In [2]:
```
#Assignment No: 03
#Title: To Implement Image classification model using CNN Deep Learning Arch
#Dataset: cifar10
```

In [23]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets,layers,models
import matplotlib.pyplot as plt
```

In [24]:
```
cifar10=keras.datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.
```

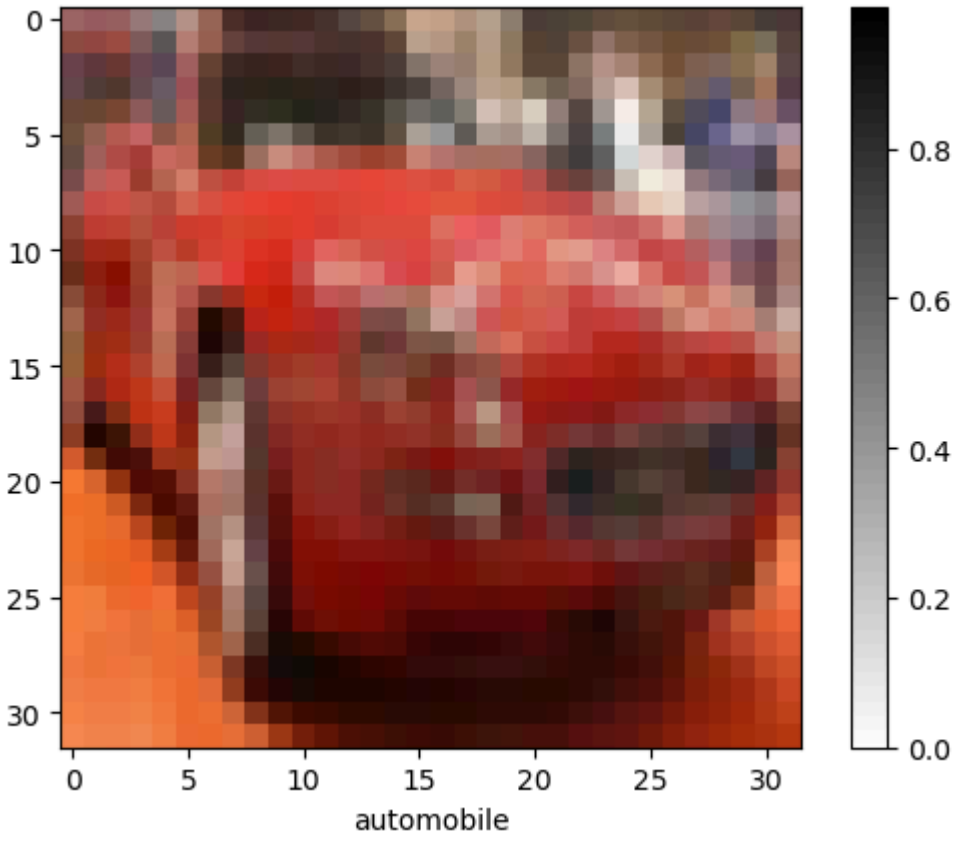
In [25]:
```
train_images, test_images = train_images/255.0, test_images/255.0
```

In [26]:
```
print("train_images: ", train_images.shape)
print("test_images: ", test_images.shape)
```

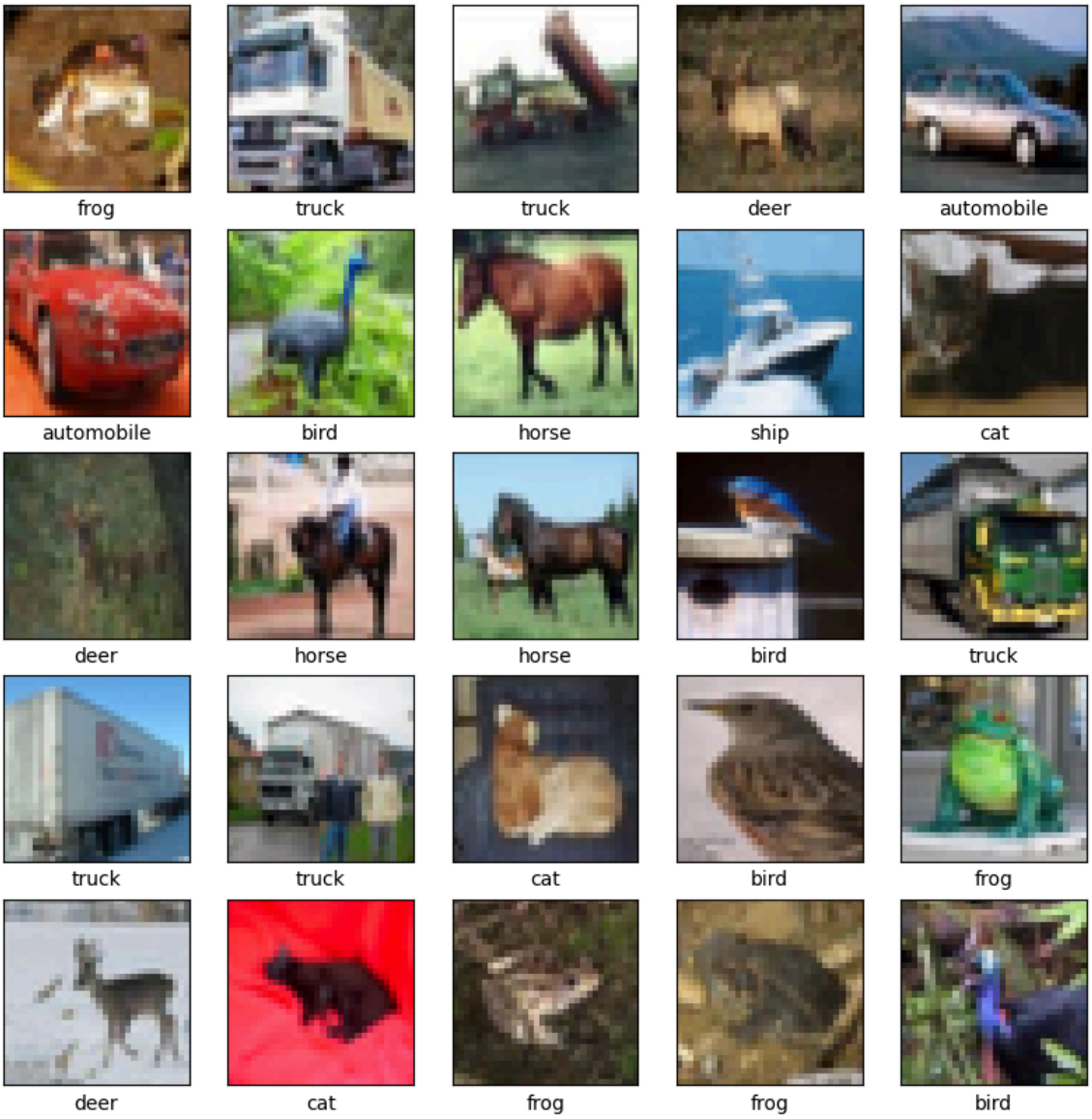
```
train_images:  (50000, 32, 32, 3)
test_images:  (10000, 32, 32, 3)
```

In [27]:
```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog
```

In [28]:
```
plt.figure()
plt.imshow(train_images[5],cmap=plt.cm.binary)
plt.colorbar()
plt.grid(False)
plt.xlabel(class_names[train_labels[5][0]])
plt.show()
```



In [29]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i],cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```



In [30]:
```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

In [31]:
```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	36,928

Total params: 56,320 (220.00 KB)

Trainable params: 56,320 (220.00 KB)

Non-trainable params: 0 (0.00 B)

In [33]:
```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation ='relu'))
model.add(layers.Dense(10))
```

In [34]:
```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	36,928
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65,600
dense_3 (Dense)	(None, 10)	650
flatten_2 (Flatten)	(None, 10)	0
dense_4 (Dense)	(None, 64)	704
dense_5 (Dense)	(None, 10)	650

Total params: 123,924 (484.08 KB)

Trainable params: 123,924 (484.08 KB)

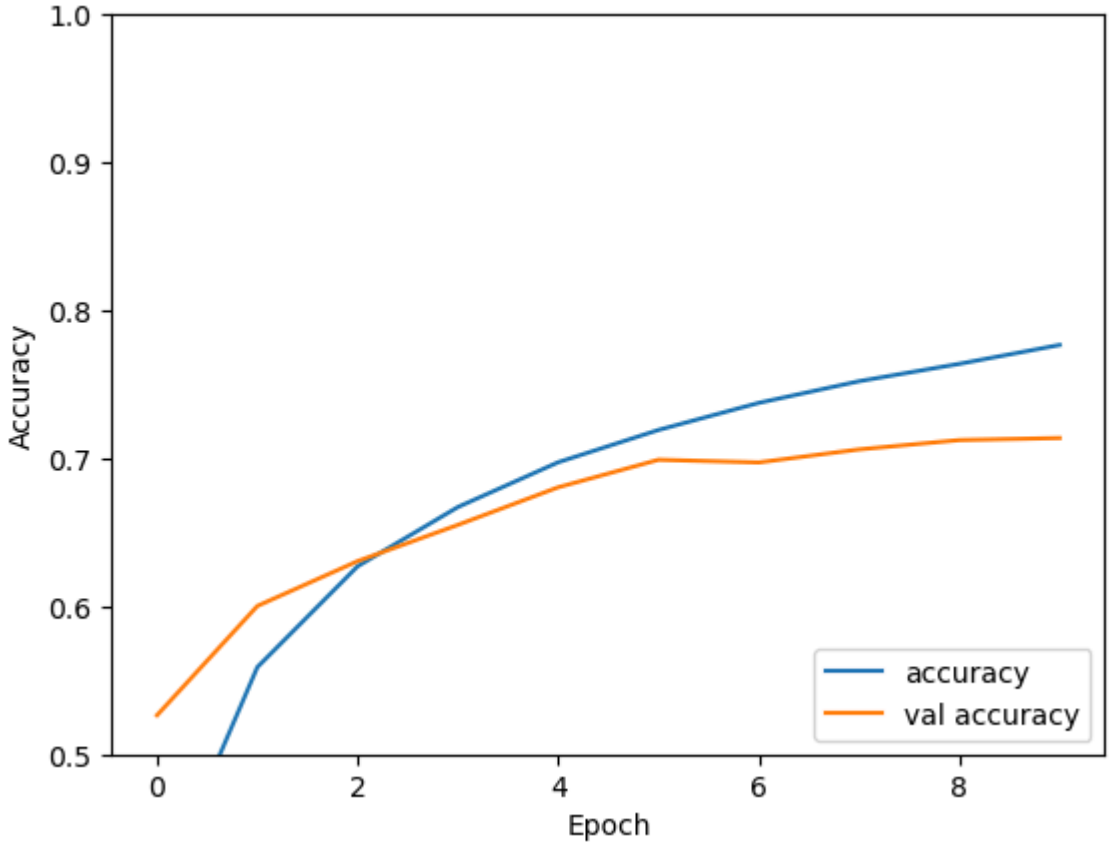
Non-trainable params: 0 (0.00 B)

In [35]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=10, validation_data=(
```

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 49s 27ms/step - accuracy: 0.3000 - loss: 1.
8483 - val_accuracy: 0.5264 - val_loss: 1.3117
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 45s 29ms/step - accuracy: 0.5391 - loss: 1.
2737 - val_accuracy: 0.6002 - val_loss: 1.1345
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 49s 31ms/step - accuracy: 0.6158 - loss: 1.
0698 - val_accuracy: 0.6305 - val_loss: 1.0436
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 48s 31ms/step - accuracy: 0.6639 - loss: 0.
9475 - val_accuracy: 0.6550 - val_loss: 0.9740
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 47s 30ms/step - accuracy: 0.6984 - loss: 0.
8551 - val_accuracy: 0.6804 - val_loss: 0.9276
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 77s 26ms/step - accuracy: 0.7199 - loss: 0.
7934 - val_accuracy: 0.6990 - val_loss: 0.8608
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 47s 30ms/step - accuracy: 0.7392 - loss: 0.
7346 - val_accuracy: 0.6972 - val_loss: 0.8603
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 44s 28ms/step - accuracy: 0.7512 - loss: 0.
6943 - val_accuracy: 0.7060 - val_loss: 0.8587
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 42s 27ms/step - accuracy: 0.7682 - loss: 0.
6589 - val_accuracy: 0.7123 - val_loss: 0.8507
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 42s 27ms/step - accuracy: 0.7781 - loss: 0.
6216 - val_accuracy: 0.7137 - val_loss: 0.8568
```

In [36]:
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label= 'val accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose = 2)
```

```
313/313 - 3s - 10ms/step - accuracy: 0.7137 - loss: 0.8568
```



In [ ]: