In [1]:
```
from keras.datasets import cifar10
from keras import layers, models, optimizers
from keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation= 'relu', input_shape= (32, 32, 3)))
model.add(layers.Conv2D(64, (3, 3), activation= 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128, (3,3), activation= 'relu'))
model.add(layers.Conv2D(128, (3,3), activation= 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation= 'relu'))
model.add(layers.Dense(10, activation= 'softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 3200)              0
_________________________________________________________________
dropout (Dropout)            (None, 3200)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1638912
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,884,874
Trainable params: 1,884,874
Non-trainable params: 0
_________________________________________________________________
```

In [3]:
```
model.compile(optimizer= optimizers.RMSprop(lr=1e-4),
              loss= 'categorical_crossentropy',
              metrics= ['accuracy'])
```

In [4]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

In [5]:
```
train_datagen = ImageDataGenerator(rescale= 1./255,
                                   rotation_range= 40,
                                   width_shift_range= 0.2,
                                   height_shift_range= 0.2,
                                   shear_range= 0.2,
                                   zoom_range= 0.2,
                                   horizontal_flip= True)
```

In [6]:
```
x_test = x_test.astype('float32') / 255

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

x_val = x_train[:1000]
x_val = x_val.astype('float32')
x_part = x_train[1000:3000] # select small data set to see effect of data augment
y_val = y_train[:1000]
y_part = y_train[1000:3000]

train_generator = train_datagen.flow(x_part, y= y_part, batch_size= 20)
```

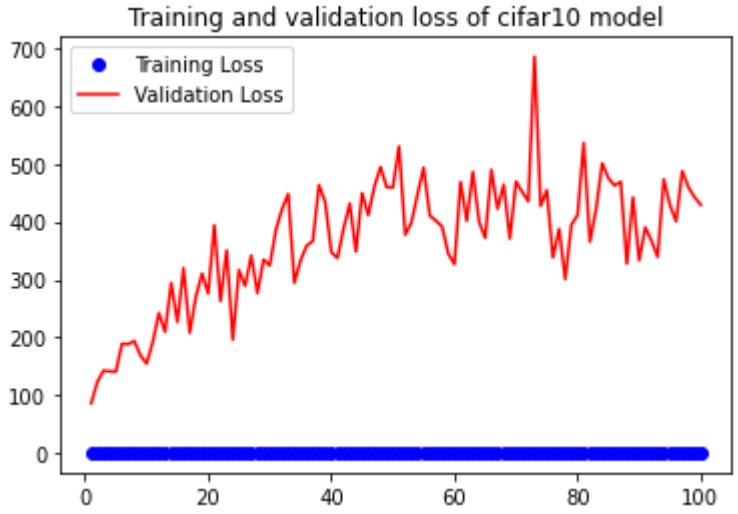
In [7]:
```
history = model.fit(train_generator, steps_per_epoch= 100, epochs= 100, validatic
```

```
Epoch 83/100
100/100 [==============================] - 3s 28ms/step - loss: 1.2942 - accu
racy: 0.5270 - val_loss: 421.3049 - val_accuracy: 0.3390
Epoch 84/100
100/100 [==============================] - 3s 28ms/step - loss: 1.2998 - accu
racy: 0.5410 - val_loss: 500.6773 - val_accuracy: 0.3030
Epoch 85/100
100/100 [==============================] - 3s 28ms/step - loss: 1.2917 - accu
racy: 0.5415 - val_loss: 475.4423 - val_accuracy: 0.3270
Epoch 86/100
100/100 [==============================] - 3s 28ms/step - loss: 1.3149 - accu
racy: 0.5260 - val_loss: 462.2011 - val_accuracy: 0.3380
Epoch 87/100
100/100 [==============================] - 3s 28ms/step - loss: 1.3038 - accu
racy: 0.5475 - val_loss: 468.8743 - val_accuracy: 0.3350
Epoch 88/100
100/100 [==============================] - 3s 28ms/step - loss: 1.3038 - accu
racy: 0.5370 - val_loss: 327.4172 - val_accuracy: 0.3950
Epoch 89/100
100/100 [==============================] - 3s 28ms/step - loss: 1.2764 - accu
```

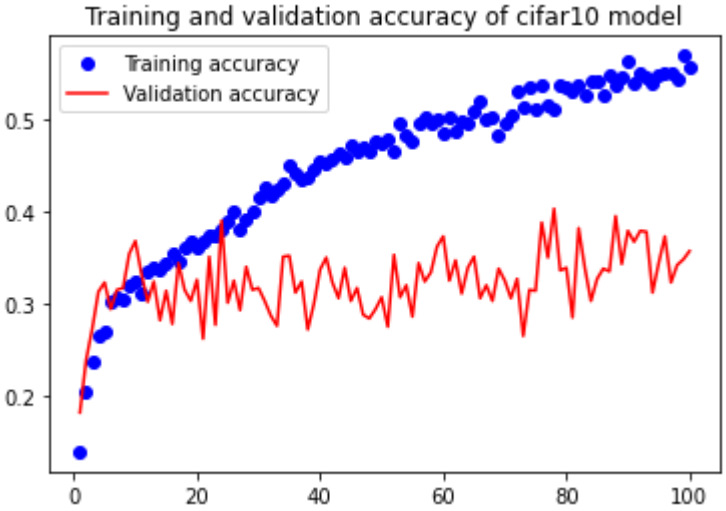
In [8]:
```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) +1)

plt.plot(epochs, loss, 'bo', label= 'Training Loss')
plt.plot(epochs, val_loss, 'r', label= 'Validation Loss')
plt.title('Training and validation loss of cifar10 model')
plt.legend()
plt.savefig('Results/6_2b/Loss.png');
```



In [10]:
```
plt.plot(epochs, acc, 'bo', label= 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label= 'Validation accuracy')
plt.title('Training and validation accuracy of cifar10 model')
plt.legend()
plt.savefig('Results/6_2b/Accuracy.png');
```



In [11]:
```
model.save('Results/6_2b/model2.h5')
```

In [12]:
```
result = model.evaluate(x_test, y_test)
print(f'The model result for the test data is loss {result[0]} and accuracy {resu
```

```
313/313 [==============================] - 3s 11ms/step - loss: 1.3944 - accura
cy: 0.5337
The model result for the test data is loss 1.3944133520126343 and accuracy 0.53
36999893188477
```

In [13]:
```
import numpy as np
import pandas as pd
preds = model.predict(x_test)
pred_class = np.argmax(preds, axis= 1)
act = np.argmax(y_test, axis= 1)
pred_df = pd.DataFrame({'Actual':act, 'Predicted':pred_class})
pred_df
```

Out[13]:

	Actual	Predicted
0	3	3
1	8	1
2	8	8
3	0	0
4	6	4
...	...	...
9995	8	5
9996	3	6
9997	5	5
9998	1	1
9999	7	7

10000 rows × 2 columns

In [14]:
```
pred_df.to_csv('Results/6_2b/predictions.csv')
```

In [ ]:
```

```