
import streamlit as st
import requests
import torch
from torchvision import transforms
from resnet import ResNetGenerator
from savedb import truncate, save_to_temp_db, update_prod_db, connect_to_db
from urllib.parse import urlparse
from PIL import Image, UnidentifiedImageError
from io import BytesIO
from pdf2image import convert_from_bytes
import base64
st.set_option('deprecation.showfileUploaderEncoding', False)
st.title('Zebrate app')
st.header('neural network that turns a horse into a zebra')
FILE_TYPES = ['png', 'jpg', 'jpeg', 'pdf']
URL = (
'https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Flifeglobe.net%2Fmedia%2Fentry%2F6445%2F1a.jpg'
)
horse_table = 'horse_files'
zebra_table = 'zebra_files'
author = """
---
made with:
* [Streamlit](https://www.streamlit.io/)
* [Book](https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf)
* [PyTorch](https://pytorch.org/)
* [CycleGAN](https://github.com/keras-team/keras-io/blob/master/examples/generative/cyclegan.py)
* [ResNet](https://www.res.net/)
* [weights for model](https://github.com/deep-learning-with-pytorch/dlwpt-code/blob/master/data/p1ch2/horse2zebra_0.4.0.pth)
* [and this dataset](http://mng.bz/8pKP)
by [Dmitry Kosarevsky](https://github.com/dKosarevsky) for [TaDS labs](https://networking-labs.ru/) in [BMSTU](https://bmstu.ru)
"""
def uploader(file):...
def prepare_model():...
def img_to_bin(im):...
def generate_zebra(net_G, preprocess, user_img, user_url, base_url, db_conn):...
def validate_url(url):...
def main():...
if __name__ == '__main__':
calls
__name__ == '__main__'
st.set_option
st.title
st.header
main()
calls
main
uploader
"""
function for upload image from user
return file object
"""
show_file = st.sidebar.empty()
if not file:
calls
not file
(not not file)
st.sidebar.empty
show_file.info('valid file extension: ' + ', '.join(FILE_TYPES))
return False
return file
calls
show_file.info
prepare_model
"""
function for prepare generative model
return model and preprocess
"""
netG = ResNetGenerator()
model_path = './horse2zebra_0.4.0.pth'
model_data = torch.load(model_path)
netG.load_state_dict(model_data)
netG.eval()
preprocess = transforms.Compose([transforms.Resize(256), transforms.ToTensor()]
)
return netG, preprocess
calls
ResNetGenerator
torch.load
netG.load_state_dict
netG.eval
transforms.Compose
img_to_bin
"""
function for encode image ti binary
return binary object
"""
buffered = BytesIO()
im.save(buffered, format='PNG')
bin_im = base64.b64encode(buffered.getvalue())
return bin_im
calls
BytesIO
im.save
base64.b64encode
generate_zebra
"""
function for generate zebra from horse-tensor
return image of zebra
"""
if user_img:
user_img
(not user_img)
img = Image.open(user_img)
img = convert_from_bytes(user_img.read(), fmt='jpeg')[0]
response = requests.get(user_url if user_url else base_url)
img = Image.open(BytesIO(response.content))
st.write(
'Something went wrong ... Try another link, or upload an image from your local device'
)
st.stop()
calls
calls
Image.open
convert_from_bytes
st.write(
'We take a random horse :racehorse: (you can load your own in the menu on the left)'
)
img_bin = img_to_bin(img)
truncate(horse_table, db_conn)
save_to_temp_db(img_bin, horse_table, db_conn)
update_prod_db(horse_table, db_conn)
st.image(img)
img_t = preprocess(img)
batch_t = torch.unsqueeze(img_t, 0)
batch_out = net_G(batch_t)
out_t = (batch_out.data.squeeze() + 1.0) / 2.0
out_img = transforms.ToPILImage()(out_t)
st.write(
"We drive the horse into tensor ... it's not a stall :grin:, it's such a multidimensional matrix"
)
if st.checkbox('Show me the horse tensor'):
requests.get
Image.open
st.write
st.stop
calls
st.checkbox('Show me the horse tensor')
(not st.checkbox('Show me the horse tensor'))
st.write
img_to_bin
truncate
save_to_temp_db
update_prod_db
st.image
preprocess
torch.unsqueeze
net_G
batch_out.data.squeeze
Call
st.write
st.code(img_t)
calls
st.code
st.write('Then there is a battle of generative adversarial networks ...')
return out_img
calls
st.write
validate_url
result = urlparse(url)
if all([result.scheme, result.netloc]):
calls
all([result.scheme, result.netloc])
(not all([result.scheme, result.netloc]))
urlparse
return url
if not url:
not url
(not not url)
return False
st.sidebar.markdown(
"it doesn't look like a picture link",
unsafe_allow_html=True)
return False
calls
st.sidebar.markdown
main
horse_url = validate_url(st.sidebar.text_input(
'Put the link to the horse picture here: '))
net, preproc = prepare_model()
img_file = uploader(st.sidebar.file_uploader('Upload your horse image:',
type=FILE_TYPES))
db_connection = connect_to_db()
zebra = generate_zebra(net, preproc, img_file, horse_url, URL, db_connection)
img_bin = img_to_bin(zebra)
truncate(zebra_table, db_connection)
save_to_temp_db(img_bin, zebra_table, db_connection)
update_prod_db(zebra_table, db_connection)
st.write('... and the horse turns into a zebra')
st.image(zebra)
st.sidebar.markdown(author)
calls
validate_url
prepare_model
uploader
connect_to_db
generate_zebra
img_to_bin
truncate
save_to_temp_db
update_prod_db
st.write
st.image
st.sidebar.markdown

zebrate.py