Training and testing deep belief networks for auto-tracing

Jeff Berry

January 20, 2011

1 Preparing the training data

The TrainNetwork program expects the images and data to be in a specific format. The directory structure is as follows:

```
.../
    Subject1/
        TongueContours.csv
        JPG/
            itemname_0001.jpg
            itemname_0002.jpg
            ...
    Subject2/
        TongueContours.csv
        JPG/
            itemname_0001.jpg
            itemname_0002.jpg
            ...
    ...
```

Requirements for file names:

- The 'Subject' directories have to have the 'S' capitalized.
- 'TongueContours.csv' must be named exactly that, with 'T' and 'C' capitalized.
- 'JPG' must be capitalized.
- 'itemname' can be any combination of letters and numbers
- For certain features in AutoTrace.py, the frame number must be separated from 'itemname' by an underscore. Zero padding on the frame number is not required.

Format of 'TongueContours.csv':

- TongueContours.csv can be created by renaming 'Palatoglossatron_output.txt', which is created by the Export function of AutoTrace.py. If this method is used, be sure that the filename of each image does *not* contain the full path. If 'Palatoglossatron_output.txt' contains the full paths of the file names, the paths must be deleted by doing a find and replace.
- TongueContours.csv has 1 header line followed by 1 line for each file in the 'JPG' directory.
- Each line should have the file name exactly matching that of an image in 'JPG', followed by a tab and 92 tab-separated numbers. The first 64 numbers contain the (x, y) coordinates of the 32 points of the trace for that image. The next 8 numbers define the positions of the Palatron head correction dots, and can be set to -1 if head correction is not used. The remaining 20 numbers are set to -1, and are a leftover from the Palatoglossatron software.

2 Training with TrainNetwork.py

TrainNetwork.py offers a python user interface to the Matlab scripts that train the network. To start TrainNetwork.py, open a terminal and navigate to the directory that it is stored in. For the Mac Pro in the APIL lab, type:

```
cd ~/Desktop/autotracer/trunk/TrainNetwork
python TrainNetwork.py
```

This will bring up the user interface. Under 'Training input' check 'Ultrasound' and 'Contours'. Under 'Runtime input' check 'Ultrasound'. This tells the machine to train a network that accepts both ultrasound images and traces during training, but only images at runtime.

Select the training data to use by clicking 'Data' and then 'Specify Training Data'. Use the file chooser to navigate to the folder with the 'Subject' folders discussed in the previous section. Select which 'Subject' folders you want to use so they are highlighted, and press 'Open'.

Press 'Go' in the main window. Matlab will start and begin printing things at the bottom of the window. Depending on the number of images, training can take from 30 minutes to many hours. When training is completed, scroll down to the bottom of the window and you will see some strange symbols followed by the > prompt. It is now ok to close the window.

The new network will be located in Desktop/autotracer/trunk/TrainNetwork/savefiles/network.mat. You should rename it and move it to another location.[1]

2.1 Changing the region of interest

Before running 'TrainNetwork.py', you should be sure that the Matlab script is set up to look at the correct region of the images. This is controlled in a file called 'combineUltrasoundandContourImages.m', located in Desktop/autotracer/trunk/TrainNetwork/NeuralNets/.

Use TextEdit or another plain text editor to open the file. Near the top of this file are the lines:

```
%for L images (Sonosite Titan)
%top = 140;
%bottom = 320;
%left = 250;
%right = 580;

%for IIT images
top = 140;
bottom = 320;
left = 200;
right = 520;
```

Remove the % symbol from the beginning of the lines specifying 'top', 'bottom', 'left', and 'right' for the region of interest you want to use. Add the % to the other lines so that 'top', etc. are only specified once without the %. If you want to change the values, please add new lines with your desired values, rather than changing the lines already present. When you run 'AutoTrace.py', you will need to make sure the same values are set in 'AutoTracer.m'. Save the file.

3 Tracing images using the new network

Tracing will be done with the 'AutoTrace.py', which calls the Matlab script called 'AutoTracer.m' located in Desktop/AutoTrace/. You will need to make some changes to 'AutoTracer.m' to use the newly trained network. Open the file using TextEdit or another plain text editor. The third line specifies where the network you want to use is stored. Change the path of the file to the new network, for example:

[1] If you are training multiple networks at once, a simple renaming program can be found in the savefiles directory, called watchdog.py. While running, it will periodically check whether a new file has appeared and rename it according to the time.

```
load /Users/apiladmin/Jeff/multitests/diverse500/network_d500;
```

Notice the absence of the .mat file extension.

Further down, around line 18, you will see lines similar to those discussed in the previous section. Make sure 'top', 'bottom', 'left', and 'right' have the same values you used to train the network by uncommenting the appropriate set of lines. Save your changes and close the file.

Run 'AutoTrace.py' in the usual way. Select the images you want to trace using the 'Open' button, and trace them using the 'Auto Trace' button. Once the tracing is finished, press 'View' to inspect the first trace. 'AutoTracer.m' saves its output to a directory called autotraceTMP. When you press 'View', the traces are moved into the same directory as the images and the autotraceTMP directory is deleted. You will want to make sure that any traces in the directory with the images are moved to another location before pressing 'View', or they will be overwritten. You can now close the view window and the AutoTrace program.

4 Quantifying the results

Use the 'CompareContours.py' program to quantify the results of the new network. To run it open a terminal and type:

```
cd ~/Desktop/CompareContours
python CompareContours.py
```

Select the new traces in 'Traces 1', and the other traces in 'Traces 2'. Select a location to save the results in 'Save Results to'.

Checking 'Trim Contours' will do the analysis only on points that are specified in both traces. In 'Machine' select which machine the images were recorded on. For images collected at UA, machine should be 'Sonosite Titan'. This is important because 'Machine' determines which grid the traces will be compared on.

After pressing 'OK', the program will print a lot of things to the terminal window. When it gives the mean distance, it is done, and you can close the window. The results file will have an average distance for each pair of traces that were compared.

If you want to inspect the details, the results file can be imported into R after deleting the last line of the file that contains the mean. A sample R session might look like this:

```
# read the results into a table
d <- read.table("results.txt", header=F, sep=",")
# look at the boxplot
boxplot(d$V4)
# get the numbers from the boxplot
boxplot.stats(d$V4)
# look at the distribution of the data
plot(density(d$V4))
```