AI Task 2 Report

Name: Abdelrahman Mohamed Sobhy

# First: Lets Test Accuracy and loss, we will change the size of image:-

## We will use same settings with change the size of the image

The split of data will be 0.7 for train and 0.3 for test and validation

model = models.Sequential()

model.add(layers.Conv2D(20, (3, 3), activation='relu', input\_shape=(100, 100, 3)))

model.add(layers.MaxPooling2D((2, 2)))

#model.add(Dropout(0.0))

model.add(layers.Conv2D(32, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

#model.add(Dropout(0.0))

model.add(layers.Conv2D(60, (3, 3) , activation='relu'))

#model.add(Dropout(0))

model.add(layers.Flatten())

model.add(layers.Dense(100, activation='relu'))

model.add(layers.Dense(102, activation='softmax'))

model summary

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_6 (Conv2D) (None, 98, 98, 20) 560

max\_pooling2d\_4 (MaxPoolin g2D) (None, 49, 49, 20) 0

conv2d\_7 (Conv2D) (None, 47, 47, 32) 5792

max\_pooling2d\_5 (MaxPoolin g2D) (None, 23, 23, 32) 0

conv2d\_8 (Conv2D) (None, 21, 21, 60) 17340

flatten\_2 (Flatten) (None, 26460) 0

dense\_4 (Dense) (None, 100) 2646100

dense\_5 (Dense) (None, 102) 10302

=================================================================

Total params: 2680094 (10.22 MB)

Trainable params: 2680094 (10.22 MB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

model.compile(optimizer=optimizers.Adam(learning\_rate=0.001),

              loss=tf.keras.losses.CategoricalCrossentropy(from\_logits=True),

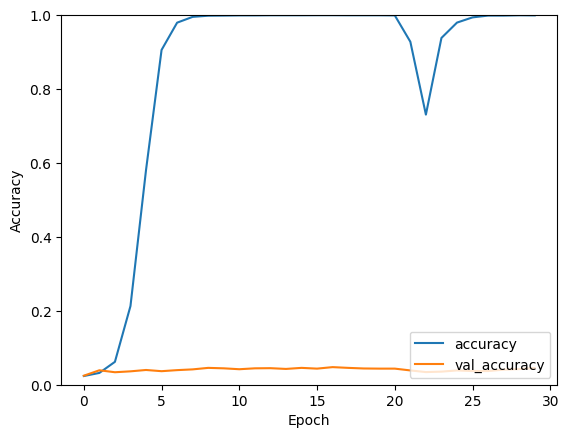
              metrics=['accuracy'])

history = model.fit(train\_images, train\_labels\_HOT, epochs=30,

                    validation\_data=(test\_images, test\_labels\_HOT),batch\_size=50)

1. image size: 100 \* 100 \* 3 “RGB”

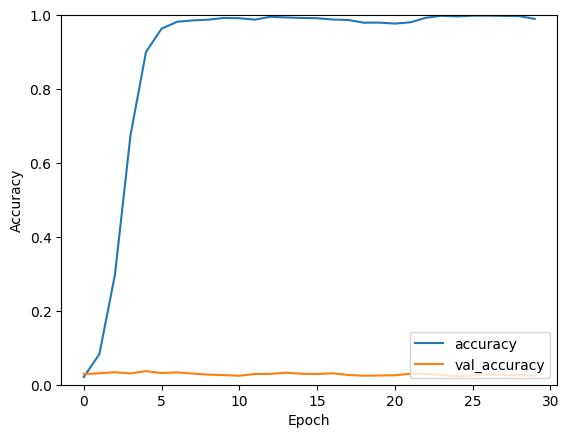
Graph for results :



77/77 - 0s - loss: 14.7025 - accuracy: 0.0452 - 310ms/epoch - 4ms/step

Total Accuarcy = 4.52%

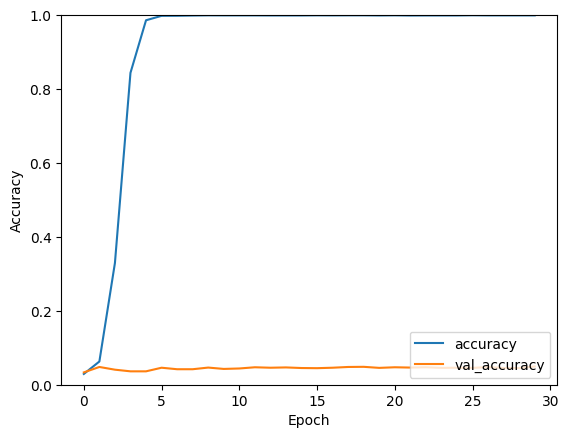
Image size: 150 \* 150 \* 3



77/77 - 1s - loss: 22.2264 - accuracy: 0.0252 - 651ms/epoch - 8ms/step

Total Accuracy: 2.52%

Image size: 200 \* 200 \* 3



77/77 - 1s - loss: 11.9203 - accuracy: 0.0492 - 753ms/epoch - 10ms/step

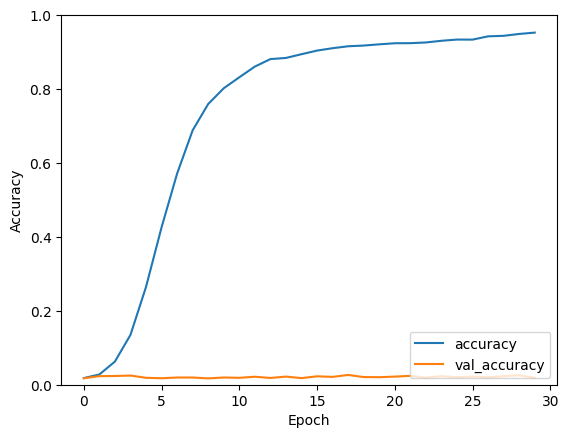
Total accuracy: 4.92%

The max size that I can test is 200 \* 200 \* 3 due to ram size

Also it has the highest accuracy

So we will make all next tests with 200 \* 200 \* 3 size

Let’s try the gray scale?

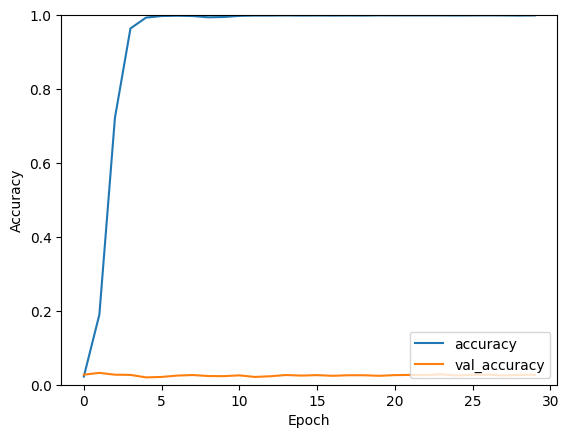


77/77 - 1s - loss: 19.5639 - accuracy: 0.0187 - 643ms/epoch - 8ms/step

Total accuracy: 1.87%

So we will continue with RGB images ☺

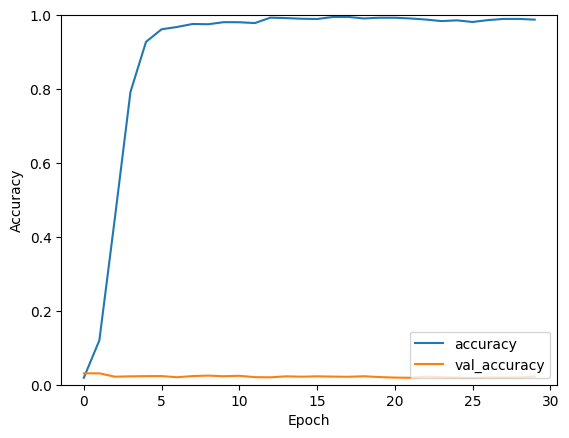
Let’s change the percentage of train? Maybe 60%?



103/103 - 1s - loss: 13.2514 - accuracy: 0.0269 - 851ms/epoch - 8ms/step

Total accuracy: 2.69%

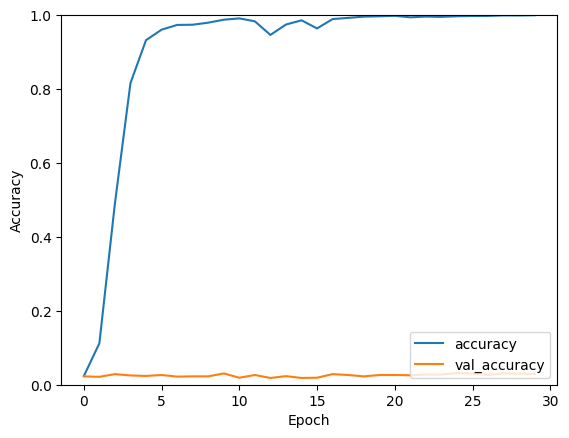
What about 50% train?



128/128 - 1s - loss: 30.3164 - accuracy: 0.0222 - 1s/epoch - 9ms/step

Total Accuracy: 2.22%

Let’s try 80%?



52/52 - 1s - loss: 26.5016 - accuracy: 0.0293 - 598ms/epoch - 11ms/step

Total Accuracy: 2.93%

So I think 70% was better, so we will use it

Let’s try other parameter

What about add new Conv2d layer, “more deep”?

model.add(layers.Conv2D(32, (3, 3), activation='relu'))

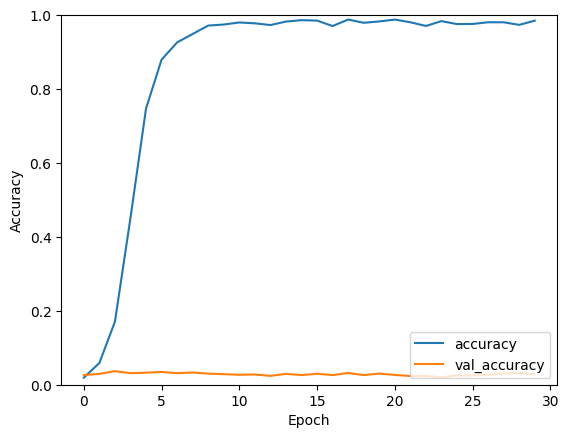
model.add(layers.MaxPooling2D((2, 2)))

#model.add(Dropout(0))

Add this layer

As third layer, so we have 4 conv2d layers

Results of 4 layers Conv2d:



77/77 - 1s - loss: 20.9577 - accuracy: 0.0281 - 796ms/epoch - 10ms/step

Total Accuracy: 2.81%

Let’s add other layer?

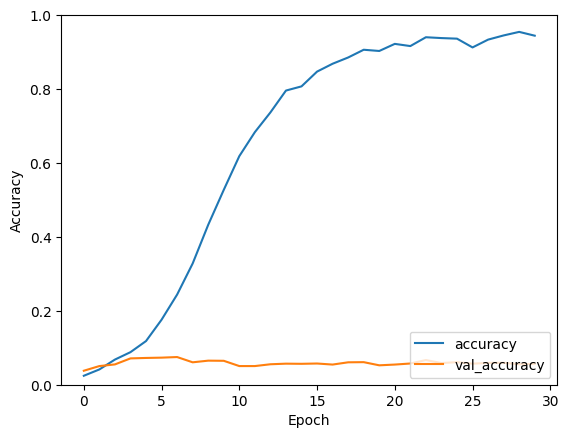
With same configuration

model.add(layers.Conv2D(32, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

#model.add(Dropout(0))

Results of 5 Conv2d Layers:



77/77 - 1s - loss: 18.9861 - accuracy: 0.0566 - 815ms/epoch - 11ms/step

Total Accuracy: 5.66%

More better? We have a bad problem of overfitting

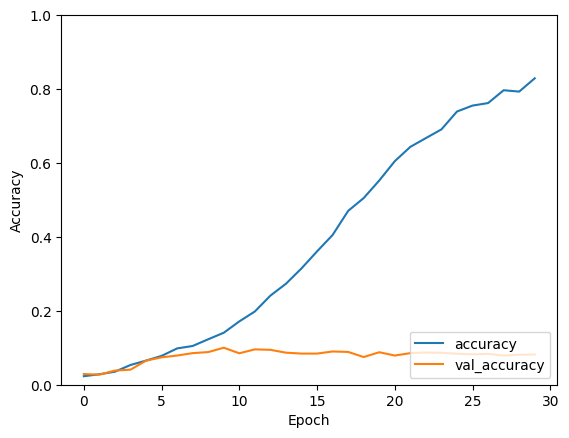
So we will try to fix it

May be using dropout?

We will make 10% dropout for all layers

With 5 conv2d layers

Dropout 10% results:

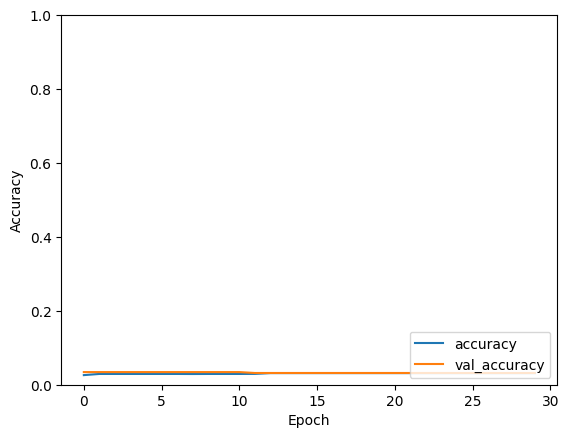


77/77 - 1s - loss: 7.6473 - accuracy: 0.0818 - 728ms/epoch - 9ms/step

Total Accuracy: 8.18%

What about 20% dropout?

Dropout 20% results:

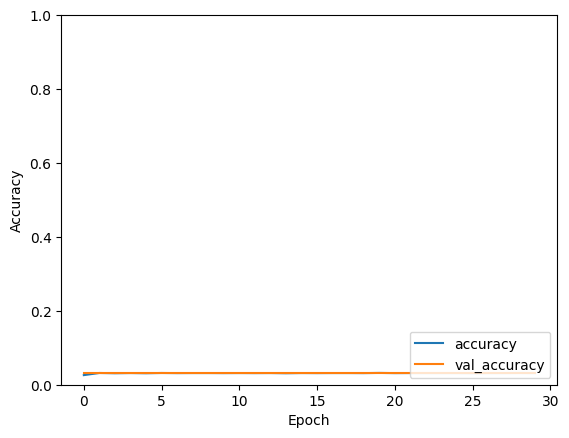


77/77 - 1s - loss: 4.5106 - accuracy: 0.0317 - 626ms/epoch - 8ms/step

Total Accuracy: 3.17%

What about 30% dropout?

Dropout 30% results:



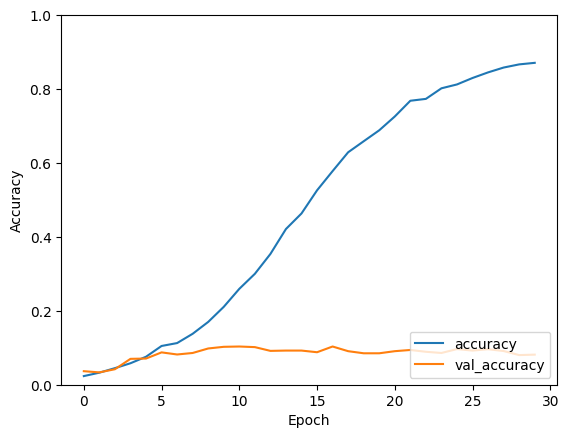
77/77 - 1s - loss: 4.5159 - accuracy: 0.0313 - 832ms/epoch - 11ms/step

Total accuracy: 3.13%

So we will use 10% drop out

And we will increase # of filters in first layer from 20 to 30

# of filters in layer 1 = 30 results:



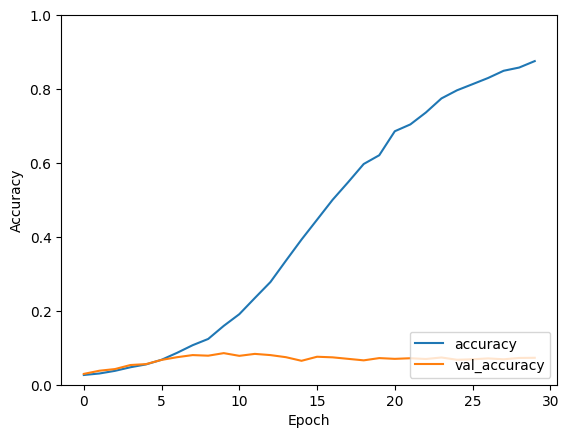
77/77 - 1s - loss: 8.7943 - accuracy: 0.0814 - 902ms/epoch - 12ms/step

Total Accuracy: 8.14%

Let’s increase all filters

30, 60, 60, 120, 240 respectively

30, 60, 60, 120, and 240 respectively # of filters results:

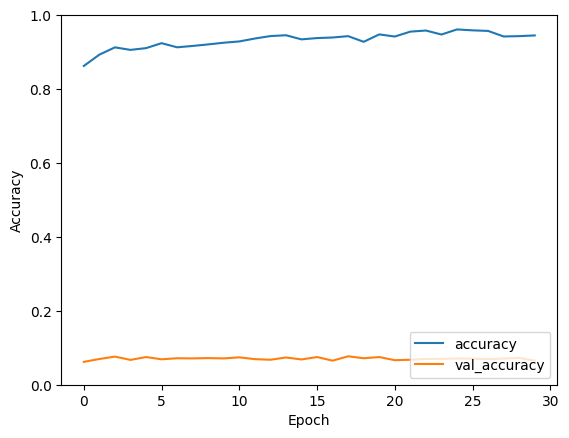


77/77 - 1s - loss: 11.7574 - accuracy: 0.0729 - 1s/epoch - 13ms/step

Total Accuracy: 7.29%

Let’s change the loss function

Loss parameter = False results:



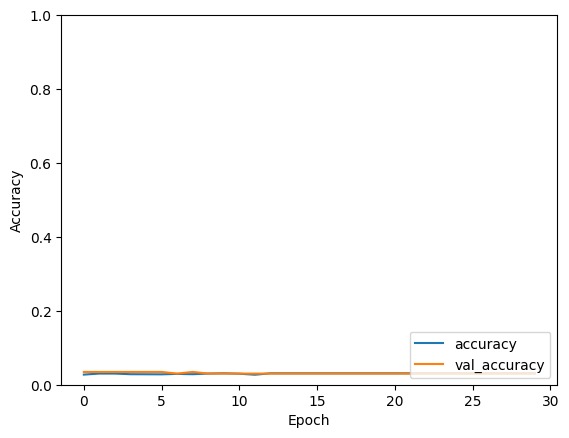
77/77 - 1s - loss: 15.2554 - accuracy: 0.0651 - 877ms/epoch - 11ms/step

Total accuracy = 6.51%

Let’s change number of neurons in first dense layer

From 100 to 10?

10 neurons in first dense layer results:

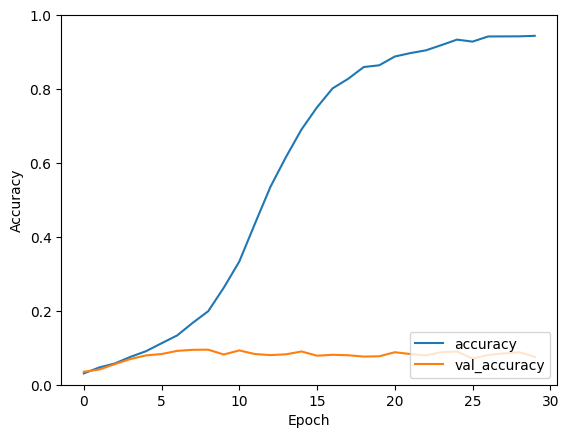


77/77 - 1s - loss: 4.5063 - accuracy: 0.0301 - 1s/epoch - 13ms/step

Total Accuracy = 3.01%

Let’s increase it to 204 neurons

204 neurons in first dense layer results:

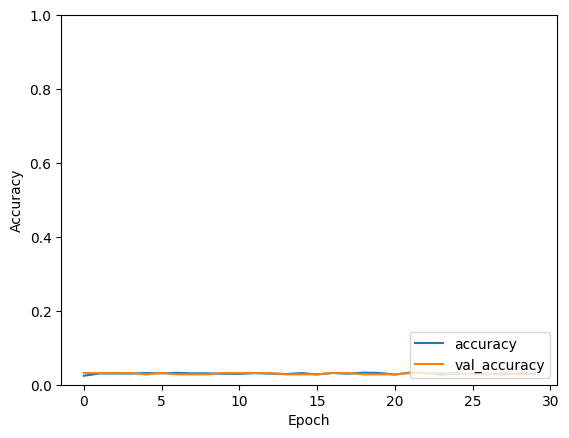


77/77 - 1s - loss: 10.2876 - accuracy: 0.0753 - 999ms/epoch - 13ms/step

Total accuracy = 7.53%

Let’s change learning rate from 0.001 to 0.01

LR 0.01 results:



77/77 - 1s - loss: 4.5154 - accuracy: 0.0317 - 1s/epoch - 14ms/step

Total accuracy = 3.17%

Let’s make learning rate smaller

From 0.01 to 0.0001