# Infrastructure for enabling fast prototyping of machine learning applications

## Summary

The work described in this abstract is about a software architecture for fast prototyping of cloud based machine learning applications. The architecture is composed by a client side javascript frontend which performs lightweight preprocessing of user data and a backend component which enables the processing of the heavyweight part of the computation in a convenient domain specific language such as Matlab, Octave[1], or R[2].
To validate the architecture, we explore two applications. The first one is a Gender Detector based on speech processing and the second one is a handwriting recognizer. The results obtained show that the architecture can deliver an adequate performance and user experience in addition to eliminating one step in the development process of prototyping machine learning applications that rely on cloud processing.

## Description of the Problem

User applications that rely on heavy computation such as the required for signal processing and machine learning techniques are often separated among a thin frontend and a thick backend where all the heavy processing is performed[3]. This carries the advantage of an improved user experience, as response time can be lowered by leveraging the backend computational power. Also, power consumption, which is of great interest in mobile platforms, can be lowered as well.
On the other hand, the most widely used[1] domain specific languages for working with signal processing and machine learning such as Octave or Matlab are not adequate for end user applications. The runtime environment for these languages are simply too heavy, and were not devised for running in platforms such as mobile phones. Consequently, the prototyping work cycle typically involves a prototyping step in one language/environment and a porting to production step in the production environment.
The main disadvantage of this approach is that it consumes too much development time. Stakeholders cannot have a feeling of the application until it is ported to the production environment. Furthermore, the development engineering team cannot receive early feedback from users to tune the system. Problems can arise in the production environment where root causes are more difficult to identify and changes are more difficult to introduce.

## Solution

The solution proposed in this work involves an infrastructure that allows the fast prototyping in an environment such as Matlab or Octave, and the exposure of the functionality directly as a web service that can be reached by the client side of the application. In this way, one prototyping step is removed, namely porting the application from Octave/Matlab to a production environment, and stakeholders can get a feeling of the real application in shorter time.
On the other hand, developers can perform several experiments in a way that closely mimics the production environment, but using the plethora of libraries and visualization capabilities that are provided in the domain specific languages mentioned. Eventually, once the problem is well understood, the backend can be evolved into technologies more appropriate for production environments.
To illustrate the advantages of such architecture, two example applications will be discussed. The first one is a Gender Detector based on a speech processing framework. The second one

1 Matlab/Octave and R, are the only three languages used for signal processing and machine learning that appear in the top 20 list of most used programming Languages at Redmonk's rank[4].

is a handwriting detector. Both are implemented in using Javascript in the frontend, and rely also on a backend layer for their most expensive computations.

## Results/Impact/Relevance

The two sample applications used to validate the proposed architecture are explained in Figure 1. The top chart depicts the gender detection app. First, the speaker produces a speech signal that is captured using the HTML5 Web Audio API[5]. Then, the preprocessing module performs some basic tasks like decimation, filtering, and coding. This step reduces the amount of data to be transmitted, incurring in minimum computation. The Web Service(WS) module handles communication with the backend where the heavy processing is performed. More complex feature extraction is performed in the backend, and the audio samples are classified and the answer is sent back to the client.
The bottom chart depicts the handwriting recognition app. First, on the client side, an HTML canvas captures the handwritten characters; the preprocessing module will perform basic tasks like character segmentation and scaling. Immediately, the web service module will send a set of images representing characters to the backend. In the backend, characters are classified, and words are predicted using a language model. Results are sent back to the client side.
We conclude then that the proposed architecture can cope with a wide range of relevant applications. Furthermore, the development cycle was dramatically accelerated, as the architecture could be easily deployed, and we leveraged on the vast amount of signal processing and machine learning libraries available for Matlab and Octave. Thus, porting code to a production environment could be delayed until after all the stakeholders had a chance to test the application.

## Lessons learned

We have shown how the proposed architecture is flexible enough for enabling fast prototyping of an interesting set of machine learning applications that rely on backend processing for parts of their computation. Furthermore, we explained how we could evolve the backend algorithms while using a domain specific programming language and receiving early feedback from end users. In this way, the overall development effort and development time were considerably reduced.

## References

[1] http://www.gnu.org/software/octave/

[2] http://www.r-project.org/

[3] A Survey of Computation Offloading for Mobile Systems. Mobile Networks and Applications. Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, Bharat Bhargava.

[4] Redmonk’s rank http://redmonk.com

[5] https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html

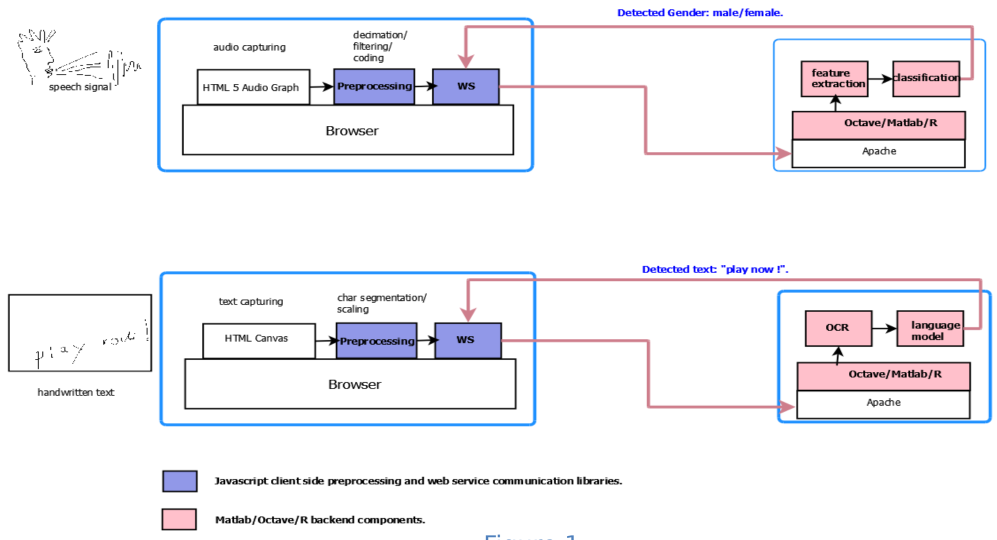

Figure 1.

The top chart depicts the gender detection app, the bottom chart depicts the handwriting detection app. The blue boxes on the left represent the client side, which are web browsers in this case. The right hand side represents web servers . The client side preprocessing module is implemented as a Javascript library, and is intended to perform basic processing of raw data. Its primary intention is to reduce the data to be transmitted to the server. The web service communication module handles communication with the backend.