Transfer Learning 之 fast-style-transfer 初体验

前言

这是一篇讲述想要跟风摸鱼，啥也不懂却也可以实现功能的解决方案。

自述

Java新手，也不会python。但是人間讃歌は「勇気」の讃歌ッ！！ 人間のすばらしさは勇気のすばしさ！！。

资源准备

1. fast_neural_style
2. fast-style-transfer
3. 基于macOS搭建一个tensorflow环境
4. Windows 10 搭建 TensorFlow 试玩 fast-style-transfer
5. 百度
6. 手和大脑

测试

路由器没带所以改个源吧。

```
sudo vi ~/.config/pip/pip.conf
```

```
[global]
#index-url = https://pypi.tuna.tsinghua.edu.cn/simple
index-url = https://mirrors.aliyun.com/pypi/simple
```

```
git clone https://github.com/lengstrom/fast-style-transfer

cd fast-style-transfer/

pyenv activate v370env

python evaluate.py --checkpoint model/udnie.ckpt --in-path xxx.jpg --out-path xxx/
```

魔改Api（python）

```
mkdir neural_style

cd ..

git clone https://github.com/pytorch/examples/

cd examples/

cp fast_neural_style/neural_style/*.py ../fast-style-transfer/neural_style
```

python restful api ?

```
pip install flask
```

python create class?

修改后的evaluate.py

```
from __future__ import print_function
import sys

sys.path.insert(0, 'src')
import  numpy as np, os
import tensorflow as tf
from src.utils import save_img, get_img, exists, list_files
from src import transform
from collections import defaultdict
from moviepy.video.io.VideoFileClip import VideoFileClip
import moviepy.video.io.ffmpeg_writer as ffmpeg_writer

BATCH_SIZE = 4
DEVICE = '/gpu:0'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

def ffwd_video(path_in, path_out, checkpoint_dir, device_t='/gpu:0', batch_size=4):
    video_clip = VideoFileClip(path_in, audio=False)
    video_writer = ffmpeg_writer.FFMPEG_VideoWriter(path_out, video_clip.size, video_clip.fps,
odec="libx264",
                                                    preset="medium", bitrate="2000k",
                                                    audiofile=path_in, threads=None,
                                                    ffmpeg_params=None)

    g = tf.Graph()
    soft_config = tf.compat.v1.ConfigProto(allow_soft_placement=True)
    soft_config.gpu_options.allow_growth = True
    with g.as_default(), g.device(device_t), \
            tf.compat.v1.Session(config=soft_config) as sess:
        batch_shape = (batch_size, video_clip.size[1], video_clip.size[0], 3)
        img_placeholder = tf.compat.v1.placeholder(tf.float32, shape=batch_shape,
                                         name='img_placeholder')

        preds = transform.net(img_placeholder)
        saver = tf.train.Saver()
        if os.path.isdir(checkpoint_dir):
            ckpt = tf.train.get_checkpoint_state(checkpoint_dir)
            if ckpt and ckpt.model_checkpoint_path:
                saver.restore(sess, ckpt.model_checkpoint_path)
            else:
                raise Exception("No checkpoint found...")
        else:
            saver.restore(sess, checkpoint_dir)

        X = np.zeros(batch_shape, dtype=np.float32)

        def style_and_write(count):
            for i in range(count, batch_size):
                X[i] = X[count - 1]  # Use last frame to fill X
            _preds = sess.run(preds, feed_dict={img_placeholder: X})
            for i in range(0, count):
                video_writer.write_frame(np.clip(_preds[i], 0, 255).astype(np.uint8))

        frame_count = 0  # The frame count that written to X
        for frame in video_clip.iter_frames():
            X[frame_count] = frame
            frame_count += 1
            if frame_count == batch_size:
                style_and_write(frame_count)
                frame_count = 0

        if frame_count != 0:
            style_and_write(frame_count)

        video_writer.close()


# get img_shape
def ffwd(data_in, paths_out, checkpoint_dir, device_t='/gpu:0', batch_size=4):
    assert len(paths_out) > 0
    is_paths = type(data_in[0]) == str
    if is_paths:
        assert len(data_in) == len(paths_out)
        img_shape = get_img(data_in[0]).shape
    else:
        assert data_in.size[0] == len(paths_out)
        img_shape = X[0].shape

    g = tf.Graph()
    batch_size = min(len(paths_out), batch_size)
    soft_config = tf.compat.v1.ConfigProto(allow_soft_placement=True)
    soft_config.gpu_options.allow_growth = True
    with g.as_default(), g.device(device_t), \
            tf.compat.v1.Session(config=soft_config) as sess:
        batch_shape = (batch_size,) + img_shape
        img_placeholder = tf.compat.v1.placeholder(tf.float32, shape=batch_shape,
                                                   name='img_placeholder')

        preds = transform.net(img_placeholder)
        saver = tf.compat.v1.train.Saver()
        if os.path.isdir(checkpoint_dir):
            ckpt = tf.train.get_checkpoint_state(checkpoint_dir)
            if ckpt and ckpt.model_checkpoint_path:
                saver.restore(sess, ckpt.model_checkpoint_path)
            else:
                raise Exception("No checkpoint found...")
        else:
            saver.restore(sess, checkpoint_dir)

        num_iters = int(len(paths_out)/batch_size)
        for i in range(num_iters):
            pos = i * batch_size
            curr_batch_out = paths_out[pos:pos+batch_size]
            if is_paths:
                curr_batch_in = data_in[pos:pos+batch_size]
                X = np.zeros(batch_shape, dtype=np.float32)
                for j, path_in in enumerate(curr_batch_in):
                    img = get_img(path_in)
                    assert img.shape == img_shape, \
                        'Images have different dimensions. ' +  \
                        'Resize images or use --allow-different-dimensions.'
                    X[j] = img
            else:
                X = data_in[pos:pos+batch_size]

            _preds = sess.run(preds, feed_dict={img_placeholder:X})
            for j, path_out in enumerate(curr_batch_out):
                save_img(path_out, _preds[j])

        remaining_in = data_in[num_iters*batch_size:]
        remaining_out = paths_out[num_iters*batch_size:]
    if len(remaining_in) > 0:
        ffwd(remaining_in, remaining_out, checkpoint_dir,
            device_t=device_t, batch_size=1)

def ffwd_to_img(in_path, out_path, checkpoint_dir, device='/cpu:0'):
    paths_in, paths_out = [in_path], [out_path]
    ffwd(paths_in, paths_out, checkpoint_dir, batch_size=1, device_t=device)

def ffwd_different_dimensions(in_path, out_path, checkpoint_dir,
            device_t=DEVICE, batch_size=4):
    in_path_of_shape = defaultdict(list)
    out_path_of_shape = defaultdict(list)
    for i in range(len(in_path)):
        in_image = in_path[i]
        out_image = out_path[i]
        shape = "%dx%dx%d" % get_img(in_image).shape
        in_path_of_shape[shape].append(in_image)
        out_path_of_shape[shape].append(out_image)
    for shape in in_path_of_shape:
        print('Processing images of shape %s' % shape)
        ffwd(in_path_of_shape[shape], out_path_of_shape[shape],
            checkpoint_dir, device_t, batch_size)

def check_opts(opts):
    exists(opts.checkpoint_dir, 'Checkpoint not found!')
    exists(opts.in_path, 'In path not found!')
    if os.path.isdir(opts.out_path):
        exists(opts.out_path, 'out dir not found!')
        assert opts.batch_size > 0
# 对象传参吧?
class Parser(object):
    def __init__(self):
        self._in_path = None
        self._out_path = None
        self._checkpoint_dir = None
        self._device=DEVICE
        self._batch_size=BATCH_SIZE

    @property
    def batch_size(self):
        return self._batch_size
    @property
    def device(self):
        return self._device
    @property
    def in_path(self):
        return self._in_path

    @property
    def out_path(self):
        return self._out_path

    @property
    def checkpoint_dir(self):
        return self._checkpoint_dir

    @in_path.setter
    def in_path(self, in_path):
        self._in_path=in_path

    @out_path.setter
    def out_path(self, out_path):
        self._out_path=out_path

    @checkpoint_dir.setter
    def checkpoint_dir(self, checkpoint_dir):
        self._checkpoint_dir=checkpoint_dir

    @in_path.deleter
    def in_path(self):
        del self._in_path

    @out_path.deleter
    def out_path(self):
        del self._out_path

    @checkpoint_dir.deleter
    def checkpoint_dir(self):
        del self._checkpoint_dir
def main(opts):
    check_opts(opts)
    if not os.path.isdir(opts.in_path):
        if os.path.exists(opts.out_path) and os.path.isdir(opts.out_path):
            out_path = \
                    os.path.join(opts.out_path,os.path.basename(opts.in_path))
        else:
            out_path = opts.out_path

        ffwd_to_img(opts.in_path, out_path, opts.checkpoint_dir,
                    device=opts.device)
    else:
        files = list_files(opts.in_path)
        full_in = [os.path.join(opts.in_path,x) for x in files]
        full_out = [os.path.join(opts.out_path,x) for x in files]
        if opts.allow_different_dimensions:
            ffwd_different_dimensions(full_in, full_out, opts.checkpoint_dir,
                    device_t=opts.device, batch_size=opts.batch_size)
        else :
            ffwd(full_in, full_out, opts.checkpoint_dir, device_t=opts.device,
                    batch_size=opts.batch_size)
```

什么是 argparse.ArgumentParser ?

修改后的 neural_style.py

```
import argparse
import os
import sys
import re

import torch
from torchvision import transforms
import torch.onnx

import neural_style.utils as utils
from neural_style.transformer_net import TransformerNet



def check_paths(args):
    try:
        if not os.path.exists(args.save_model_dir):
            os.makedirs(args.save_model_dir)
        if args.checkpoint_model_dir is not None and not (os.path.exists(args.checkpoint_model
dir)):
            os.makedirs(args.checkpoint_model_dir)
    except OSError as e:
        print(e)
        sys.exit(1)


def stylize(args):
    device = torch.device("cuda" if args.cuda else "cpu")

    content_image = utils.load_image(args.content_image, scale=args.content_scale)
    content_transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Lambda(lambda x: x.mul(255))
    ])
    content_image = content_transform(content_image)
    content_image = content_image.unsqueeze(0).to(device)

    if args.model.endswith(".onnx"):
        output = stylize_onnx_caffe2(content_image, args)
    else:
        with torch.no_grad():
            style_model = TransformerNet()
            state_dict = torch.load(args.model)
            # remove saved deprecated running_* keys in InstanceNorm from the checkpoint
            for k in list(state_dict.keys()):
                if re.search(r'in\d+\.running_(mean|var)$', k):
                    del state_dict[k]
            style_model.load_state_dict(state_dict)
            style_model.to(device)
            if args.export_onnx:
                assert args.export_onnx.endswith(".onnx"), "Export model file should end with .onnx

                output = torch.onnx._export(style_model, content_image, args.export_onnx).cpu()
            else:
                output = style_model(content_image).cpu()
    utils.save_image(args.output_image, output[0])


def stylize_onnx_caffe2(content_image, args):
    assert not args.export_onnx

    import onnx
    import onnx_caffe2.backend

    model = onnx.load(args.model)

    prepared_backend = onnx_caffe2.backend.prepare(model, device='CUDA' if args.cuda else
CPU')
    inp = {model.graph.input[0].name: content_image.numpy()}
    c2_out = prepared_backend.run(inp)[0]

    return torch.from_numpy(c2_out)


def main(content_image,model,output_image):
    args = argparse.ArgumentParser(description="parser for fast-neural-style").parse_args()
    args.content_image=content_image
    args.model=model
    args.output_image=output_image
    args.content_scale=None
    args.export_onnx=None
    args.cuda=0
    stylize(args)
```

简单的RESTful实现

```
pip install jinja2
```

新建 app.py

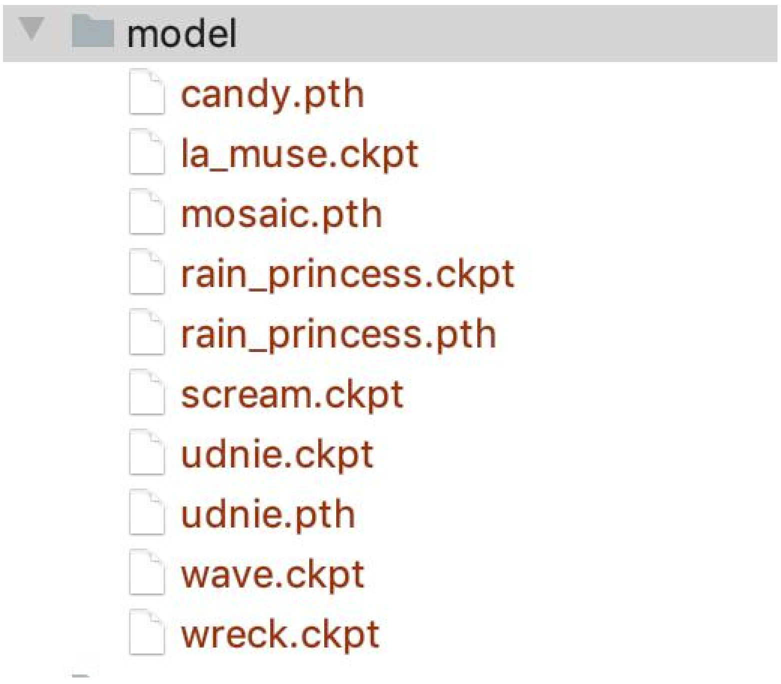
```
from flask import Flask, request, jsonify
from werkzeug.utils import secure_filename
import time
import os
import evaluateapi
import neural_style.neural_style as nn

app = Flask(__name__)
UPLOAD_FOLDER = 'upload'
UPLOAD_PATH = '/Users/apple/PycharmProjects/fast-style-transfer/upload/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['JSON_AS_ASCII'] = False
basedir = os.path.abspath(os.path.dirname(__file__))
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'JPG', 'PNG', 'jpeg', 'JPEG', 'gif'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS

@app.route('/api/upload', methods=['POST'], strict_slashes=False)
def api_upload():
    file_dir = os.path.join(basedir, app.config['UPLOAD_FOLDER'])
    type = request.form['type']
    if not os.path.exists(file_dir):
        os.makedirs(file_dir)
    f = request.files['myfile']
    if f and allowed_file(f.filename):
        fname = secure_filename(f.filename)
        print(fname)
        ext = fname.rsplit('.', 1)[1]
        unix_time = int(time.time())
        new_filename = str(unix_time) + '.' + ext
        f.save(os.path.join(file_dir, new_filename))
        file_path = UPLOAD_PATH + new_filename
        if type.endswith('.ckpt'):
            opts = evaluateapi.Parser()
            opts.in_path = file_path
            opts.out_path = file_path
            opts.checkpoint_dir = type
            evaluateapi.main(opts)
        else:
            content_image = file_path
            model = type
            output_image = file_path
            nn.main(content_image, model, output_image)
        print(file_path)
        return jsonify({"code": 0,"filePath": file_path})
    else:
        return jsonify({"code": 1001, "errmsg": "上传失败"})
if __name__ == "__main__":
    app.run(debug=True)
```

客户端（Java）

swagger 怎么写？

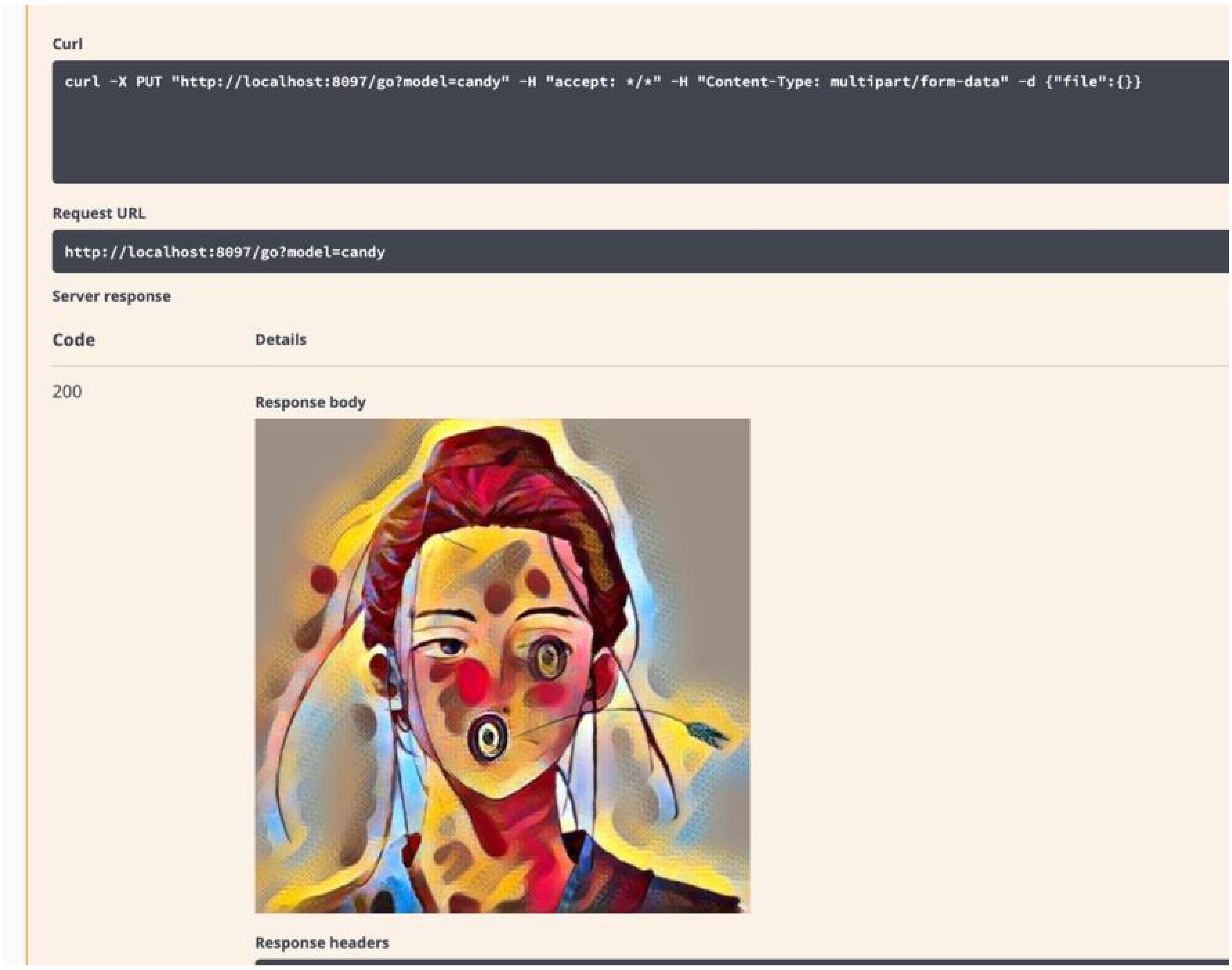
```
/**
 * @author：czx.me 2020/3/23
 */
@Slf4j
@Api(tags = "fast-style-transfer（算力有限，图片大小限制在1M）", value = "fast-style-transfe
（算力有限，图片大小限制在1M）")
@Controller
public class TransferController {
    @PutMapping(value = "go", consumes = MediaType.MULTIPART_FORM_DATA_VALUE)
    @ApiOperation(value = "transfer", notes = "fast-style-transfer")
    @ApiImplicitParam(name = "model",
            defaultValue = "la_muse",
            value = "风格模型(la_muse rain_princess scream udnie wave wreck candy mosaic udnie
pth rain_princess_pth 十种)任选其一",
            dataType = "String", paramType = "query")
    public void transferStart(String model,
                              @ApiParam(name = "file", value = "图片文件", required = true) @ModelAttr
bute("file") MultipartFile file,
                              HttpServletResponse response,
                              HttpServletRequest request) throws IOException {
        File copyFile = null;
        File inFile = null;
        BufferedOutputStream out = null;
        InputStream input = null;
        String url = "http://127.0.0.1:5000/api/upload";
        try {
            String fileName = file.getOriginalFilename();
            //模型枚举 关键字+路径的组合
            model = Model.fromTypeName(model);
            if (null == model || "".equals(model)) {
                log.error("没有 [{}] 这个类型？", model);
                throw new Exception();
            }
            inFile = new File("./temp/" + fileName);
            FileUtils.writeByteArrayToFile(inFile, file.getBytes());
            // 一个封装的HttpURLConnection
            String json = Request.post(url)
                    .connectTimeout(50000)
                    .readTimeout(50000)
                    .contentType("multipart/form-data")
                    .part("myfile", fileName, "multipart/form-data", inFile)
                    .part("type", model)
                    .body();
            JSONObject jsonObject = JSON.parseObject(json);
            String filePath = jsonObject.getString("filePath");
            String code = jsonObject.getString("code");
            if ("0".equals(code)) {
                copyFile = new File(filePath);
                input = new FileInputStream(copyFile);
                String contentType = request.getServletContext().getMimeType(fileName);
                response.setContentType(contentType);
                out = new BufferedOutputStream(response.getOutputStream());
                IOUtils.copy(input, out);
                out.flush();
                input.close();
            } else {
                log.error("好像算不过来。");
                throw new Exception();
            }
        } catch (Exception e) {
            response.sendRedirect("404");
        } finally {
            FileUtils.deleteQuietly(copyFile);
            FileUtils.deleteQuietly(inFile);
        }
    }
}
```

目前拥有的模型



```
model
    candy.pth
    la_muse.ckpt
    mosaic.pth
    rain_princess.ckpt
    rain_princess.pth
    scream.ckpt
    udnie.ckpt
    udnie.pth
    wave.ckpt
    wreck.ckpt
```

大功告成?



Curl

```
curl -X PUT "http://localhost:8097/go?model=candy" -H "accept: */*" -H "Content-Type: multipart/form-data" -d {"file":{}}
```

Request URL

```
http://localhost:8097/go?model=candy
```

Server response

Code	Details
200	Response body

Response headers

限时体验

现已部署在阿里云轻量级服务器上

<a href="https://www.czx.me/" target="_blank">戳这里</a>