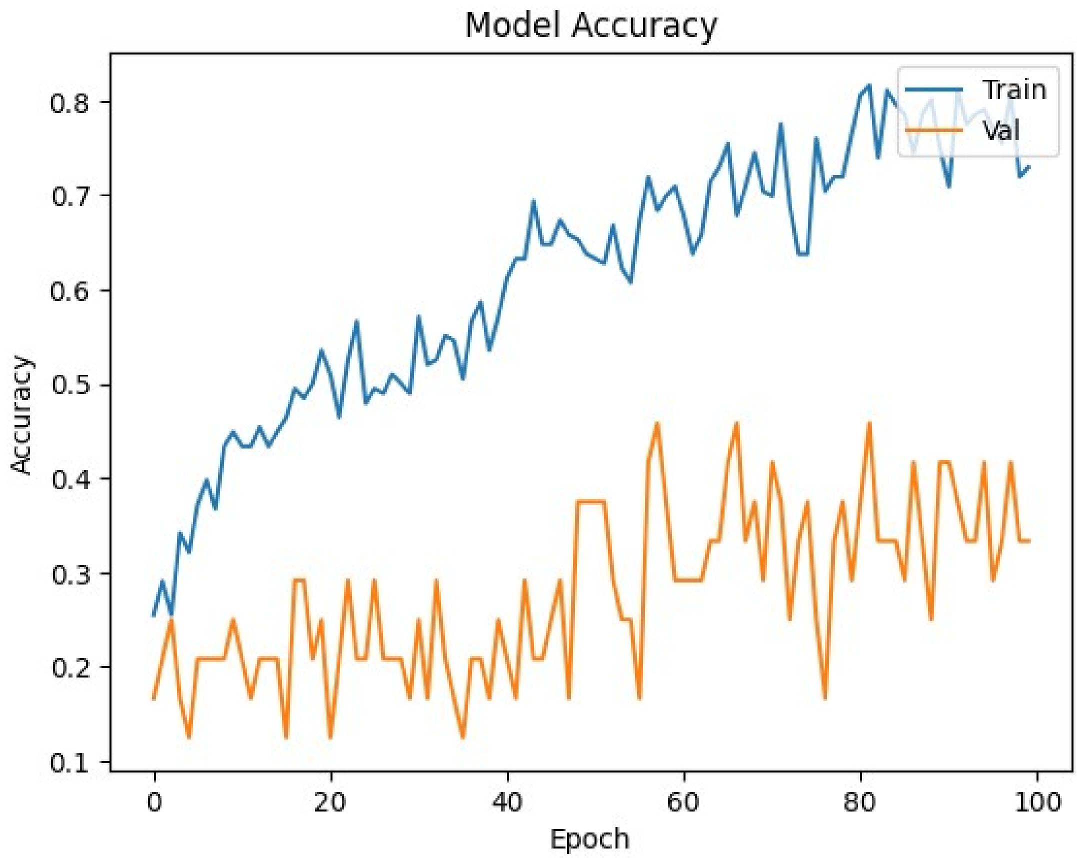
```python
loss3,acc3=model_2c.evaluate(test)
print("Accuracy of CNN Model: ",acc3)
print("Loss of CNN Model: ",loss3)
```

```
1/1 [==============================] - 0s 366ms/step - loss: 2.7406 -
accuracy: 0.3077
Accuracy of CNN Model:  0.3076923191547394
Loss of CNN Model:  2.7406129837036133
```

```python
# plotting model loss
plt.plot(result3.history['loss'])
plt.plot(result3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```




```python
# plotting accuracy
plt.plot(result3.history['accuracy'])
plt.plot(result3.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Model inilah yang menjadi final yang mana akan digunkanan dengan akurasi model final adalah 76%, Tetapi model tersebut memliki akurasi val 33% dan akutasi test adalah 34%. Hal ini dapat dikarenakan kekurangan data dari model yang membuat akurasi dari val dan test menurun. Tetapi plot model loss memiliki hasil yang baik.

Berikut penjelasan model yang digunakan :

- arskitektur

  Arsitektur akan diganti dari awalnya sedikit kompleks menjadi sederhana. Arsitektur yang digunakan sebgai berikut :

  ```
  input -> convulsion(k = 5,s = 1,f = 64) -> pool(k = 3,s = 1) ->
  convulison(k = 3,s = 2,f = 256) -> pool(k = 3,s = 1) ->
  convulison(k = 3,s = 1,f = 128) -> pool(k = 3,s = 2) ->
  convulison(k = 3,s = 1,f = 64) -> pool(k = 2,s = 2)
  ```

  Dengan penjelasan k adalah kernel size, s adalah stride, dan f adalah filter.

- activation function

  Pada arsitektur ini activation function di akhir layer akan ditambahkan dengan activation function softmax yang mana awalnya tidak terdapat activation function sama sekali.

- Batch Normalization

  Pada arsitektur ini batch normlization akan di apply di setiap convulsion layer. Hal ini dilakukan untuk menurukan overfit dan menabahkan kecepatan model.

- Hidden layer

  Pada model ini hidden layer akan diubah yang mana awalnya 2 menjadi satu. Hal ini dikarenakan model ini perlu di simplekan karena sudah overfit dengan kompleknya model base.

- Jumlah neuron

  Pada model awalnya neuron yang diberikan 4090 hal ini dapat menurunkan loss model tapi akurasi model akan juga berkurang. Sehingga untuk menaikan akurasi haruslah diturunkan neuron yang akan digunkan sehingga dengan menggunkana 1200 dapat menaikan akurasi model.

- Menambahakan drop out

  Pada hidden layer akan ditambahkan dropout untuk meingkat akurasi.

- Menambahkan regulasi

  Pada model ini juga aka ditambahkan regulasi yang mana dengan menambakan regulasi akan meingkatkan model loss karena regulasi akan menghilangkan beberapa layer setiap epoch agar mempercepat model. Tapi dengan menambahkan ini akan meningkatkan akurasi model.

- Menggunakan optimizer

  Pada model ini akan ditambahkan optimzer adam dengan leraning rate 0.01, optimizer akan membantu model meningkatkan akurasi dan juga kecepatannya dan juga learning rate yang rendah akan menghilangkan model spike di model lossnya.

2E. [LO 3, 5 poin] Evaluasi performa dari arsitektur nomor 2d dan jelaskan hasil yang kalian dapatkan. Gunakan testing set yang diberikan untuk memprediksi nilai ground truth dengan predicted result.

```
loss3,acc3=model_2c.evaluate(test)
print("Accuracy of CNN Model: ",acc3)
print("Loss of CNN Model: ",loss3)
```

```
1/1 [==============================] - 0s 271ms/step - loss: 2.7406 -
accuracy: 0.3077
Accuracy of CNN Model:  0.3076923191547394
Loss of CNN Model:  2.740612506866455
```

Setelah di bentuk beberapa model model 2c lah yang dapat memenuhi dengan akurasi antar test size 34%. Walaupun terlihat kecil, hal ini dikarenakan kekurangan data yang didapatkan. Tetapi untuk tess loss dapat 3.7.

Untuk melakukan meningkat performa dari model ini dapat dengan menambahkan dataset dan menggunakan gambar yang lebih jelas dengan bentuk dari batik yang digunakan.

2F. [LO 1, LO 2, LO 3, LO 4 5 poin] Buatlah video presentasi yang menjelaskan arsitektur yang dibangun untuk mengklasifikasikan batik ini.

Link Video : https://drive.google.com/file/d/1t83v8CT8d9lLnSYiENNXsj-q--JbEtnh/view?usp=sharing