Winning Model Documentation

Name: Xiaozhou WANG
Location: Canada
Email: xiaozhou@ualberta.ca
Competition: TensorFlow Speech Recognition Challenge

Background

1. *What is your academic/professional background?*
    a. Xiaozhou Wang currently works as Chief Data Scientist at Quartic.ai.

2. *Did you have any prior experience that helped you succeed in this competition?*
    a. This was my first time participating in a speech recognition challenge. I have some experience with deep learning but had no domain knowledge on audio/speech processing.

Model Summary

The competition was challenging mainly because the test data has much noisier audio clips than training data so there was a big gap between local validation scores and scores on test data. I believe applying semi supervised learning with test data was the key to my result. First part of my solution was just building various CNNs (i.e. vgg, senet, resnet, densenet) on different inputs (i.e. mel, mfcc, raw wav). Each model's accuracy is around 0.87 - 0.88, and taking a weighted average could push the score to 0.89-0.90. And then two different semi supervised learning techniques were applied. One was directly using (generated) test labels together with training data to train the model again and the other was to "pretrain" a model with the test labels and then fine tune the model with training data. Each technique could boost a model that previously only scored 0.87 to above 0.90. The final submission was just a weighted average of all these models.

Data Preprocessing

- Silence clips were created by randomly chopping from background noise. Up to 3 background noise sounds were randomly mixed and the scale could go up to 200%.
- Unknown clips were subsampled to 20%.
- Clips were padded/chopped to length 16000.
- Each Clip was uniformly scaled within 75% - 125%
- Up to 2 noise sounds were mixed in clips with probability 10% or 30%.
- Noise can scale up to 150% when it is mixed in word clips.

- Random time shift between 0.8 - 1.2.

Features

Three types of features were used for modelling

- mfcc with n_mel = 40
- mel spectrogram with n_mel = 40
- raw input

Models

10 main models were trained

- Resnet with 9 layers on mfcc and mel (input reshaped to 128 * 128)
- Senet18 on mfcc and mel (input reshaped to 128 * 128)
- Densenet121 on mfcc and mel (input reshaped to 128 * 128)
- VGG (with GlobalMaxPooling+GlobalAveragePooling in the end) on mfcc and mel (input reshaped to 128 * 128)
- VGG (with GlobalMaxPooling+GlobalAveragePooling in the end) on raw input
- VGG (with fully connected layers in the end) on mel where it only convs along the time axis

Semi Supervised Learning

Weighted average of previously trained 10 models was used as the “ground truth” labels for test data. Two semi supervised learning techniques were used:

- 100% of training data + 20%-35% of test data selected per epoch were used as the new training data. Resnet (mfcc and mel), Senet (mfcc and mel) and VGG (mel and raw) were trained this way.
- 100% of test data were used to train a “pretrained” model. And then the weights of the “pretrained” model were loaded as initial weights to train the same model with only training data (fine tuning with training data). It turned out that fine tuning with only one epoch was enough to “correct” the overfitted “pretrained” model.

Both techniques were used with the hope that they could learn/approximate the test distribution better (instead of “overfitting” training data distribution). First one does it on the data level and second one does it on the model level.

Appendix

A1. Model Execution Time

- What software did you use for training and prediction?
    - Pytorch
- What hardware (CPUS spec, number of CPU cores, memory)?
    - At the beginning I used one GTX 1080, and GCE P100, and in the last week of the competition, I had access to 4 Titan pascal.
- How long does it take to train your model?
    - With no bagging, the training should be finished in one week with 1 GTX 1080ti. So if bagging number is set to 4, that means 4 weeks of 1080ti training.

A2. Dependencies

older or newer version of below packages should theoretically work fine

python 2.7 or 3.5/3.6
numpy 1.13.3
pandas 0.20.3
pytorch 0.3.0
librosa 0.5.1
skimage 0.12.3

A3. How To Generate the Solution (aka README file)

First make sure that the training and test data are put in `../input/`

Then execute the scripts as follows:

- python train.py
- python base_average.py
- python semi_train.py
- python finetune_train.py
- python final_average.py

And then you will be able to find the submission in `sub/final_average.csv`