1 第二题

用VGG16进行特征提取，进行cifar-10的分类

In [59]:

```
# 导入预训练模型VGG16模型的卷积基
from tensorflow.keras.applications import VGG16
conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(32, 32, 3))
```

In [60]:

```
# 数据及准备
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
import numpy as np
# 从keras中读取cifar10数据集
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# 数据预处理，标准化处理
x_train = x_train.reshape((50000, 32, 32, 3))
x_train = x_train.astype('float32')/255
x_test = x_test.reshape((10000, 32, 32, 3))
x_test = x_test.astype('float32')/255
# 训练数据划分为训练集和验证集
x_val = x_train[:1000]                          ## 前1000个样本作验证集Validation
partial_x_train = x_train[1000:]                ## 其余样本作训练集
```

In [61]:

```
y_train.shape
```

Out[61]:

```
(50000, 1)
```

In [62]:

```
partial_x_train.shape
```

Out[62]:

```
(49000, 32, 32, 3)
```

In [63]:

```
# 提取数据特征：调用卷积基输出函数分别提取训练集、验证集、测试集的特征(samples,4,4,512)，并将其展
train_features = np.reshape(conv_base.predict(partial_x_train), (49000, 1 * 1 * 512))
validation_features = np.reshape(conv_base.predict(x_val), (1000, 1 * 1 * 512))
test_features = np.reshape(conv_base.predict(x_test), (10000, 1 * 1 * 512))
```

```
WARNING:tensorflow:AutoGraph could not transform <function Model.make_
predict_function.<locals>.predict_function at 0x168909820> and will ru
n it as-is.
Please report this to the TensorFlow team. When filing the bug, set th
e verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and atta
ch the full output.
Cause: unsupported operand type(s) for -: 'NoneType' and 'int'
To silence this warning, decorate the function with @tf.autograph.expe
rimental.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_predict_fu
nction.<locals>.predict_function at 0x168909820> and will run it as-i
s.
Please report this to the TensorFlow team. When filing the bug, set th
e verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and atta
ch the full output.
Cause: unsupported operand type(s) for -: 'NoneType' and 'int'
To silence this warning, decorate the function with @tf.autograph.expe
rimental.do_not_convert
```

In [64]:

```
# 将类别标签转化为独热矩阵表示
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [65]:

```
# 将类别标签也相应划分为训练集、验证集、测试集
y_val = y_train[:1000]              ## 前1000个验证集样本标签
partial_y_train = y_train[1000:] ## 训练集样本标签
train_labels = partial_y_train
validation_labels = y_val
test_labels = y_test
```

In [66]:

```
# 定义全连接分类器
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
model = models.Sequential()
##model.add(layers.Flatten(input_shape=conv_base.output_shape[1:]))
model.add(layers.Dense(256, activation='relu', input_dim=1 * 1 * 512))
model.add(layers.Dropout(0.5))
# model.add(layers.Dense(64, activation='relu'))
# model.add(layers.Dropout(0.25))
model.add(layers.Dense(10, activation='softmax'))
```

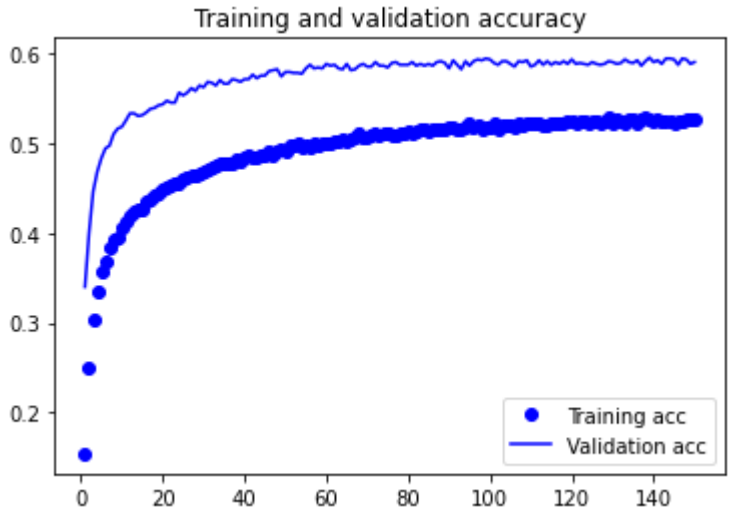
In [67]:

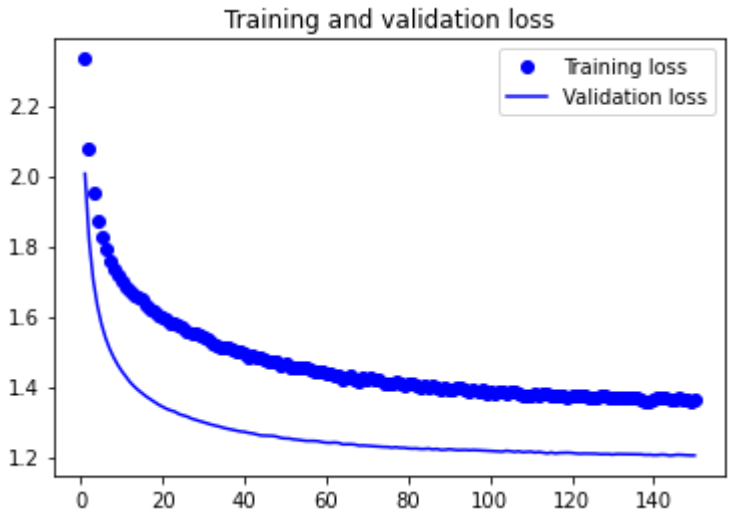
```
# 编译和训练全连接分类器
model.compile(optimizer=optimizers.RMSprop(lr=2e-5), loss='categorical_crossentropy'
history = model.fit(train_features, train_labels, epochs=150, batch_size=64, validat
##history = model.fit(train_features, train_labels, epochs=30, batch_size=256, valid
```

```
Epoch 145/150
766/766 [==============================] - 1s 801us/step - loss: 1.363
6 - accuracy: 0.5225 - val_loss: 1.2052 - val_accuracy: 0.5930
Epoch 146/150
766/766 [==============================] - 1s 858us/step - loss: 1.361
6 - accuracy: 0.5257 - val_loss: 1.2068 - val_accuracy: 0.5870
Epoch 147/150
766/766 [==============================] - 1s 786us/step - loss: 1.363
0 - accuracy: 0.5245 - val_loss: 1.2060 - val_accuracy: 0.5940
Epoch 148/150
766/766 [==============================] - 1s 800us/step - loss: 1.359
0 - accuracy: 0.5296 - val_loss: 1.2051 - val_accuracy: 0.5930
Epoch 149/150
766/766 [==============================] - 1s 774us/step - loss: 1.360
5 - accuracy: 0.5274 - val_loss: 1.2047 - val_accuracy: 0.5880
Epoch 150/150
766/766 [==============================] - 1s 775us/step - loss: 1.357
2 - accuracy: 0.5282 - val_loss: 1.2047 - val_accuracy: 0.5900
```

In [68]:

```
# 绘制训练过程中的损失曲线和精度曲线
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, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





In [69]:

```
# 在测试集上评估模型性能
test_loss, test_acc = model.evaluate(test_features, test_labels)
print('loss = {}, accuracy = {}'.format(test_loss, test_acc))
# 实验结论
# loss = 1.2414, accuracy = 0.5653   ## epochs=150, batch_size=64
```

```
313/313 [==============================] - 0s 271us/step - loss: 1.241
4 - accuracy: 0.5653
loss = 1.241441249847412, accuracy = 0.5652999877929688
```

2 第三题

微调最后一个卷积层，进行cifar-10的分类

In [70]:

```
# 数据及准备
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
# 从keras中读取cifar10数据集
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# 数据预处理，标准化处理
x_train = x_train.reshape((50000, 32, 32, 3))
x_train = x_train.astype('float32')/255
x_test = x_test.reshape((10000, 32, 32, 3))
x_test = x_test.astype('float32')/255
# 训练数据划分为训练集和验证集
x_val = x_train[:1000]                          ## 前1000个样本作验证集Validation
partial_x_train = x_train[1000:]                ## 其余样本作训练集
train_x = partial_x_train
# 将类别标签转化为独热矩阵表示
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
# 将类别标签也相应划分为训练集、验证集、测试集
y_val = y_train[:1000]              ## 前1000个验证集样本标签
partial_y_train = y_train[1000:] ## 训练集样本标签
train_y = partial_y_train
```

In [71]:

```
# 第一次训练
# 定义全连接分类器
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
model = models.Sequential()
# 添加卷积基并冻结
model.add(conv_base)
conv_base.trainable = False       ## 卷积基冻结，不参加训练，否则网络参数太多！
# 添加Dense层
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
print(model.summary())            ## 查看模型
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 1, 1, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_19 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_20 (Dense)             (None, 10)                2570
=================================================================
Total params: 14,848,586
Trainable params: 133,898
Non-trainable params: 14,714,688
_________________________________________________________________
None
```

In [72]:

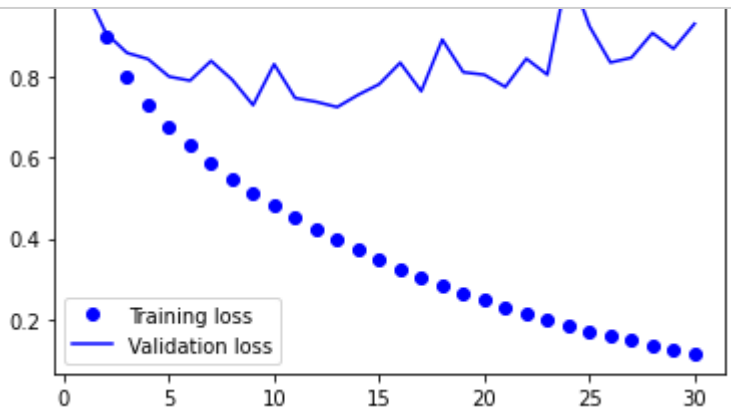
```
# 编译和训练全连接分类器
model.compile(optimizer=optimizers.RMSprop(lr=2e-5), loss='categorical_crossentropy'
history = model.fit(train_x, train_y, epochs=30, batch_size=256, validation_data=(x_
```

```
Epoch 1/30
WARNING:tensorflow:AutoGraph could not transform <function Model.make_
train_function.<locals>.train_function at 0x2c77b41f0> and will run it
as-is.
Please report this to the TensorFlow team. When filing the bug, set th
e verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and atta
ch the full output.
Cause: unsupported operand type(s) for -: 'NoneType' and 'int'
To silence this warning, decorate the function with @tf.autograph.expe
rimental.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_train_func
tion.<locals>.train_function at 0x2c77b41f0> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set th
e verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and atta
ch the full output.
Cause: unsupported operand type(s) for -: 'NoneType' and 'int'
To silence this warning, decorate the function with @tf.autograph.expe
rimental.do_not_convert
192/192 [==============================] - ETA: 0s - loss: 2.4064 - ac
```

In [73]:

```
# 第二次训练
# 从顶层开始解冻直到'block5_conv1'
conv_base.trainable = True          ## 解冻卷积基
set_trainable = False               ## 逻辑变量赋初值
# 循环从block1_conv1开始，直到block5_conv1将set_trainable 设为true，此后各层均layer.train
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True    ## 设置该层解冻
    else:
        layer.trainable = False   ## 设置该层冻结
# 编译和训练全连接分类器
model.compile(optimizer=optimizers.RMSprop(lr=2e-5), loss='categorical_crossentropy'
history = model.fit(train_x, train_y, epochs=30, batch_size=256, validation_data=(x_
# 绘制训练过程中的损失曲线和精度曲线
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, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()

# 在测试集上评估模型性能
test_loss, test_acc = model.evaluate(x_test, y_test)
print('loss = {}, accuracy = {}'.format(test_loss, test_acc))
#出现了一定的过拟合现象
```



In [74]:

```
# 在测试集上评估模型性能
test_loss, test_acc = model.evaluate(x_test, y_test)
print('loss = {}, accuracy = {}'.format(test_loss, test_acc))

#loss = 0.9345357418060303, accuracy = 0.7401000261306763
```

```
313/313 [==============================] - 10s 31ms/step - loss: 0.934
5 - accuracy: 0.7401
loss = 0.9345357418060303, accuracy = 0.7401000261306763
```