**OpenUVF**

***Copyright Southern Company 2019***

Windows 10 Setup Instructions

**Overview**

OpenUVF utilizes Python, Tensorflow, and MATLAB to analyze UVF images and automatically detect cracks. Presently, MATLAB is essential for most of the preprocessing work and Python with Google’s Tensorflow package is necessary for the machine learning implementation of automatic crack detection. For faster crack detection, or if you intend to further train the crack detection model, TensorFlow can utilize your computers GPU (if you have one) to accelerate the process.

These instructions are adapted from those provided [here](https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html) by Lyudmil Vladimirov

**GPU Computing Setup (TensorFlow GPU Only)**

1. Install CUDA Toolkit v9.0, available at [Nvidia’s Website](https://developer.nvidia.com/cuda-90-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exenetwork)
   1. Select:
      1. Windows
      2. X86\_64
      3. 10
      4. Either exe (network) or exe (local) depending on your preference
   2. Download the Base Installer and all Patches
   3. Install the Base Installer
   4. Restart your Computer
   5. Install the patches in numerical order
2. Download cuDNN, available on [Nvidia’s Website](https://developer.nvidia.com/rdp/cudnn-download)
   1. You must create a user profile and log in
   2. Agree to the terms
   3. At the bottom of the page, select “Archived cuDNN Releases”
   4. Scroll down to and select “Download cuDNN v7.0.5 (Dec 5, 2017), for CUDA 9.0
   5. Select cuDNN v7.0.5 Library for Windows 10
   6. Extract the cuDNN zip file
   7. Navigate to your CUDA v9.0 folder (*Default: C:\Program Files\Nvidia GPU Computing Toolkit\CUDA\v9.0)*
      1. From the extracted cuDNN folder, copy the cuda folder to here.
   8. Add the Nvidia tools to your Path Variable:
      1. From the start menu search “environment variables”
      2. Select “Edit the system environment variables”
      3. Select “Environment Variables” at the bottom of the window
      4. In the “System variables” panel, scroll down and select the Path system variable, choosing edit:
      5. Add the following paths by clicking new
         1. <Install Path>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin
         2. <Install Path>\NVIDIA GPU Computing Toolkit\CUDA\cuda\bin
         3. <Install Path>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\libnvvp
         4. <Install Path>\NVIDIA GPU Computing Toolkit\CUDA\v9.0\extras\CUPTI\libx64
         5. Where <Install Path> by default should be: C:\Program Files

**Python Setup**

1. Go to <https://www.anaconda.com/download/> and download the Python 3.X version
   1. 64 bit version
2. Install *Anaconda* as you would any other software
   1. Destination folder: For later access, and to avoid potential issues, it is best to install at **C:\Anaconda3**
   2. Reference the Anaconda installation guide for detailed installation instructions: <http://docs.anaconda.com/anaconda/install/windows/>

**TensorFlow Environment Setup using Conda**

Open UVF utilizes conda to manage packages. This allows you to have several independent environments with their own sets of packages. Here it namely allows having two independent tensorflow setups: one for CPU computing and one for GPU computing. If you intend to retrain an OpenUVF model, a GPU is practically essential. For running crack detection with one of the light-weight models, a CPU is sufficient. Tensorflow’s cpu installation also encounters fewer issues than the GPU variant, so it is nice to have as a backup.

*In the below instructions, we refer to the environment as tensorflowGPU, but feel free to use alternative names. The installation from here on is CPU/GPU independent.*

1. From the start menu, open the Anaconda Prompt
2. Using conda create a tensorflow environment using this command in the prompt:
   1. conda create -n tensorflowGPU pip python=3.6
3. Activate the environment using:
   1. activate tensorflowGPU
4. Install TensorFlow using:
   1. pip install –ignore-installed –upgrade tensorflow-gpu
5. For assurance, you can test your installation by:
6. Install git using the installer available [here](https://git-scm.com/download/win). **Note, automatic download at link.**
7. Install Visual C++ Build Tools available [here](https://go.microsoft.com/fwlink/?LinkId=691126). **Note, automatic download at link.**
8. Install Required packages:
   1. Pillow: conda install pillow
   2. Lxml: conda install lxml
   3. Jupyter: conda install jupyter
   4. Matplotlib: conda install matplotlib
   5. Opencv: conda install opencv
   6. Cython: conda install cython
   7. Contextlib2: conda install contextlib2
   8. Pycocotools: pip install git+https://github.com/philferriere/cocoapi.git#egg=pycocotools^&subdirectory=PythonAPI
   9. Pandas: pip install pandas

**OpenUVF Setup**

1. If you haven’t already, download the OpenUVF repository from [github](https://github.com/williamhobbs/OpenUVF) (basic) or dropbox (complete)
2. Extract the OpenUVF directory wherever you prefer (*default assumed to be C:\Users\<username>\Documents\OpenUVF*)
3. Setup/Install the Tensorflow Object Detection API
   1. Open the Anaconda prompt from the start menu
   2. Change directories into the OpenUVF core using:
      1. cd <install path>\core
   3. Run these commands to install the API:
      1. run python setup.py build
      2. run python setup.py install
4. From core, copy the contents of nets.
5. Navigate to your Anaconda3 folder, going to:
   1. <install path>\envs\tensorflowGPU\Lib\site-packages\object\_detection-0.1-py3.6.egg
6. Paste the contents of nets there.
7. Navigate (cd) back to the OpenUVF folder.
8. If not active, activate your tensorflow environment using:
   1. activate tensorflowGPU
9. Open jupyter using:
   1. jupyter notebook
10. Jupyter will open in your default web browser and show the contents of the OpenUVF folder.
11. Open the detection.ipynb file
12. From “Cell,” select “Run All”
13. If everything is properly setup, the script will run and output an image in the outputs folder.

Install contextlib2 <https://pypi.org/project/contextlib2/>

Install pycocotools using pip install pycocotools

Install Microsoft Visual C++ Build tools, available here: <https://visualstudio.microsoft.com/visual-cpp-build-tools/>

**Useful Tutorials/References**

**Creating a Frozen Detection Model**

**Debugging**

1. **Tensorflow failed to load native tensorflow runtime (Import Error: DLL load failed: The specified module could not be found)**
   1. Problem: When using your tensorflowGPU environment, importing tensorflow causes this. This is caused by windows now being able to find the cuDNN DLL in your CUDA/v9.0 folder.
   2. Solution:
      1. Check that the cuDNN bin folder (CUDA\v9.0\cuda\bin) was saved when you added it to your system path variable. If it appears, restart, check again, and add it if it is missing.
      2. The version of the CUDA toolkit and the cuDNN you installed may not be supported. Try reverting to older versions or progressing to newer versions.
2. **ModuleNotFoundError: No module named ‘object detection’:**
   1. Open anaconda prompt
   2. Activate your tensorflow environment
   3. cd into Tensorflow/research
   4. run python setup.py build
   5. run python setup.py install
3. **ModuleNotFoundError: No module named ‘nets’:** 
   1. Problem: The system cannot find the code from nets that is referenced in the code.
   2. Solution:
      1. Navigate to your TensorFlow folder
      2. Go to models\slim
      3. Copy all of the contents
      4. Navigate to your Anaconda3 folder
      5. Go to envs\tensorflowXXX\Lib\site-packages\object\_detection-0.1-py3.6.egg and paste the contents of the slim folder that you copied.
      6. Rerun the script
4. **Unknown Meta Architecture: None**
   1. Use your backup pipeline config to resave pipeline.config
5. **cuDNN failed to initialize**
   1. Go to task manager, find Nvidia Container Background Process and end it.

<https://github.com/tensorflow/models/issues/2031>

Copy contents of Nets to

Install git using Github desktop or [SOME ALTERNTIVE]. Add git to path using

References

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

<https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html#exporting-a-trained-inference-graph>

<https://pythonprogramming.net/creating-tfrecord-files-tensorflow-object-detection-api-tutorial/>

<https://github.com/tensorflow/models/tree/master/research/object_detection>

<https://github.com/philferriere/cocoapi>

<https://pypi.org/project/pycocotools/>

<https://github.com/protocolbuffers/protobuf>

[**https://visualstudio.microsoft.com/visual-cpp-build-tools/**](https://visualstudio.microsoft.com/visual-cpp-build-tools/)

[**https://medium.com/@rohitrpatil/how-to-use-tensorflow-object-detection-api-on-windows-102ec8097699**](https://medium.com/@rohitrpatil/how-to-use-tensorflow-object-detection-api-on-windows-102ec8097699)