In [1]:

```
# Importing Libraries
import numpy as np
import tensorflow as tf
import keras
from keras import datasets,models,layers
from keras.models import Sequential
from keras.layers import Conv2D
import matplotlib.pyplot as plt
```

In [2]:

```
#Loading the data
(x_train,y_train ) ,( x_test,y_test)=keras.datasets.cifar100.load_data()
```

In [3]:

```
#checking the shape of train and test data
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

Out[3]:

```
((50000, 32, 32, 3), (50000, 1), (10000, 32, 32, 3), (10000, 1))
```

In [4]:

```
# checking number of different outputcomes
np.unique(y_train)
```

Out[4]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [5]:

```
#scaling the train data
x_train=x_train/255
x_test=x_test/255
```

In [6]:

```
model=models.Sequential([layers.Conv2D(filters=64,kernel_size=(2,2),strides=(2,2) ,padding="same" ,input_shape=
                        layers.MaxPool2D(pool_size=(2,2),padding="same",strides=(1,1)),
                        layers.Conv2D(32,kernel_size=(2,2),strides=(2,2) ,padding="same" ),
                        layers.MaxPool2D(pool_size=(2,2),padding="same",strides=(1,1)) ,
                        layers.Flatten(),
                        layers.Dense(300,activation="relu"),
                        layers.Dense(100,activation="softmax")])
```

In [7]:

```
model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics="accuracy")
```

In [8]:

```
history=model.fit(x_train,y_train,epochs=10,validation_data=(x_test,y_test),batch_size=200)
```

```
Epoch 1/10
250/250 [==============================] - 14s 54ms/step - loss: 3.5932 - accuracy: 0.1671 - val_loss: 3.1631 -
val_accuracy: 0.2429
Epoch 2/10
250/250 [==============================] - 14s 55ms/step - loss: 2.9101 - accuracy: 0.2886 - val_loss: 2.8328 -
val_accuracy: 0.3094
Epoch 3/10
250/250 [==============================] - 14s 55ms/step - loss: 2.5986 - accuracy: 0.3497 - val_loss: 2.6963 -
val_accuracy: 0.3358
Epoch 4/10
250/250 [==============================] - 15s 59ms/step - loss: 2.3867 - accuracy: 0.3929 - val_loss: 2.5725 -
val_accuracy: 0.3628
Epoch 5/10
250/250 [==============================] - 15s 62ms/step - loss: 2.1984 - accuracy: 0.4357 - val_loss: 2.5244 -
val_accuracy: 0.3773
Epoch 6/10
250/250 [==============================] - 23s 93ms/step - loss: 2.0480 - accuracy: 0.4677 - val_loss: 2.4839 -
val_accuracy: 0.3914
Epoch 7/10
250/250 [==============================] - 18s 71ms/step - loss: 1.8984 - accuracy: 0.5030 - val_loss: 2.5050 -
val_accuracy: 0.3872
Epoch 8/10
250/250 [==============================] - 15s 62ms/step - loss: 1.7690 - accuracy: 0.5335 - val_loss: 2.5441 -
val_accuracy: 0.3900
Epoch 9/10
250/250 [==============================] - 13s 52ms/step - loss: 1.6367 - accuracy: 0.5664 - val_loss: 2.5513 -
val_accuracy: 0.3984
Epoch 10/10
250/250 [==============================] - 14s 54ms/step - loss: 1.5174 - accuracy: 0.5907 - val_loss: 2.5813 -
val_accuracy: 0.3957
```

In [9]:

```
model.summary()
```

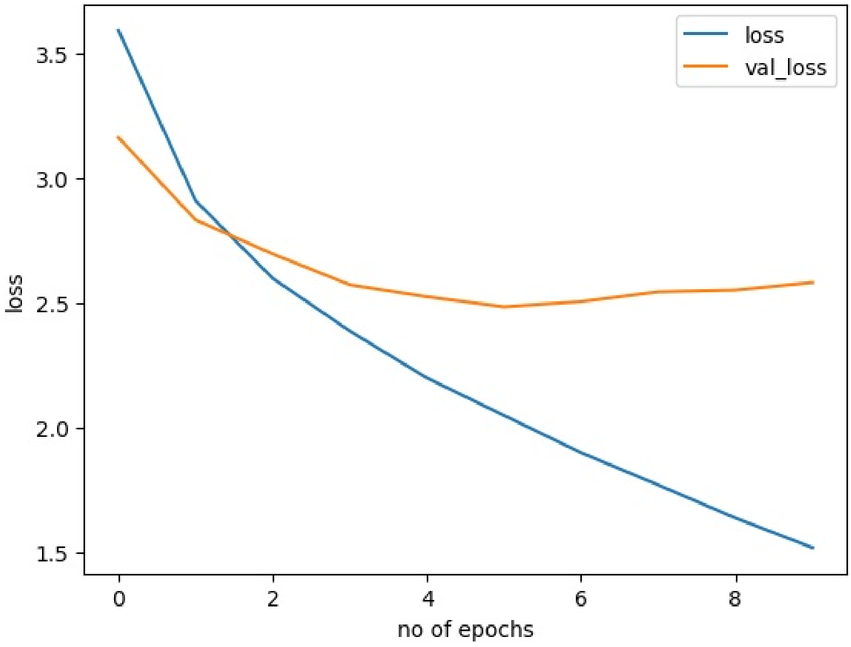
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 16, 16, 64)        832

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 8, 8, 32)          8224

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 300)               614700

 dense_1 (Dense)             (None, 100)               30100

=================================================================
Total params: 653,856
Trainable params: 653,856
Non-trainable params: 0
_________________________________________________________________
```

In [10]:

```python
import matplotlib.pyplot as plt
```

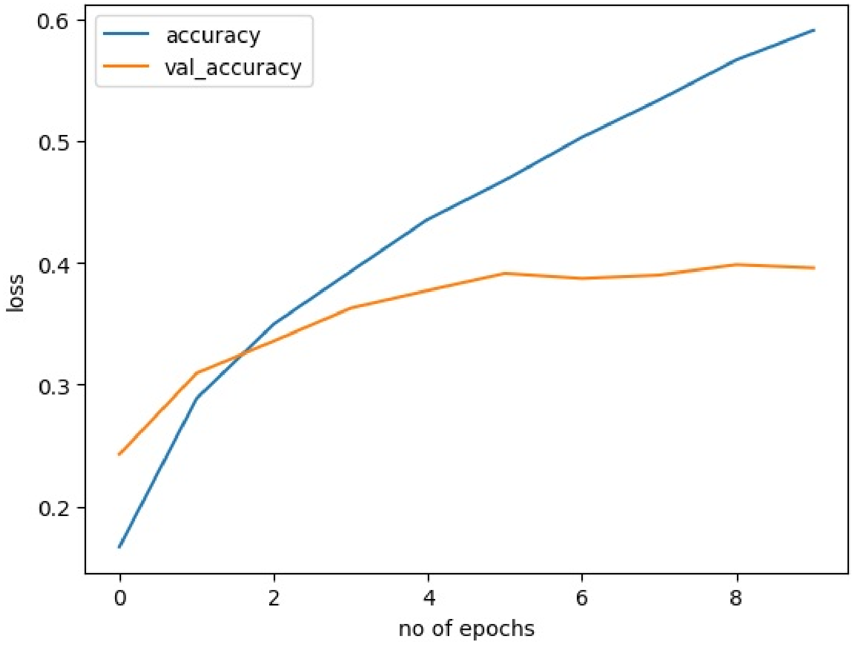
In [11]:

```python
plt.plot(history.history["loss"],label="loss")
plt.plot(history.history["val_loss"],label="val_loss")
plt.legend()
plt.xlabel("no of epochs")
plt.ylabel("loss")
plt.show()
```



In [12]:

```python
plt.plot(history.history["accuracy"],label="accuracy")
plt.plot(history.history["val_accuracy"],label="val_accuracy")
plt.legend()
plt.xlabel("no of epochs")
plt.ylabel("loss")
plt.show()
```



```
In [13]: model.evaluate(x_test,y_test)

313/313 [==============================] - 2s 6ms/step - loss: 2.5813 - accuracy: 0.3957
Out[13]: [2.581334114074707, 0.39570000767707825]

In [14]: prediction=model.predict(x_test)

313/313 [==============================] - 2s 6ms/step

In [15]: prediction.shape

Out[15]: (10000, 100)

In [16]: pred=(np.argmax(i) for i in prediction)

In [17]: y_test[0:10]

Out[17]: array([[49],
                [33],
                [72],
                [51],
                [71],
                [92],
                [15],
                [14],
                [23],
                [ 0]])

In [18]: y_true=[]
         for i in pred:
             y_true.append(i)
         y_true[0:10]

Out[18]: [49, 80, 55, 91, 71, 79, 27, 26, 23, 83]

In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js