# 使用Keras进行MobileNet微调

这里使用Keras进行MobileNet的微调，用于两类的识别。这里虽是一个微调的实验，也演示使用Keras进行大量图像数据的读取和训练，及进Tensorboard及checkpoint的使用。

本实验的重点是使用ImageDataGenerator进行数据的扩充和大量图像数据的读取及网络的测试。

## 数据准备

数据的目录结构如下：



|  |
| --- |
| #data prepare  train\_datagen**=**ImageDataGenerator**(**rescale**=**1.**/**255**,**  shear\_range**=**0.2**,**  zoom\_range**=**0.2**,**  horizontal\_flip**=True)**  train\_generator**=**train\_datagen**.**flow\_from\_directory**(**directory**=**'E:/DataSet/tinySet/train'**,**  target\_size**=(**224**,** 224**),**  batch\_size**=**BATCH\_SIZE**,**  class\_mode**=**'categorical'**,**  color\_mode**=**'rgb'**)**  val\_datagen**=**ImageDataGenerator**(**rescale**=**1.**/**255**)**  val\_generator**=**val\_datagen**.**flow\_from\_directory**(**directory**=**'E:/DataSet/tinySet/val'**,**  target\_size**=(**224**,** 224**),**  batch\_size**=**BATCH\_SIZE**,**  class\_mode**=**'categorical'**,**  color\_mode**=**'rgb'**)**  class\_map **=** get\_class\_from\_generator**(**train\_generator**)** |

这里使用到了ImageDataGenerator及其方法flow\_from\_directory，ImageDataGenerator可以对数据进行增强扩充，其flow\_from\_directory方法可以得到数据生成器，用于从本地读取图像文件数据用于训练。

## 模型定义

模型的定义如下，使用Keras自带的MobileNet，去除top后，添加自己的全连接层，并将MobileNet的主体网络设置成不可训练，后面的全连接层设置成可训练。

|  |
| --- |
| **from** keras**.**layers **import** GlobalAveragePooling2D**,** Dense  **from** keras**.**applications**.**mobilenet **import** MobileNet  **from** keras**.**models **import** Model  **def** MobileNetTF**(**class\_num**):**  base\_model **=** MobileNet**(**include\_top**=False)**  x **=** base\_model**.**output  x **=** GlobalAveragePooling2D**()(**x**)**  x **=** Dense**(**1024**,** activation**=**'relu'**)(**x**)**  x **=** Dense**(**512**,** activation**=**'relu'**)(**x**)**  output **=** Dense**(**class\_num**,** activation**=**'softmax'**)(**x**)**  model **=** Model**(**inputs**=**base\_model**.**input**,** outputs**=**output**)**  layer\_num **=** len**(**model**.**layers**)**  **for** layer **in** model**.**layers**[:**layer\_num **-** 4**]:**  layer**.**trainable**=False**  **for** layer **in** model**.**layers**[**layer\_num **-** 4**:]:**  layer**.**trainable**=True**  **return** model |

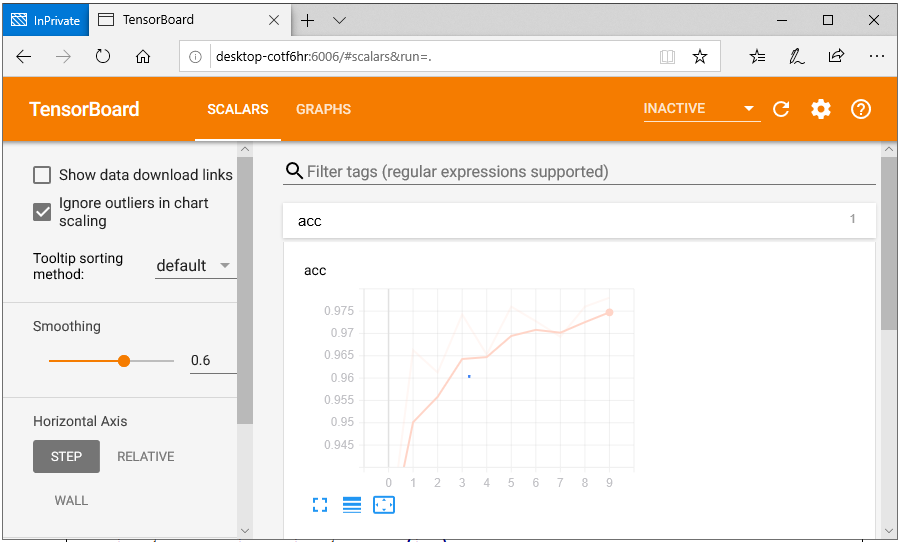
## 编译模型并训练

|  |
| --- |
| **if** TRAIN**:**  #train  checkpoint **=** ModelCheckpoint**(**filepath**=**'log/model-{epoch:02d}-{val\_acc:03f}.h5'**)**  tensorboard **=** TensorBoard**(**log\_dir**=**'log'**)**  model**.**compile**(**optimizer**=**'Adam'**,** loss**=**'categorical\_crossentropy'**,** metrics**=[**'accuracy'**])**  step\_size\_train **=** train\_generator**.**n **//** BATCH\_SIZE  validation\_steps **=** val\_generator**.**n **//** BATCH\_SIZE  model**.**fit\_generator**(**generator**=**train\_generator**,**  steps\_per\_epoch**=**step\_size\_train**,**  epochs**=**10**,**  validation\_data**=**val\_generator**,**  validation\_steps**=**validation\_steps**,**  callbacks**=[**checkpoint**,** tensorboard**]**  **)**  model**.**save**(**'dog\_cat.h5'**)**  **else:**  model**=**load\_model**(**'dog\_cat.h5'**)** |

在命令行中输入tensorboard,并在--logdir=后指定log目录：



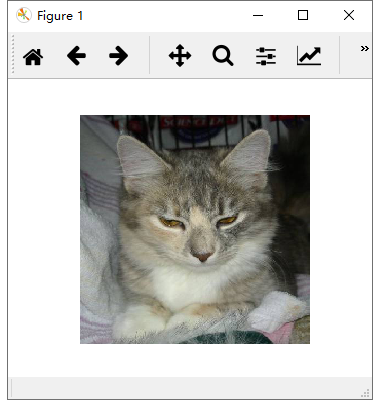
在浏览器中打开http://DESKTOP-COTF6HR:6006

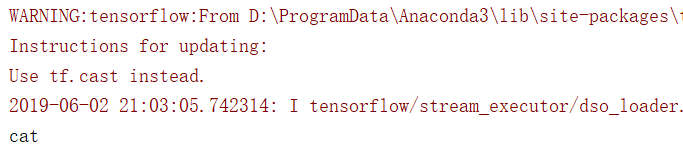


## 测试

从前面的ImageDataGenerator可以看出，图像数据读入后需要resize到（224，224）大小，乘以1./255，且应为4维张量，因此需要对测试的图像进行预处理。

|  |
| --- |
| #predict  **def** load\_image**(**img\_path**,** show**=False):**  img **=** image**.**load\_img**(**img\_path**,** target\_size**=(**224**,** 224**))**  img\_tensor **=** image**.**img\_to\_array**(**img**)**  img\_tensor **=** np**.**expand\_dims**(**img\_tensor**,** axis**=**0**)**  img\_tensor **/=** 255.  **if** show**:**  plt**.**imshow**(**img\_tensor**[**0**])**  plt**.**axis**(**'off'**)**  plt**.**show**()**  **return** img\_tensor  img\_list **=** glob**(**'imgs/\*.jpg'**)**  **for** img\_path **in** img\_list**:**  img **=** load\_image**(**img\_path**,** **True)**  result **=** model**.**predict**(**img**)**  index **=** np**.**argmax**(**result**)**  **print(**class\_map**[**index**])** |





## 代码

Mobilenet\_transfer.py:

|  |
| --- |
| **from** keras**.**layers **import** GlobalAveragePooling2D**,** Dense  **from** keras**.**applications**.**mobilenet **import** MobileNet  **from** keras**.**models **import** Model  **def** MobileNetTF**(**class\_num**):**  base\_model **=** MobileNet**(**include\_top**=False)**  x **=** base\_model**.**output  x **=** GlobalAveragePooling2D**()(**x**)**  x **=** Dense**(**1024**,** activation**=**'relu'**)(**x**)**  x **=** Dense**(**512**,** activation**=**'relu'**)(**x**)**  output **=** Dense**(**class\_num**,** activation**=**'softmax'**)(**x**)**  model **=** Model**(**inputs**=**base\_model**.**input**,** outputs**=**output**)**  layer\_num **=** len**(**model**.**layers**)**  **for** layer **in** model**.**layers**[:**layer\_num **-** 4**]:**  layer**.**trainable**=False**  **for** layer **in** model**.**layers**[**layer\_num **-** 4**:]:**  layer**.**trainable**=True**  **return** model |

Fine\_tune.py

|  |
| --- |
| **from** keras**.**models **import** load\_model  **import** numpy **as** np  **import** matplotlib**.**pyplot **as** plt  **from** keras**.**preprocessing **import** image  **from** mobilenet\_transfer **import** MobileNetTF  **from** keras**.**preprocessing**.**image **import** ImageDataGenerator  **from** glob **import** glob  **from** keras**.**callbacks **import** TensorBoard**,** ModelCheckpoint  TRAIN **=** 0  BATCH\_SIZE **=** 32  **def** load\_image**(**img\_path**,** show**=False):**  img **=** image**.**load\_img**(**img\_path**,** target\_size**=(**224**,** 224**))**  img\_tensor **=** image**.**img\_to\_array**(**img**)**  img\_tensor **=** np**.**expand\_dims**(**img\_tensor**,** axis**=**0**)**  img\_tensor **/=** 255.  **if** show**:**  plt**.**imshow**(**img\_tensor**[**0**])**  plt**.**axis**(**'off'**)**  plt**.**show**()**  **return** img\_tensor  **def** get\_class\_from\_generator**(**generator**):**  class\_idx **=** generator**.**class\_indices  idx\_class **=** dict**()**  **for** key **in** class\_idx**.**keys**():**  idx\_class**[**class\_idx**[**key**]]** **=** key  **return** idx\_class  #data prepare  train\_datagen**=**ImageDataGenerator**(**rescale**=**1.**/**255**,**  shear\_range**=**0.2**,**  zoom\_range**=**0.2**,**  horizontal\_flip**=True)**  train\_generator**=**train\_datagen**.**flow\_from\_directory**(**directory**=**'E:/DataSet/tinySet/train'**,**  target\_size**=(**224**,** 224**),**  batch\_size**=**BATCH\_SIZE**,**  class\_mode**=**'categorical'**,**  color\_mode**=**'rgb'**)**  val\_datagen**=**ImageDataGenerator**(**rescale**=**1.**/**255**)**  val\_generator**=**val\_datagen**.**flow\_from\_directory**(**directory**=**'E:/DataSet/tinySet/val'**,**  target\_size**=(**224**,** 224**),**  batch\_size**=**BATCH\_SIZE**,**  class\_mode**=**'categorical'**,**  color\_mode**=**'rgb'**)**  class\_map **=** get\_class\_from\_generator**(**train\_generator**)**  #build model  model **=** MobileNetTF**(**class\_num**=**2**)**  model**.**summary**()**  **if** TRAIN**:**  #train  checkpoint **=** ModelCheckpoint**(**filepath**=**'log/model-{epoch:02d}-{val\_acc:03f}.h5'**)**  tensorboard **=** TensorBoard**(**log\_dir**=**'log'**)**  model**.**compile**(**optimizer**=**'Adam'**,** loss**=**'categorical\_crossentropy'**,** metrics**=[**'accuracy'**])**  step\_size\_train **=** train\_generator**.**n **//** BATCH\_SIZE  validation\_steps **=** val\_generator**.**n **//** BATCH\_SIZE  model**.**fit\_generator**(**generator**=**train\_generator**,**  steps\_per\_epoch**=**step\_size\_train**,**  epochs**=**10**,**  validation\_data**=**val\_generator**,**  validation\_steps**=**validation\_steps**,**  callbacks**=[**checkpoint**,** tensorboard**]**  **)**  model**.**save**(**'dog\_cat.h5'**)**  **else:**  model**=**load\_model**(**'dog\_cat.h5'**)**  #predict  img\_list **=** glob**(**'imgs/\*.jpg'**)**  **for** img\_path **in** img\_list**:**  img **=** load\_image**(**img\_path**,** **True)**  result **=** model**.**predict**(**img**)**  index **=** np**.**argmax**(**result**)**  **print(**class\_map**[**index**])** |