TUT – CTC TRAINING WITH PYCHARM

I. Chuẩn bị data cho training:

1. **File âm thanh:** Tiến hành thu âm bằng máy điện thoại hoặc máy chuyên dụng các từ cần training: Ví dụ từ "Thầy Thơ", "Ja Long", "Hoàng Hùng"…

   Các file thu âm thường có định dạng . wav

2. **Labeling**: chuẩn bị các file nhãn tương ứng với các file thu âm. Nhãn là các file có đuôi .txt, tên các file này giống với tên file âm thanh tương ứng. Ví dụ: ta có file âm thanh là đặt tên là file1. wav thì file nhãn là file1.txt. Nội dung của file .txt là chuỗi được viết theo kiểu telex, ví dụ: từ "THẦY THƠ" được viết là "THAAFY THOW". Sẽ có một hàm chuyển đổi kiểu telex thành unicode.
3. **Các file âm thanh được đặt vào thư mục sau:**

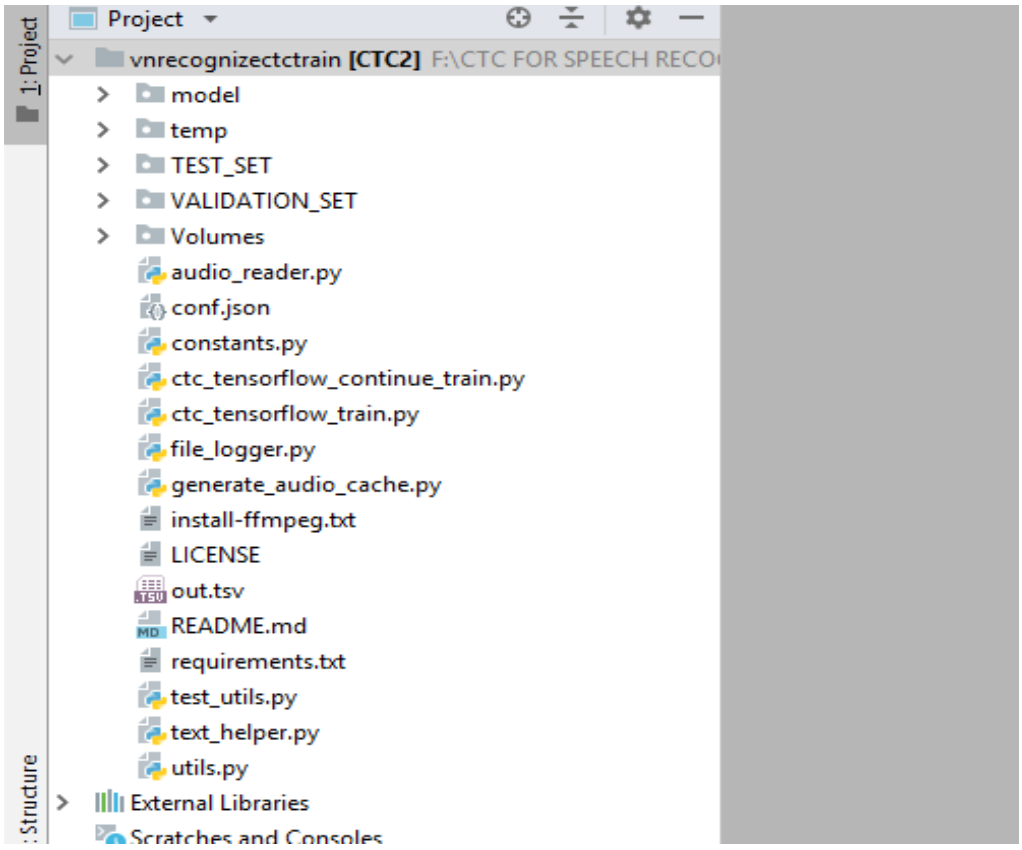
   \Volumes\Transcend\VCTK-Corpus\vctk-p225\wav48\p225

4. **Các file *.txt được đặt vào thư mục:**

   \Volumes\Transcend\VCTK-Corpus\vctk-p225\txt\p225

   Chúng ta có thể cấu hình lại thư mục lưu trữ các file này.

II. Training:



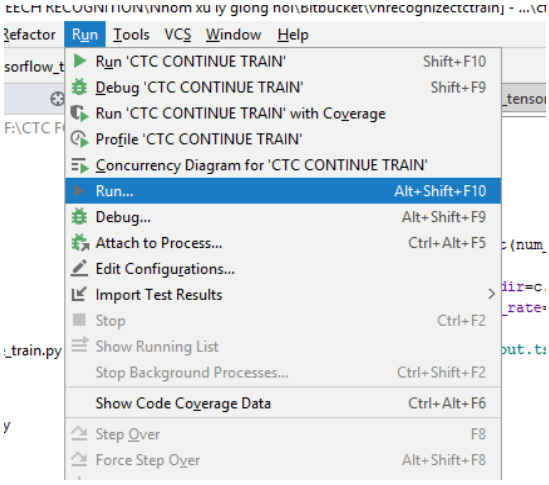
Hình trên đây là cấu trúc các file trong dự án.

- **Model**: là thư mục sẽ chứa model sau khi đã train
- **Temp**: là thư mực chứa các file cache của các file audio, khi training thì sẽ đọc các file cache này để train.
- **TEST_SET**: chứa tập các file âm thanh chưa được train (thường là 20-30% của tổng các file data có được), dùng để test model vừa train xong.
- **VALIDATION_SET**: chứa tập các file đã được train, dùng để test model vừa mới train xong.
- **Volumes**: Chứa các file cần training, bao gồm file âm thanh và các file nhãn.
- **audio_reader.py**: chứa các hàm đọc file âm thanh
- **conf.json**: Chứa cấu hình, các hằng số cho dự án như "SAMPLE_RATE"…
- **constants.py**: Chứa các hàm để đọc hằng số từ file conf.json
- **ctc_tensorflow_continue_train.py**: Dùng để train tiếp model đã được train trước đó
- **ctc_tensorflow_train.py**: Dùng để train mới một model hoặc train lại từ đầu một model
- **file_logger.py**: Ghi log
- **generate_audio_cache.py**: Tạo các file cache
- **out.tsv**: Chứa các tham số được ghi lại trong quá trình train
- **requirements.txt**: Khai báo các package cần cài đặt để chạy được dự án
- **test_utils.py**: Dùng để test model đã train
- **text_helper.py**: Chứa hàm chuyển đổi kiểu telex thành unicode
- **utils.py**: Chứa các hàm thông dụng như hàm chuyển đổi dữ liệu âm thành thành định dạng đầu vào ctc.

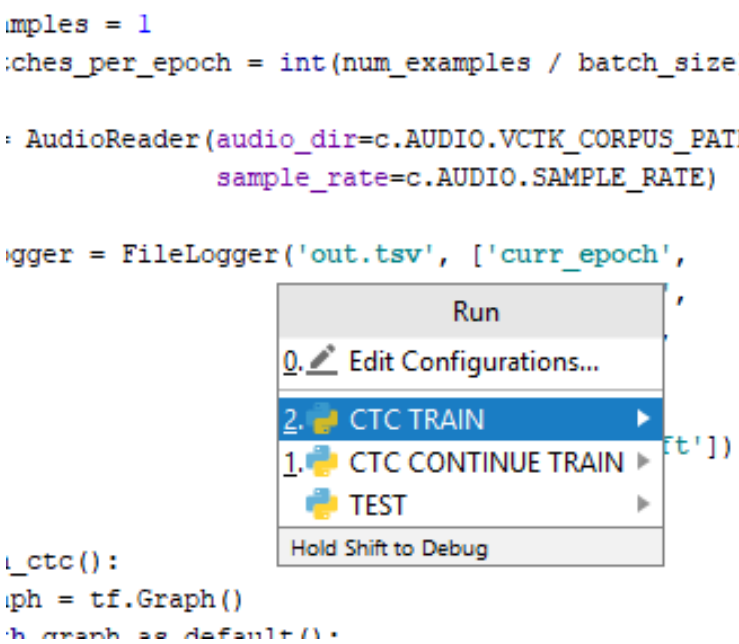
Sau khi mở ứng dụng bằng phần mềm **Pycharm**(đăng ký miễn phí bằng tài khoản email của trường đại học), tiến hành mở file **ctc_tensorflow_train.py**:

Điều chỉnh hai tham số quan trọng là num_epochs và learning_rate để đạt được kết quả tốt nhất.

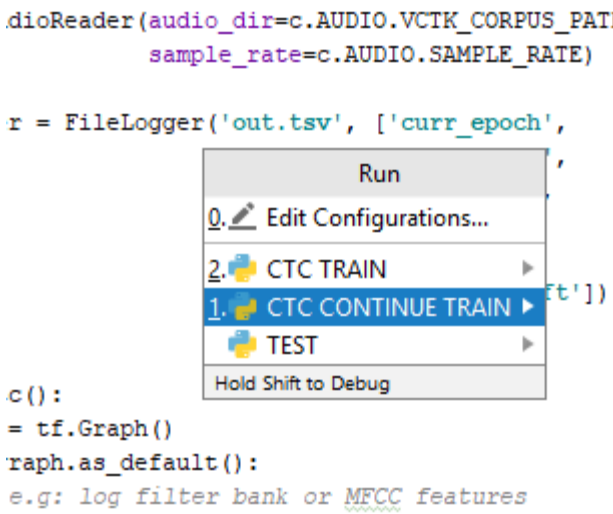
Chọn run:



Chọn **CTC_TRAIN** để tiến hành train:

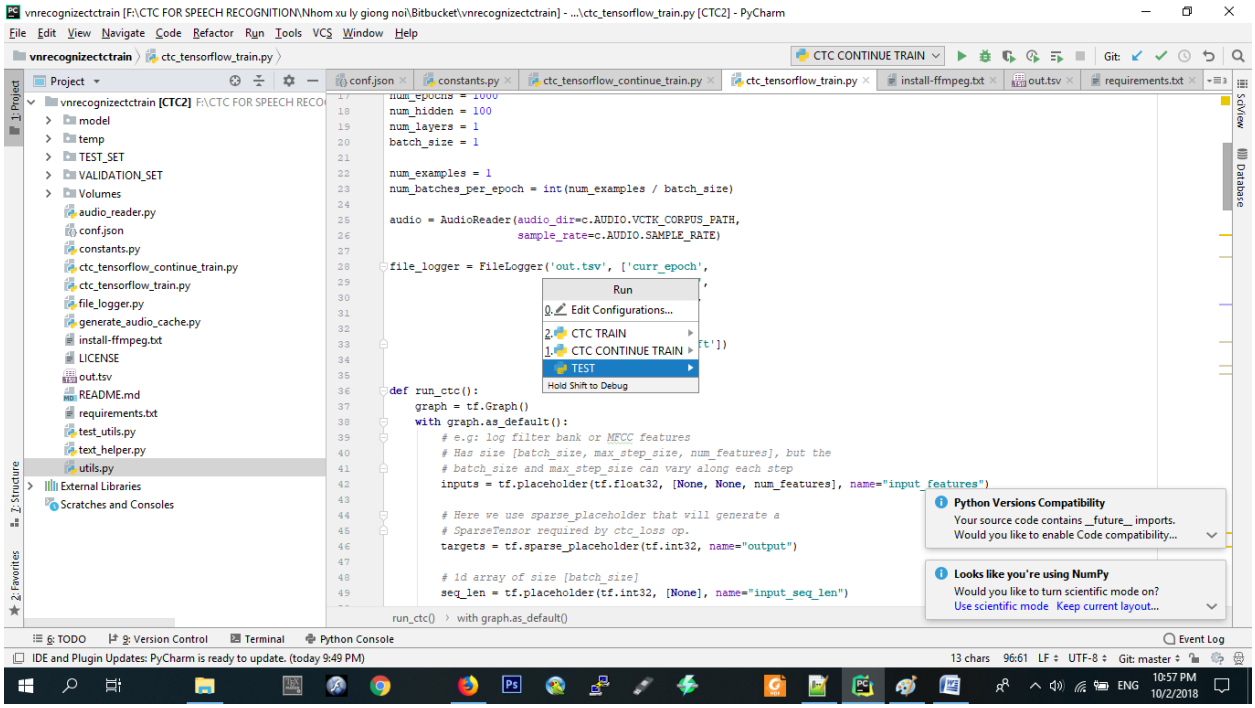


Để train tiếp model trước đó(checkpoint đã được tạo sẵn) chọn **CTC CONTINUE TRAIN**:



III. Testing:

Chọn run-> chọn **TEST**:


vnrecognizectctrain [F:\CTC FOR SPEECH RECOGNITION\Nhom xu ly giong noi\Bitbucket\vnrecognizectctrain] - ...\ctc_tensorflow_train.py [CTC2] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CTC CONTINUE TRAIN
Project
vnrecognizectctrain [CTC2] F:\CTC FOR SPEECH RECO
model
temp
TEST_SET
VALIDATION_SET
Volumes
audio_reader.py
conf.json
constants.py
ctc_tensorflow_continue_train.py
ctc_tensorflow_train.py
file_logger.py
generate_audio_cache.py
install-ffmpeg.txt
LICENSE
out.tsv
README.md
requirements.txt
test_utils.py
text_helper.py
utils.py
External Libraries
Scratches and Consoles
conf.json
constants.py
ctc_tensorflow_continue_train.py
ctc_tensorflow_train.py
install-ffmpeg.txt
out.tsv
requirements.txt
num_epochs = 1000
num_hidden = 100
num_layers = 1
batch_size = 1
num_examples = 1
num_batches_per_epoch = int(num_examples / batch_size)
audio = AudioReader(audio_dir=c.AUDIO.VCTK_CORPUS_PATH,
sample_rate=c.AUDIO.SAMPLE_RATE)
file_logger = FileLogger('out.tsv', ['curr_epoch',
Run
0. Edit Configurations...
2. CTC TRAIN
1. CTC CONTINUE TRAIN
TEST
Hold Shift to Debug
def run_ctc():
graph = tf.Graph()
with graph.as_default():
# e.g: log filter bank or MFCC features
# Has size [batch_size, max_step_size, num_features], but the
# batch_size and max_step_size can vary along each step
inputs = tf.placeholder(tf.float32, [None, None, num_features], name="input_features")
# Here we use sparse_placeholder that will generate a
# SparseTensor required by ctc_loss op.
targets = tf.sparse_placeholder(tf.int32, name="output")
# 1d array of size [batch_size]
seq_len = tf.placeholder(tf.int32, [None], name="input_seq_len")
Python Versions Compatibility
Your source code contains __future__ imports.
Would you like to enable Code compatibility...
Looks like you're using NumPy
Would you like to turn scientific mode on?
Use scientific mode Keep current layout...
run_ctc() › with graph.as_default()
6: TODO
9: Version Control
Terminal
Python Console
Event Log
IDE and Plugin Updates: PyCharm is ready to update. (today 9:49 PM)
13 chars 96:61 LF UTF-8 Git: master
ENG
10:57 PM
10/2/2018