In [25]:
```
from keras.datasets import cifar10
from keras import layers, models, optimizers
from keras.utils import to_categorical
```

In [26]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

In [27]:
```
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.Dense(512, activation= 'relu'))
model.add(layers.Dense(10, activation= 'softmax'))

model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 30, 30, 32)        896
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 12, 12, 128)       73856
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 10, 10, 128)       147584
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               1638912
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,884,874
Trainable params: 1,884,874
Non-trainable params: 0
_________________________________________________________________
```

In [28]:
```
model.compile(optimizer= optimizers.RMSprop(lr=1e-4),
              loss= 'categorical_crossentropy',
              metrics= ['accuracy'])
```

In [29]:
```
x_train = x_train.astype('float32') / 255
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_part = x_train[1000:3000] # select small data set to see effect of data augm
entation in part 2
y_val = y_train[:1000]
y_part = y_train[1000:3000]
```

In [30]:
```
x_train.shape, y_train.shape
```

Out[30]:
```
((50000, 32, 32, 3), (50000, 10))
```

In [32]:
```
history = model.fit(x_part, y_part, epochs= 30, validation_data= (x_val, y_val
))
```

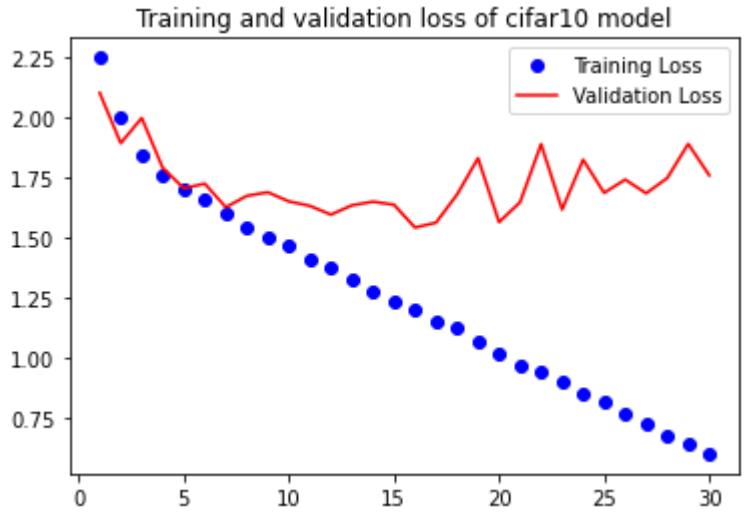
```
Epoch 1/30
63/63 [==============================] - 2s 36ms/step - loss: 2.2512 - accura
cy: 0.1645 - val_loss: 2.1037 - val_accuracy: 0.2490
Epoch 2/30
63/63 [==============================] - 3s 44ms/step - loss: 1.9988 - accura
cy: 0.2725 - val_loss: 1.8941 - val_accuracy: 0.3090
Epoch 3/30
63/63 [==============================] - 2s 33ms/step - loss: 1.8471 - accura
cy: 0.3395 - val_loss: 1.9991 - val_accuracy: 0.2910
Epoch 4/30
63/63 [==============================] - 2s 33ms/step - loss: 1.7605 - accura
cy: 0.3780 - val_loss: 1.7903 - val_accuracy: 0.3500
Epoch 5/30
63/63 [==============================] - 2s 33ms/step - loss: 1.7042 - accura
cy: 0.4015 - val_loss: 1.7056 - val_accuracy: 0.3990
Epoch 6/30
63/63 [==============================] - 2s 34ms/step - loss: 1.6563 - accura
cy: 0.4150 - val_loss: 1.7250 - val_accuracy: 0.3940
Epoch 7/30
63/63 [==============================] - 2s 33ms/step - loss: 1.6003 - accura
cy: 0.4345 - val_loss: 1.6265 - val_accuracy: 0.4260
Epoch 8/30
63/63 [==============================] - 2s 33ms/step - loss: 1.5456 - accura
cy: 0.4445 - val_loss: 1.6740 - val_accuracy: 0.4040
Epoch 9/30
63/63 [==============================] - 2s 32ms/step - loss: 1.5008 - accura
cy: 0.4665 - val_loss: 1.6890 - val_accuracy: 0.3900
Epoch 10/30
63/63 [==============================] - 2s 33ms/step - loss: 1.4637 - accura
cy: 0.4915 - val_loss: 1.6509 - val_accuracy: 0.3960
Epoch 11/30
63/63 [==============================] - 2s 33ms/step - loss: 1.4124 - accura
cy: 0.4930 - val_loss: 1.6323 - val_accuracy: 0.4060
Epoch 12/30
63/63 [==============================] - 2s 33ms/step - loss: 1.3789 - accura
cy: 0.5150 - val_loss: 1.5957 - val_accuracy: 0.4450
Epoch 13/30
63/63 [==============================] - 2s 33ms/step - loss: 1.3240 - accura
cy: 0.5315 - val_loss: 1.6346 - val_accuracy: 0.4080
Epoch 14/30
63/63 [==============================] - 2s 32ms/step - loss: 1.2717 - accura
cy: 0.5545 - val_loss: 1.6498 - val_accuracy: 0.3980
Epoch 15/30
63/63 [==============================] - 2s 33ms/step - loss: 1.2294 - accura
cy: 0.5650 - val_loss: 1.6371 - val_accuracy: 0.4340
Epoch 16/30
63/63 [==============================] - 2s 33ms/step - loss: 1.1968 - accura
cy: 0.5845 - val_loss: 1.5417 - val_accuracy: 0.4600
Epoch 17/30
63/63 [==============================] - 2s 33ms/step - loss: 1.1474 - accura
cy: 0.5895 - val_loss: 1.5620 - val_accuracy: 0.4450
Epoch 18/30
63/63 [==============================] - 2s 32ms/step - loss: 1.1200 - accura
cy: 0.6045 - val_loss: 1.6792 - val_accuracy: 0.4430
Epoch 19/30
63/63 [==============================] - 2s 32ms/step - loss: 1.0627 - accura
cy: 0.6335 - val_loss: 1.8313 - val_accuracy: 0.4030
```

```
Epoch 20/30
63/63 [==============================] - 2s 33ms/step - loss: 1.0158 - accura
cy: 0.6410 - val_loss: 1.5643 - val_accuracy: 0.4670
Epoch 21/30
63/63 [==============================] - 2s 32ms/step - loss: 0.9686 - accura
cy: 0.6630 - val_loss: 1.6474 - val_accuracy: 0.4440
Epoch 22/30
63/63 [==============================] - 2s 32ms/step - loss: 0.9395 - accura
cy: 0.6785 - val_loss: 1.8900 - val_accuracy: 0.4350
Epoch 23/30
63/63 [==============================] - 2s 33ms/step - loss: 0.8998 - accura
cy: 0.6900 - val_loss: 1.6169 - val_accuracy: 0.4730
Epoch 24/30
63/63 [==============================] - 2s 33ms/step - loss: 0.8512 - accura
cy: 0.7055 - val_loss: 1.8251 - val_accuracy: 0.4340
Epoch 25/30
63/63 [==============================] - 2s 33ms/step - loss: 0.8152 - accura
cy: 0.7200 - val_loss: 1.6865 - val_accuracy: 0.4450
Epoch 26/30
63/63 [==============================] - 2s 33ms/step - loss: 0.7613 - accura
cy: 0.7390 - val_loss: 1.7419 - val_accuracy: 0.4480
Epoch 27/30
63/63 [==============================] - 2s 33ms/step - loss: 0.7224 - accura
cy: 0.7475 - val_loss: 1.6848 - val_accuracy: 0.4720
Epoch 28/30
63/63 [==============================] - 2s 33ms/step - loss: 0.6751 - accura
cy: 0.7700 - val_loss: 1.7483 - val_accuracy: 0.4710
Epoch 29/30
63/63 [==============================] - 2s 34ms/step - loss: 0.6349 - accura
cy: 0.7870 - val_loss: 1.8911 - val_accuracy: 0.4530
Epoch 30/30
63/63 [==============================] - 2s 32ms/step - loss: 0.5975 - accura
cy: 0.8065 - val_loss: 1.7600 - val_accuracy: 0.4840
```

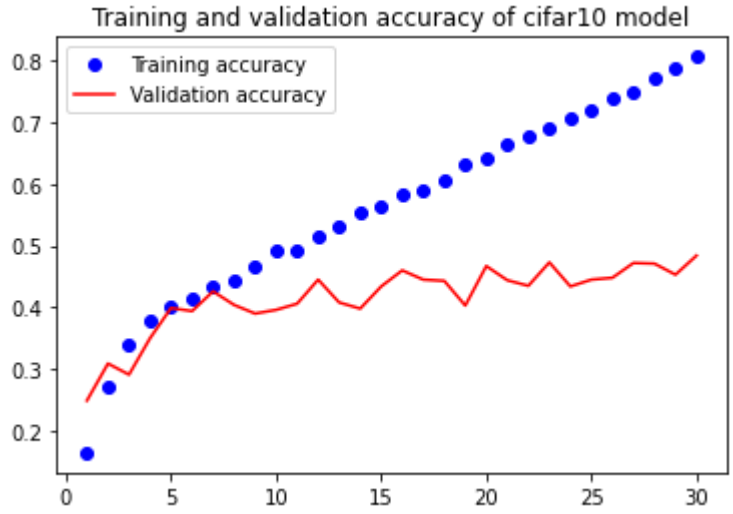
In [35]:
```
import matplotlib.pyplot as plt

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_2a/Loss.png');
```



In [36]:
```
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_2a/Accuracy.png');
```



In [38]:
```
model.save('Results/6_2a/model1.h5')
```

In [39]:
```
result = model.evaluate(x_test, y_test)
print(f'The model result for the test data is loss {result[0]} and accuracy {result[1]}')
```

```
313/313 [==============================] - 3s 11ms/step - loss: 1.7394 - accuracy: 0.4712
The model reult for the test data is loss 1.7394400835037231 and accuracy 0.47119998931884766
```

In [40]:
```
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[40]:

	Actual	Predicted
0	3	3
1	8	1
2	8	9
3	0	8
4	6	5
...	...	...
9995	8	5
9996	3	6
9997	5	3
9998	1	4
9999	7	7

10000 rows × 2 columns

In [41]:
```
pred_df.to_csv('Results/6_2a/predictions.csv')
```

In [ ]: