5.5 实践：基于ResNet18网络完成图像分类任务

在本实践中，我们实践一个更通用的图像分类任务。

图像分类（Image Classification）是计算机视觉中的一个基础任务，将图像的语义将不同图像划分到不同类别。很多任务也可以转换为图像分类任务。比如人脸检测就是判断一个区域内是否有人脸，可以看作一个二分类的图像分类任务。

这里，我们使用的计算机视觉领域的经典数据集：CIFAR-10数据集，网络为ResNet18模型，损失函数为交叉熵损失，优化器为Adam优化器，评价指标为准确率。

Adam优化器的介绍参考《神经网络与深度学习》第7.2.4.3节。

5.5.1 数据处理

5.5.1.1 数据集介绍

CIFAR-10数据集包含了10种不同的类别、共60,000张图像，其中每个类别的图像都是6000张，图像大小均为32×32像素。CIFAR-10数据集的示例如 **图5.15** 所示。

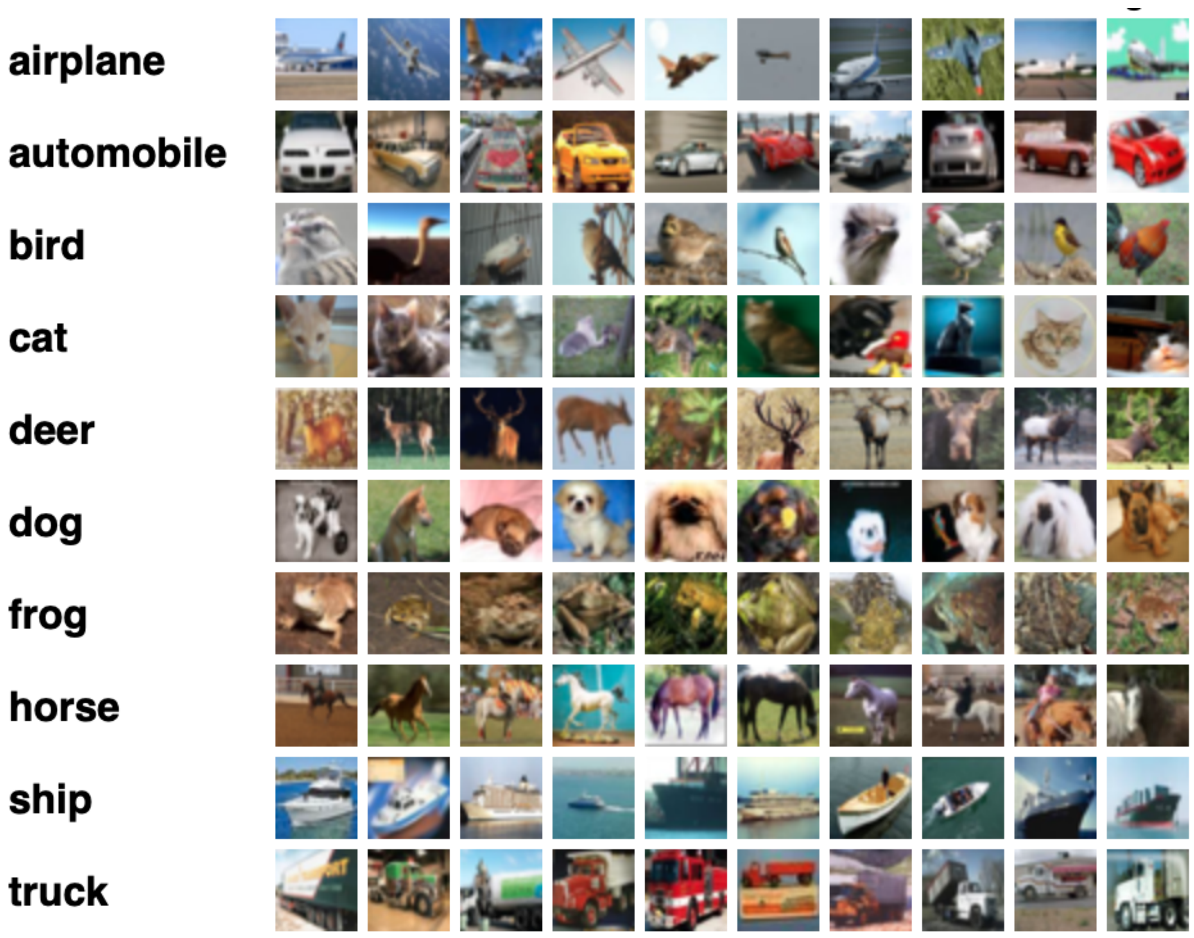


图5.15：CIFAR-10数据集示例

将数据集文件进行解压：

In [1]:
```
# 解压数据集
# 初次运行时将注释取消，以便解压文件
# 如果已经解压过，不需要运行此段代码，否则由于文件已经存在，解压时会报错
!mkdir /home/aistudio/datasets/
!tar -xvf /home/aistudio/data/data9154/cifar-10-python.tar.gz -C /home/aistudio/datasets/
```

```
mkdir: cannot create directory '/home/aistudio/datasets/': File exists
cifar-10-batches-py/
cifar-10-batches-py/data_batch_4
cifar-10-batches-py/readme.html
cifar-10-batches-py/test_batch
cifar-10-batches-py/data_batch_3
cifar-10-batches-py/batches.meta
cifar-10-batches-py/data_batch_2
cifar-10-batches-py/data_batch_5
cifar-10-batches-py/data_batch_1
```

5.5.1.2 数据读取

在本实验中，将原始训练集拆分成了train_set、dev_set两个部分，分别包括40 000条和10 000条样本。将data_batch_1到data_batch_4作为训练集，data_batch_5作为验证集，test_batch作为测试集。 最终的数据集构成为：

- 训练集：40 000条样本。
- 验证集：10 000条样本。

- 测试集：10 000条样本。

读取一个batch数据的代码如下所示：

In [2]:
```
import os
import pickle
import numpy as np

def load_cifar10_batch(folder_path, batch_id=1, mode='train'):
    if mode == 'test':
        file_path = os.path.join(folder_path, 'test_batch')
    else:
        file_path = os.path.join(folder_path, 'data_batch_'+str(batch_id))

    # 加载数据集文件
    with open(file_path, 'rb') as batch_file:
        batch = pickle.load(batch_file, encoding = 'latin1')

    imgs = batch['data'].reshape((len(batch['data']),3,32,32)) / 255.
    labels = batch['labels']

    return np.array(imgs, dtype='float32'), np.array(labels)

imgs_batch, labels_batch = load_cifar10_batch(folder_path='datasets/cifar-10-batches-py',
                                              batch_id=1, mode='train')
```

查看数据的维度：

In [3]:
```
# 打印一下每个batch中X和y的维度
print ("batch of imgs shape: ",imgs_batch.shape, "batch of labels shape: ", labels_batch.shape)
```

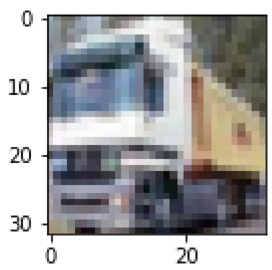
```
batch of imgs shape:  (10000, 3, 32, 32) batch of labels shape:  (10000,)
```

可视化观察其中的一张样本图像和对应的标签，代码如下所示：

In [4]:
```
%matplotlib inline
import matplotlib.pyplot as plt

image, label = imgs_batch[1], labels_batch[1]
print("The label in the picture is {}".format(label))
plt.figure(figsize=(2, 2))
plt.imshow(image.transpose(1,2,0))
plt.savefig('cnn-car.pdf')
```

```
The label in the picture is 9
```



5.5.1.3 构造Dataset类

构造一个CIFAR10Dataset类，其将继承自 `paddle.io.Dataset` 类，可以逐个数据进行处理。代码实现如下：

In [5]:
```
import paddle
import paddle.io as io
from paddle.vision.transforms import Normalize

class CIFAR10Dataset(io.Dataset):
    def __init__(self, folder_path='/home/aistudio/cifar-10-batches-py', mode='train'):
        if mode == 'train':
            # 加载batch1-batch4作为训练集
            self.imgs, self.labels = load_cifar10_batch(folder_path=folder_path, batch_id=1, mode='train')
            for i in range(2, 5):
                imgs_batch, labels_batch = load_cifar10_batch(folder_path=folder_path, batch_id=i, mode='train')
                self.imgs, self.labels = np.concatenate([self.imgs, imgs_batch]), np.concatenate([self.labels, labels_batch])
        elif mode == 'dev':
            # 加载batch5作为验证集
            self.imgs, self.labels = load_cifar10_batch(folder_path=folder_path, batch_id=5, mode='dev')
        elif mode == 'test':
            # 加载测试集
            self.imgs, self.labels = load_cifar10_batch(folder_path=folder_path, mode='test')
        self.transform = Normalize(mean=[0.4914, 0.4822, 0.4465], std=[0.2023, 0.1994, 0.2010], data_format='CHW')

    def __getitem__(self, idx):
        img, label = self.imgs[idx], self.labels[idx]
        img = self.transform(img)
        return img, label

    def __len__(self):
        return len(self.imgs)

paddle.seed(100)
train_dataset = CIFAR10Dataset(folder_path='/home/aistudio/datasets/cifar-10-batches-py', mode='train')
```

```
dev_dataset = CIFAR10Dataset(folder_path='/home/aistudio/datasets/cifar-10-batches-py', mode='dev')
test_dataset = CIFAR10Dataset(folder_path='/home/aistudio/datasets/cifar-10-batches-py', mode='test')
```

5.5.2 模型构建

对于Reset18这种比较经典的图像分类网络，飞桨高层API中都为大家提供了实现好的版本，大家可以不再从头开始实现。这里首先使用飞桨高层API中的Resnet18进行图像分类实验。

In [6]:
```
from paddle.vision.models import resnet18

resnet18_model = resnet18()
```

```
W0628 14:46:17.054970  4638 device_context.cc:447] Please NOTE: device: 0, GPU Compute Capability: 7.0, Driver API Version: 11.2, R
untime API Version: 10.1
W0628 14:46:17.059675  4638 device_context.cc:465] device: 0, cuDNN Version: 7.6.
```

飞桨高层 API是对飞桨API的进一步封装与升级，提供了更加简洁易用的API，进一步提升了飞桨的易学易用性。其中，飞桨高层API封装了以下模块：

1. Model类，支持仅用几行代码完成模型的训练；
2. 图像预处理模块，包含数十种数据处理函数，基本涵盖了常用的数据处理、数据增强方法；
3. 计算机视觉领域和自然语言处理领域的常用模型，包括但不限于mobilenet、resnet、yolov3、cyclegan、bert、transformer、seq2seq等等，同时发布了对应模型的预训练模型，可以直接使用这些模型或者在此基础上完成二次开发。

飞桨高层 API主要包含在`paddle.vision`和`paddle.text`目录中。

5.5.3 模型训练

复用RunnerV3类，实例化RunnerV3类，并传入训练配置。 使用训练集和验证集进行模型训练，共训练30个epoch。 在实验中，保存准确率最高的模型作为最佳模型。代码实现如下：

In [7]:
```
import paddle.nn.functional as F
import paddle.optimizer as opt
from nndl import RunnerV3, metric

# 指定运行设备
use_gpu = True if paddle.get_device().startswith("gpu") else False
if use_gpu:
    paddle.set_device('gpu:0')
# 学习率大小
lr = 0.001
# 批次大小
batch_size = 64
# 加载数据
train_loader = io.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
dev_loader = io.DataLoader(dev_dataset, batch_size=batch_size)
test_loader = io.DataLoader(test_dataset, batch_size=batch_size)
# 定义网络
model = resnet18_model
# 定义优化器，这里使用Adam优化器以及l2正则化策略，相关内容在7.3.3.2和7.6.2中会进行详细介绍
optimizer = opt.Adam(learning_rate=lr, parameters=model.parameters(), weight_decay=0.005)
# 定义损失函数
loss_fn = F.cross_entropy
# 定义评价指标
metric = metric.Accuracy(is_logist=True)
# 实例化RunnerV3
runner = RunnerV3(model, optimizer, loss_fn, metric)
# 启动训练
log_steps = 3000
eval_steps = 3000
runner.train(train_loader, dev_loader, num_epochs=30, log_steps=log_steps,
                eval_steps=eval_steps, save_path="best_model.pdparams")
```

```
/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/paddle/nn/layer/norm.py:653: UserWarning: When training, we now
always track global mean and variance.
  "When training, we now always track global mean and variance.")
```

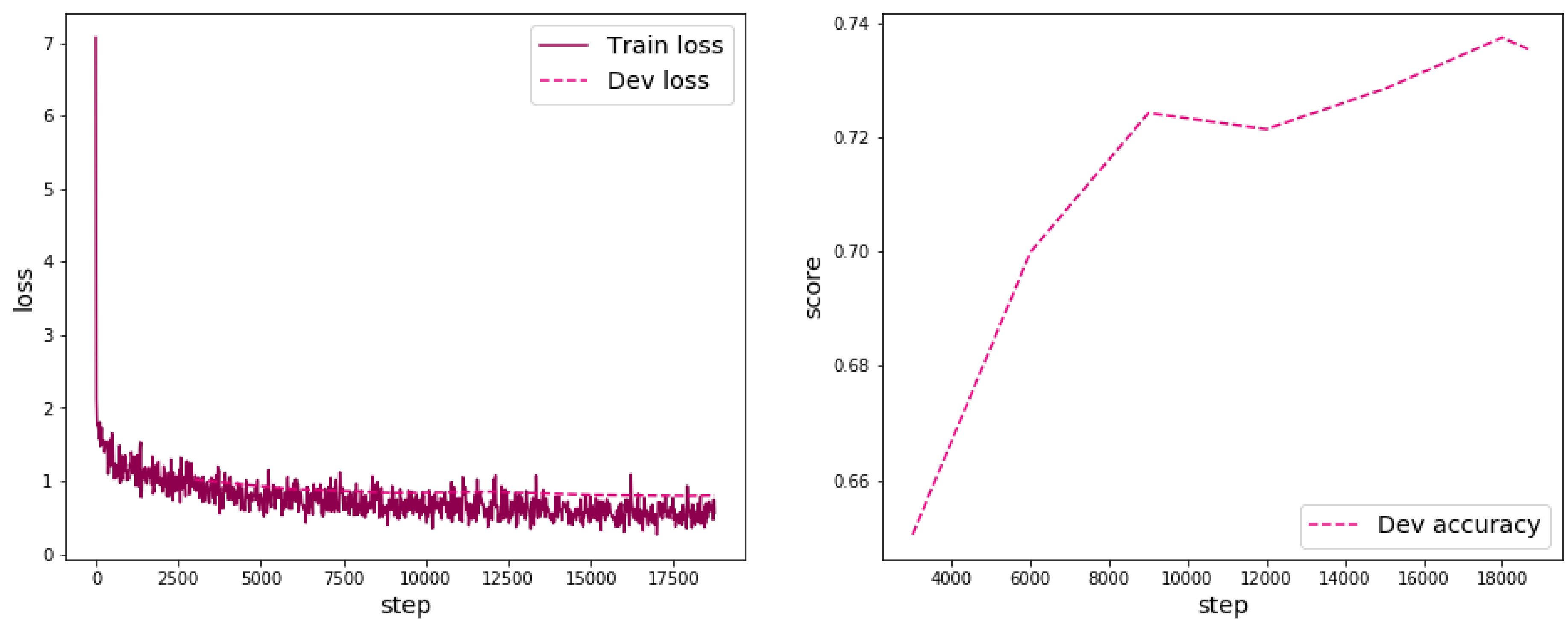
```
[Train] epoch: 0/30, step: 0/18750, loss: 7.07560
[Train] epoch: 4/30, step: 3000/18750, loss: 0.87814
[Evaluate]  dev score: 0.65050, dev loss: 1.01652
[Evaluate] best accuracy performence has been updated: 0.00000 --> 0.65050
[Train] epoch: 9/30, step: 6000/18750, loss: 0.74190
[Evaluate]  dev score: 0.69990, dev loss: 0.88473
[Evaluate] best accuracy performence has been updated: 0.65050 --> 0.69990
[Train] epoch: 14/30, step: 9000/18750, loss: 0.60101
[Evaluate]  dev score: 0.72420, dev loss: 0.83375
[Evaluate] best accuracy performence has been updated: 0.69990 --> 0.72420
[Train] epoch: 19/30, step: 12000/18750, loss: 0.46642
[Evaluate]  dev score: 0.72140, dev loss: 0.84962
[Train] epoch: 24/30, step: 15000/18750, loss: 0.44676
[Evaluate]  dev score: 0.72840, dev loss: 0.80577
[Evaluate] best accuracy performence has been updated: 0.72420 --> 0.72840
[Train] epoch: 28/30, step: 18000/18750, loss: 0.50379
[Evaluate]  dev score: 0.73740, dev loss: 0.79273
[Evaluate] best accuracy performence has been updated: 0.72840 --> 0.73740
[Evaluate]  dev score: 0.73510, dev loss: 0.80100
[Train] Training done!
```

可视化观察训练集与验证集的准确率及损失变化情况。

In [8]:

```python
from nndl import plot_training_loss_acc

plot_training_loss_acc(runner, fig_name='cnn-loss4.pdf')
```



在本实验中，使用了第7章中介绍的Adam优化器进行网络优化，如果使用SGD优化器，会造成过拟合的现象，在验证集上无法得到很好的收敛效果。可以尝试使用第7章中其他优化策略调整训练配置，达到更高的模型精度。

5.5.4 模型评价

使用测试数据对在训练过程中保存的最佳模型进行评价，观察模型在测试集上的准确率以及损失情况。代码实现如下：

In [9]:

```python
# 加载最优模型
runner.load_model('best_model.pdparams')
# 模型评价
score, loss = runner.evaluate(test_loader)
print("[Test] accuracy/loss: {:.4f}/{:.4f}".format(score, loss))
```

```
[Test] accuracy/loss: 0.7279/0.8233
```

5.5.5 模型预测

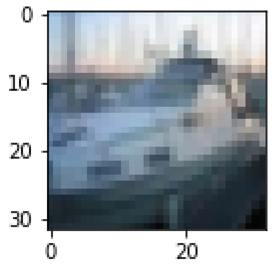
同样地，也可以使用保存好的模型，对测试集中的数据进行模型预测，观察模型效果，具体代码实现如下：

In [10]:

```python
id2label = {0:'airplane', 1:'automobile', 2:'bird', 3:'cat', 4:'deer', 5:'dog', 6:'frog', 7:'horse', 8:'ship', 9:'truck'}
# 获取测试集中的一个batch的数据
X, label_ids = next(test_loader())
logits = runner.predict(X)
# 多分类，使用softmax计算预测概率
pred = F.softmax(logits)
# 获取概率最大的类别
pred_class_id = paddle.argmax(pred[2]).numpy()
label_id = label_ids[2][0].numpy()
pred_class = id2label[pred_class_id[0]]
label = id2label[label_id[0]]
# 输出真实类别与预测类别
print("The true category is {} and the predicted category is {}".format(label, pred_class))
# 可视化图片
plt.figure(figsize=(2, 2))
imgs, labels = load_cifar10_batch(folder_path='/home/aistudio/datasets/cifar-10-batches-py', mode='test')
```

```
plt.imshow(imgs[2].transpose(1,2,0))
plt.savefig('cnn-test-vis.pdf')
```

```
The true category is ship and the predicted category is truck
```



5.5.7 基于自定义的ResNet18网络进行图像分类实验

这里使用自定义的 `Model_ResNet18` 模型进行图像分类实验，观察两者结果是否一致。

5.5.7.1 模型训练

In [11]:
```
import paddle.nn.functional as F
import paddle.optimizer as opt
from nndl import RunnerV3, metric, op

# 指定运行设备
use_gpu = True if paddle.get_device().startswith("gpu") else False
if use_gpu:
    paddle.set_device('gpu:0')
# 学习率大小
lr = 0.001
# 批次大小
batch_size = 64
# 加载数据
train_loader = io.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
dev_loader = io.DataLoader(dev_dataset, batch_size=batch_size)
test_loader = io.DataLoader(test_dataset, batch_size=batch_size)
# 定义网络
model = op.Model_ResNet18(in_channels=3, num_classes=10, use_residual=True)

# 定义优化器，这里使用Adam优化器以及l2正则化策略，相关内容在7.3.3.2和7.6.2中会进行详细介绍
optimizer = opt.Adam(learning_rate=lr, parameters=model.parameters(), weight_decay=0.005)
# 定义损失函数
loss_fn = F.cross_entropy
# 定义评价指标
metric = metric.Accuracy(is_logist=True)
# 实例化RunnerV3
runner = RunnerV3(model, optimizer, loss_fn, metric)
# 启动训练
log_steps = 3000
eval_steps = 3000
runner.train(train_loader, dev_loader, num_epochs=30, log_steps=log_steps,
                eval_steps=eval_steps, save_path="best_model.pdparams")
```

```
[Train] epoch: 0/30, step: 0/18750, loss: 3.24308
[Train] epoch: 4/30, step: 3000/18750, loss: 1.01935
[Evaluate]  dev score: 0.63760, dev loss: 1.02542
[Evaluate] best accuracy performence has been updated: 0.00000 --> 0.63760
[Train] epoch: 9/30, step: 6000/18750, loss: 0.59227
[Evaluate]  dev score: 0.70920, dev loss: 0.84104
[Evaluate] best accuracy performence has been updated: 0.63760 --> 0.70920
[Train] epoch: 14/30, step: 9000/18750, loss: 0.45998
[Evaluate]  dev score: 0.71350, dev loss: 0.84286
[Evaluate] best accuracy performence has been updated: 0.70920 --> 0.71350
[Train] epoch: 19/30, step: 12000/18750, loss: 0.45465
[Evaluate]  dev score: 0.71330, dev loss: 0.85694
[Train] epoch: 24/30, step: 15000/18750, loss: 0.54665
[Evaluate]  dev score: 0.72870, dev loss: 0.82094
[Evaluate] best accuracy performence has been updated: 0.71350 --> 0.72870
[Train] epoch: 28/30, step: 18000/18750, loss: 0.42726
[Evaluate]  dev score: 0.73260, dev loss: 0.81823
[Evaluate] best accuracy performence has been updated: 0.72870 --> 0.73260
[Evaluate]  dev score: 0.73800, dev loss: 0.78462
[Evaluate] best accuracy performence has been updated: 0.73260 --> 0.73800
[Train] Training done!
```

5.5.7.2 模型评价

In [12]:
```
# 加载最优模型
runner.load_model('best_model.pdparams')
# 模型评价
score, loss = runner.evaluate(test_loader)
print("[Test] accuracy/loss: {:.4f}/{:.4f}".format(score, loss))
```

```
[Test] accuracy/loss: 0.7303/0.8039
```

可以看到，使用自定义的Resnet18模型与高层API中的Resnet18模型训练效果可以基本一致。

5.6 实验拓展

- 尝试加深残差网络的层数或使用《神经网络与深度学习》中介绍的其他模型完成基于CIFAR10的图像分类实验，观察是否能够得到更高的精度；
- 尝试使用CIFAR-100进行实验，观察不同模型在不同数据集上的学习效果；