Formant Estimation and Tracking: A Deep Learning Approach

Yehoshua Dissen,[1, a)] Jacob Goldberger,[2, b] and Joseph Keshet[1, c]
[1] *Dept. of Computer Science, Bar-Ilan University, Ramat Gan, 52900, Israel*
[2] *Faculty of Engineering , Bar-Ilan University, Ramat Gan, 52900, Israel*

(Dated: 7 February 2019)

Formant frequency estimation and tracking are among the most fundamental problems in speech processing. In the estimation task, the input is a stationary speech segment such as the middle part of a vowel, and the goal is to estimate the formant frequencies, whereas in the tracking task the input is a series of speech frames and the goal is to track the trajectory of the formant frequencies throughout the signal.
We propose using supervised machine learning techniques trained on an annotated corpus of read speech for these tasks. We evaluated two sets of deep networks architectures: feed-forward networks and convolutional networks. The input to the former is composed of LPC-based cepstral coefficients with a range of model orders and pitch-synchronous cepstral coefficients, where the input to the latter is the raw spectrogram.
The performance of our method compares favorably with alternative methods for formant estimation and tracking. We further propose a change in the network architecture that allows adaption of the models to new domains and speaker types. We evaluated our adapted networks on three datasets, each of which had different speaker characteristics and speech styles. After adaptation, the performance is further improved and can handle a variety of conditions.
 [http://dx.doi.org(DOI number)]

[XYZ] Pages: 1–11

I. INTRODUCTION

Formants are considered to be resonances of the vocal tract during speech production. There are 3 to 5 formants, each at a different frequency, roughly one in each 1 kHz band. They play a key role in the perception of speech and they are useful in the coding, synthesis and enhancement of speech, as they can express important aspects of the signal using a very limited set of parameters (O'Shaughnessy, 2007). An accurate estimate of these frequencies is also desired in many phonological experiments in the fields of laboratory phonology, sociolinguistics, and bilingualism (Clopper and Tamati, 2014; Munson and Solomon, 2004).

The problem of formant estimation has received considerable attention in speech recognition research as formant frequencies are known to be important in determining the phonetic content as well as articulatory information about the speech signal. They can either be used as additional acoustic features or can be utilized as hidden dynamic variables as part of the speech recognition model (Deng and Ma, 2000).

The formant frequencies approximately correspond to the peaks of the spectrum of the vocal tract. These peaks cannot be easily extracted from the spectrum, since the spectrum is also tainted with pitch harmonics. Most commonly, the spectral envelope is estimated using a time-invariant all-pole linear system, and the formants are estimated by finding the peaks of the spectral envelope (McCandless, 1974; O'Shaughnessy, 2007). While this method is very simple and efficient it lacks the accuracy required by some systems (Clopper and Tamati, 2014; Munson and Solomon, 2004).

Most algorithms for tracking are based on traditional peak picking from Linear Predictive Coding (LPC) spectral analysis or cross-channel correlation methods coupled with continuity constraints (Deng and Geisler, 1987; McCandless, 1974; O'Shaughnessy, 2007). More elaborate methods used dynamic programming and HMMs to force continuity (Kopec, 1986; Lee *et al.*, 2005; Toledano *et al.*, 2006). Other algorithms for formant tracking are based on Kalman filtering (Deng *et al.*, 2004, 2007) and extended in (Mehta *et al.*, 2012). Other authors (Cadzow, 1982; Hernando *et al.*, 1997) have used autocorrelation sequence for representing speech in a noisy speech recognition system and (Anand Joseph *et al.*, 2006; Murthy and Yegnanarayana, 2011; Ribas Gonzalez *et al.*, 2014) use LPC of the zero phase version of the signal and the peaks of its group delay function.

In 2006 a publicly available corpus of manually-annotated formant frequencies of read speech was released (Deng *et al.*, 2006). The corpus is a subset of the TIMIT corpus, and includes around 30 min of tran-

[a)] shua.dissen@gmail.com
[b)] goldbej@eng.biu.ac.il
[c)] jkeshet@cs.biu.ac.il

scription of the first 4 formants at the level of 10 ms frames. The release of this database enables researchers to develop and evaluate new algorithms for formant estimation.

In this paper we present deep learning methods for estimating and tracking formant frequencies using deep networks trained on the aforementioned annotated corpus. In the task of formant *estimation* the input is a stationary speech segment (such as the middle of a vowel) and the goal is to estimate the first 3 formants. In the task of formant *tracking* the input is a sequence of speech frames and the goal is to predict the sequence of the first 3 formants corresponding to the input sequence. In both tasks the signal is represented either spectrograms or using two sets of acoustic features. The first set is composed of LPC cepstral coefficients extracted from a range of LPC model orders, and additionally of cepstral coefficients derived from quasi-pitch-synchronous spectrum.Additional set is based on merely the raw spectrogram.

We use standard and convolutional feed-forward networks for the task of estimation including a domain adaptation technique to promote model universality and recurrent and convolutional recurrent neural network architectures for the task of tracking.

The paper is organized as follows. In the next section we describe the formal problem setting. Section III describes the different sets of features used. Section IV presents the neural network architectures used for estimation and tracking. Section V presents the domain adaptation technique. Section VI details the different datasets used for evaluation. Section VII evaluates the proposed methods by comparing them to state of the art LPC implementations, namely WaveSurfer (Sjölander and Beskow, 2000) and Praat (Boersma and Weenink, 2002), and to two state of the art tracking algorithms: MSR (Deng *et al.*, 2004) and KARMA (Mehta *et al.*, 2012). We conclude the paper in Section VIII.

II. PROBLEM SETTING

We start with a formal problem definition. Formant estimation and tracking is a time-series multiple-regression problem. The input is a sequence of n-dimensional real values, $\bar{\boldsymbol{x}} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, where $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^n$ for $t = 1, \ldots, T$, and T is the number of frames in the input. Note that T is known but not fixed, and can be changed from one example to another. In the case of estimation the output is a k-dimensional real vector $\boldsymbol{y} \in \mathcal{Y} \subset \mathbb{R}^k$, where k is the number of formants we wish to estimate (in our case it was 4). In the case of tracking the output is a sequence of k-dimensional real values, $\bar{\boldsymbol{y}} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$, where $\bar{\boldsymbol{y}} \in \mathcal{Y}^*$.

Our goal is to find a mapping that at least approximately models the dependency between the sequences $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$. Since this is not always possible we will focus on trying to find a mapping between sub-sequences of length τ , between $(\boldsymbol{x}_{t-\tau}, \ldots, \boldsymbol{x}_t)$ and $(\boldsymbol{y}_{t-\tau}, \ldots, \boldsymbol{y}_t)$ or even just between $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$. Here we consider the parametric case $\boldsymbol{y}_t = f_\theta(\boldsymbol{x}_t)$ where f_θ is the prediction function parametrized by a set of parameters θ, and $\boldsymbol{y}_t$ in $\mathcal{Y}$ is the prediction. Thus, we wish to estimate parameters θ such that, in some sense, $f_\theta(\boldsymbol{x}_t) \approx \boldsymbol{y}_t$ for all t.

The goodness of the fit is quantitatively measured in terms of a loss function; For each input $\boldsymbol{x}_t$ the difference between the predicted value $\hat{\boldsymbol{y}}_t$ and the target value $\boldsymbol{y}_t$ or more simply the "performance" are assessed using a local loss function. Here we use the basic absolute error loss

$$\gamma_t\left(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t\right) = |\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t|_1, \tag{1}$$

where $|\cdot|_1$ is the ℓ_1 norm.

Denote by $\bar{\boldsymbol{y}}' = f(\bar{\boldsymbol{x}}) = (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T)$ the predicted sub-sequence of formant frequencies. The overall performance measure is the local loss function averaged over multiple samples

$$\gamma(\bar{\boldsymbol{y}}, \bar{\boldsymbol{y}}') = \frac{1}{T}\sum_{t=1}^{T} \gamma_t\left(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t\right). \tag{2}$$

There are two regression tasks relating to the prediction of formant frequencies. The first task is called *formant estimation* where the input is a single feature vector $\boldsymbol{x}_t$ and the output $\boldsymbol{y}_t$ is the vector of 3 ($k = 3$) formant frequencies. In this task there are two options either $\boldsymbol{x}_t$ is a single frame of speech or $\boldsymbol{x}_t$ stands for a series of frames, belonging to the same speech segment. In both cases only one set of formant frequencies need to be predicted. The second task is called *formant tracking* which essentially is predicting the formants at each time t consecutively (can be many times for a single vowel) or more formally either $\boldsymbol{y}_t$ given $(\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}, \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2})$ or $\boldsymbol{y}_t, \boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}$ given $\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}$.

III. ACOUSTIC SIGNAL REPRESENTATION

In this section we describe the acoustic features, used as input to the formant estimation and tracking models.

A key assumption is that in the task of estimation, the whole speech segment is considered to be stationary, which mainly holds for monophthongs (pure vowels). In the task of tracking, the speech signal is considered stationary in frames of tens of milliseconds. In the former case the features are extracted from the whole segment, while in the latter case the input signal is divided into frames, and the acoustic features are extracted from each frame. The spacing between frames is 10 msec, and frames are overlapping with analysis windows of 30 msec. As with all processing of this type, we apply a pre-emphasis filter, $H(z) = 1 - 0.97\, z^{-1}$, to the input speech signal, and a Hamming window to each frame.

At this phase, two sets of spectral features are extracted. The goal of each of the sets is to parametrize the envelop of the short-time Fourier transform (STFT). The first set is based on LPC analysis, while the second is based on the pitch-synchronous spectra. We now describe in detail and motivate each set of features.

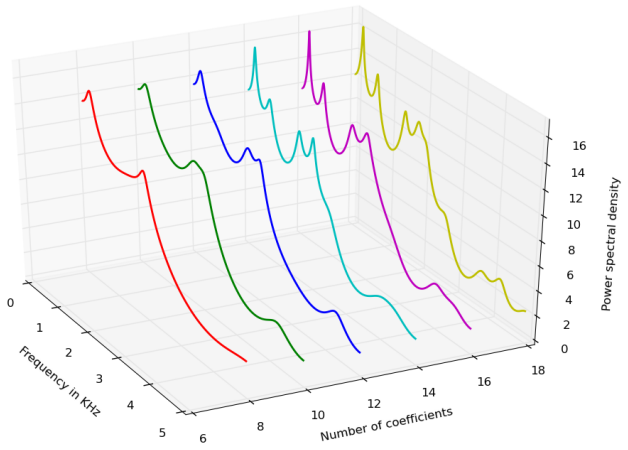

FIG. 1. *LPC spectrum of the vowel /uw/ produced for 262 msec for values of p 8,10,12,14,16, and 18.*

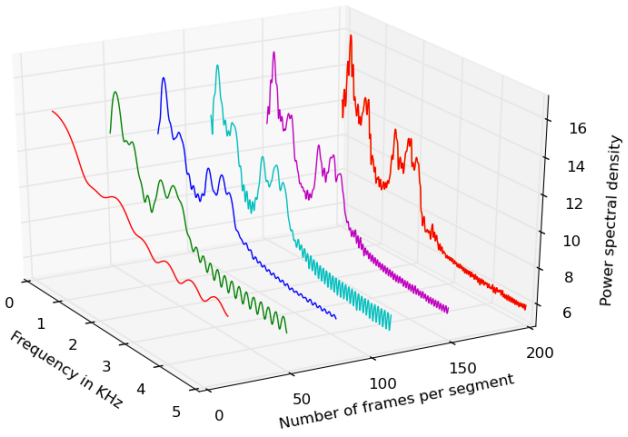

FIG. 2. *Quasi pitch-synchronous sepctra of the vowel /uw/ produced for 262 msec with different values of pitch. The true value of the pitch was 123.4 frames.*

A. LPC-based features

LPC model determines the coefficients of a forward linear predictor by minimizing the prediction error in the least squares sense. Consider a frame of speech of length N denoted by $\bar{s} = (s_1, \ldots, s_N)$, where s_n the n-th sample. The LPC model assumes that the speech signal can be approximated as a linear combination of the past p samples:

$$\hat{s}_n = \sum_{k=1}^{p} a_k s_{n-k} \tag{3}$$

where $\boldsymbol{a} = (a_1, \ldots, a_p)$ is a vector of p coefficients. The values of the coefficients $\boldsymbol{a}$ are estimated so as to minimize the mean square error between the signal $\bar{s}$ and the predicted signal $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_N)$,

$$\boldsymbol{a} = \arg\min_{\boldsymbol{a}} \frac{1}{N} \sum_{n=1}^{N} (s_n - \hat{s}_n)^2. \tag{4}$$

Plugging Eq. (3) into Eq. (4), this optimization problem can be solved by a linear equation system.

The spectrum of the LPC model can be interpreted as the envelop of the speech spectrum. The model order p determines how smooth the spectral envelop will be. Low values of p represent the coarse properties of the spectrum, and as p increases, more of the detailed properties are preserved. Beyond some value of p, the details of the spectrum do not reflect only the spectral resonances of the sound, but also the pitch and some noise. Figure 1 illustrates this concept, by showing the spectrum of the all-pole filter with values of p ranging from 8 to 18. A disadvantage of this method is that if p is not well chosen (i.e., to match the number of resonance present in the speech), then the resulted LPC spectrum is not as accurate as desired (Birch *et al.*, 1988).

Our first set of acoustic features are based on the LPC model. Instead of using a single value of the number of LPC coefficients, we used a range of values between 8 and 17. This way the classifier can combine or filter out information from different model resolutions. More specifically, in our setting after applying pre-emphasize and windowing, the LPC coefficients for each value of p were extracted using the autocorrelation method, where the Levinson-Durbin recursion was used for the autocorrelation matrix inversion, and the FFT for the autocorrelation computation.

The final processing stage is to convert the LPC spectra to cepstral coefficients. This is done efficiently by the method proposed by (Atal, 1974). Denoted by $\boldsymbol{c} = (c_1, \ldots, c_n)$ is the vector of the cepstral coefficients where $n > p$:

$$c_m = \begin{cases} a_m + \sum_{k=1}^{m-1} \left(1 - \frac{k}{m}\right) a_k c_{m-k} & 1 \le m \le p \\ \sum_{k=1}^{p} \left(1 - \frac{k}{m}\right) a_k c_{m-k} & p < m \le n \end{cases}.$$

We tried different values for n and found that $n = 30$ gave reasonable results.

B. Pitch-synchronous spectrum-based features

The spectrum of a periodic speech signal is known to exhibit a impulse train structure located at multiples of the fundamental frequency period. A major concern when using the spectrum directly for locating the formants is that the resonance peaks might fall between two fundamental frequency periods, and then they are not "visible". The LPC model estimates the spectrum envelop to overcome this problem.

Another concern in formant measurements is the bias by the particular fundamental frequency used to excite the formants (Shadle *et al.*, 2016). In order to overcome this problem we added another feature set, namely the *pitch synchronous spectrum* (Medan and Yair, 1989). Ac-

cording to this method the DFT is taken over frames the size of the instantaneous pitch estimation[1].

One of the main problem of this method is the need of a very accurate pitch estimator. Another issue is how to implement the method in the case of formant estimation, when the input is a speech segment that represents a single vowel, which typically spans a few pitch periods, and the pitch in not fixed along the segment. We found out that using a pitch period which is close enough to its exact value is suitable in our application. This can be observed in Figure 2, where the quasi pitch-synchronous FFT for different values of pitch periods are depicted. It can be seen that except for extreme cases, the peaks of the spectra are well-smoothed and clearly defined.

In out implementation we extract quasi-pitch synchronous spectrum similar to (Medan and Yair, 1989). For the task of formant estimation we use the median pitch computed in frames of 10 msec along the input segment, and use the average spectra.

At the final stage, the resulting quasi pitch-synchronous spectrum is converted to cepstral coefficients by applying log compression and then Discrete Cosine transform (DCT). We use the first 50 DCT coefficients as our second set of features.

C. Spectrogram as features

A recent trend in deep learning is to build end-to-end systems where the input is the unprocessed signal (Amodei *et al.*, 2016; Chan *et al.*, 2016). When we work with convolutional neural networks we also used the raw spectrograms. The reason is that the convolutional networks dynamically learns the filters that should be applied on the signal and does not need any further processing. In the task of formant estimation we normalized the spectrogram by remove the DC component, that is we subtract the mean of whole spectrogram. In the task of formant tacking we normalize for each frequency band separately.

IV. NETWORK ARCHITECTURES

A. Feed Forward Architectures For Estimation

In this section we describe the network architectures that are used for formant estimation where the input is a speech segment representing a single vowel and the goal is to extract the first three formants. The networks are multiple regression networks which predict all formants together and use the common loss function mean absolute error $\gamma(\boldsymbol{y}, \boldsymbol{y}) = |\boldsymbol{y} - \hat{\boldsymbol{y}}|_1$. We use neural nets over multiple regression models given that they have been shown to outperform multiple regression models (Brey *et al.*, 1996).

1. Multilayer Perceptrons

The method chosen to classify the LPC based data was a standard feed forward neural network. The the input of the network is a vector of 350 features (30 DCT features for each of the 10 LPC model sizes plus 50 features of the quasi pitch-synchronous spectrum), and the output is a vector of the three annotated formants.

The network has three hidden layers with 1024, 512 and 256 neurons respectively and all of them are fully connected. The activations for said layers are sigmoid functions. The network was trained using Adagrad (Duchi *et al.*, 2011) to minimize the mean absolute error or the absolute difference between the predicted and true formant frequencies with weights randomly initialized. The training of the networks weights was done as regression rather than classification. The network predicts all 3 formants simultaneously to exploit inter-formant constraints.

2. Convolutional net

Convolutional neural networks were used on the raw spectrogram as inputs. Specifically, the input to the CNN are 55×50 spectrograms. The network has four 2D convolutional layers with the ReLU activation and 2D max-pooling after which there two fully connected feed forward layers with ReLU and linear activation functions respectively the output of the latter being the 3 formant frequencies. All convolution windows were 3×3 and all max-pooling windows were 2×2. As before the network was trained using Adagrad to minimize the MAE.

B. Recurrent Architectures For Tracking

In this section we describe the network architectures that are used for formant tracking where the input is a series of speech frames and the goal is to extract the corresponding series of values of the first three formants.

A recurrent neural network (RNN) is a type of neural network that is a powerful sequence learner. In particular, the Long Short-Term Memory (LSTM) architecture has shown to provide excellent modeling of sequential data such as language, music, facial expressions and speech (Graves *et al.*, 2013). Because in the tracking task our data is sequential we will use RNNs that take into account temporal structure as opposed to a standard feed-forward neural networks, which only look at each individual frame as an independent unit.

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a *memory* which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps.

LSTMs do not have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs is called a *cell*. Internally these cells decide what to keep in (and what to erase from) memory, by combining the

previous state, the current memory, and the input. In turns out that these types of units are very efficient at capturing long-term dependencies. Here we define the input as $(\boldsymbol{x}_{t-\tau}, \ldots, \boldsymbol{x}_t, \boldsymbol{y}_{t-\tau}, \ldots, \boldsymbol{y}_{t-1})$ and the predicted label is defined as $\hat{\boldsymbol{y}}_t$, the previous target labels will be injected into the hidden layers.

1. Recurrent Neural Network (RNN)

For the LPC based tracker we use an RNN consisting of an input layer with 350 features as in the estimation task. In addition to these features extracted from the current segment of speech on account of the fact that this is an RNN the predictions and features of the previous speech segment (i.e. temporal context) are taken into account when predicting the current segments formants. Next are two LSTM layers with 512 and 256 neurons respectively, a time distributed fully connected layer with 256 neurons and an output layer consisting of the 3 formant frequencies. As in the estimation network the activations were all sigmoid, the optimizer was adagrad and the function to minimize was MAE.

2. Convolutional Recurrent Neural Network (CRNN)

For the CRNN we use 55 $\times$ 50 spectrograms as input as in the CNN. The first three layers of the network are 2D convolutional LSTMs as described in (Xingjian *et al.*, 2015) Next the network has two time distributed 2D convolutional and max-pooling layers. Again here we use 3 $\times$ 3 window sizes for convolutions and ReLU as the activation function. Finally, we have two fully connected feed forward layers with ReLU and linear activation functions respectively the output of the latter being the 3 formant frequencies. As in the estimation network the activations were all sigmoid, the optimizer was Adagrad and the function to minimize was MAE.

V. A DOMAIN ADAPTATION NETWORK

In this section we describe a network architecture and training procedure that jointly form a domain adaptation approach for formant estimation. The network used for formant estimation is a feed-forward network trained in two phases. First we train a network using only the VTR dataset which is small, with a limited set of speakers and the formant annotation is relatively reliable. The network consists of an input layer with 350 features, 3 fully connected hidden layers of 1024, 512 and 256 neurons, respectively, and an output linear layer that provides an estimation of the 4 formants. Since the network is trained solely on the VTR dataset we denote it hereafter as *the VTR network*.

The trained VTR network yields good results on the test subset of the VTR dataset. However, when it is applied to other datasets there is a significant performance degradation. The goal of this study is to train a single network that can be successfully used on different datasets. Naturally we do not assume that at train or test time the domain identity of a given input is revealed to the network rather we want the network to learn to adapt its predictions based on the original features.

The approach we take in this study is to transfer the VTR network originally trained on the VTR dataset into a Domain Adaptation (DA) network. This network adaptation is done by adding two new elements to the VTR network. First we use the VTR formant estimation output layer as input to another linear layer that produces a new formant estimation. We also add a selection neuron s whose input is the same input of the original network. This selection neuron controls the amount of network adaptation that should be applied to each given input. The control element s is a non-linear function of the input feature vector $\mathbf{c}$ and is implemented by a linear operation followed by a sigmoid activation function, i.e.:

$$s(\mathbf{c}) = \sigma(\mathbf{w_s} \cdot \mathbf{c} + \mathbf{b_s}) \tag{5}$$

such that $\mathbf{c}$ is the input vector, and w_s and b_s are network parameters that are learned at training phase. Denote the formant output of the original VTR network by $\hat{y}_1, ..., \hat{y}_4$. The new formant estimation $\tilde{y}_1, ..., \tilde{y}_4$ is computed as follows:

$$\tilde{y}_i = \sum_{j=1}^{4} w_{ij}\hat{y}_j + b_i + v_i \cdot s(\mathbf{c}) \tag{6}$$

where w_{ij} and b_i are the parameters of the additional linear layer and v_i is a multiplicative term that defines the contribution of the dataset control element $s(\mathbf{c})$ to the estimation of the i-th formant. A scheme of the domain adaptation (DA) network is shown in Figure 3.

We can use all the available datasets to train the DA network from scratch starting from a random initialization of the network parameters. However, since the network topology is complicated, this yields inferior results, as shown in Section VII. Instead, we train the network in two steps. First we train the VTR sub-network only using only the VTR dataset. Next we freeze the parameters of the VTR network and only train the adaptation parameters in Eq. (6) and Eq. (5) using the VTR, Clopper, and Hillenbrand datasets. This two-step training procedure ensures that the VTR sub-network is responsible for the core formant estimation, and the adaptation part of the network is responsible solely for adapting the formant estimation to the conditions of the specific input vector.

One of the advantages of training the network in two steps, is that there is only a small number of adaptation parameters so large amounts of data from each dataset are not necessary. Hence, domains with limited labeled data as in our case can still be used to learn a good formant estimator.

To better visualize what the network has learned we show the histograms of the domain parameters activation values s for each dataset. The histogram axis are the number of examples in each of the 10 buckets of activations between 0 and 1 (which is the output range of a sigmoid function). As seen in the right-most histograms in Figure 4, the s values for the VTR database are almost

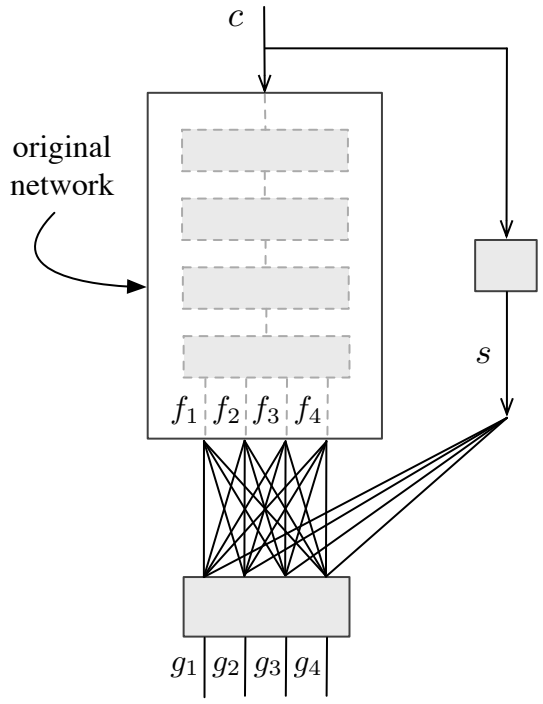

FIG. 3. A scheme of the domain adaptation formant estimation network. The network contains the VTR network component and the adaptation elements.

exclusively concentrated in the same area showing that the network automatically found that no adaptation is needed for data from the VTR dataset. This coincides with the fact that the original network was trained on VTR dataset. In contrast, predictions from the Hillenbrand dataset needed to be corrected occasionally and predictions from the Clopper dataset consistently needed to be corrected, as seen by the variance in s values.

VI. DATASETS

In order to get reliable results we used three different datasets to evaluate the performance of our system. In this section we give a short description of each dataset.

A. Vocal Tract Resonance (VTR)

The first being the Vocal Tract Resonance (VTR) corpus introduced by (Deng *et al.*, 2006). This corpus is composed of 538 utterances selected as a representative subset of the well-known and widely-used TIMIT corpus (Garofolo *et al.*, 1993). These were split into 346 utterances for the training set and 192 utterances for the test set. These utterances were manually annotated for the first 3 formants and their bandwidths for every 10 msec frame. The fourth formant was annotated by the automatic tracking algorithm described in (Deng *et al.*, 2004), and it is not used here for evaluation.

B. Clopper and Tamati, 2014 (CT)

The second dataset (Clopper and Tamati, 2014) contains segments of acoustic signal from 20 female native English speakers aged 18-22 with no history of speech or language deficits. The participants were evenly split between two American English dialects (Northern and Midland). As part of a larger study (Clopper *et al.*, 2002), participants read aloud a list of 991 CVC words. This study focused on 39 target words (777 tokens) which did or did not have a lexical contrast between either /ɛ/ vs. /ae/ (e.g., dead-dad vs. deaf-*daff) or /ɑ/ vs. /ɔ/ (e.g., cot-caught vs. dock-*dawk). Words with a lexical contrast are referred to as *competitor* items and those without are referred to as *no competitor* items.

C. Hillenbrand, Getty, Clark, and Wheeler, 1995 (HGCW)

The third dataset consists of data from a laboratory study conducted by (Hillenbrand *et al.*, 1995). It contains segments of acoustic signal from 45 men, 48 women, and 46 ten-to 12-year-old children (27 boys and 19 girls). 87% of the participants were raised in Michigan, primarily in the southeastern and southwestern parts of the state. The audio recordings contain 12 different vowels (/i, ɪ, ɛ, ae, ɑ, ɔ, ʊ, u, ʌ, ɝ, e, o/) from the words: heed, hid, head, had, hod, hawed, hood, who'd, hud, heard, hayed, hoed.

VII. RESULTS

A. Estimation

We begin by presenting the results for the estimation algorithms. The estimation algorithms apply only to vowels (monophthongs and diphthongs). We used the whole vowel segments of the VTR, Clopper and Hillenbrand datasets . Their corresponding annotation were taken to be the average formants along the segments. The test sets of each dataset are as follows. The VTR dataset has a predetermined test set, the Hillenbrand and Clopper sets were split into train and test sets using about two thirds for train and a third for test, this was done by removing a third of the speakers from each dataset so we as not test on the speakers from the training set.

We start with Table I showing the influence of the different LPC based feature sets trained with a feed forward network. The loss is the mean absolute difference between predicted values and their manually annotated counterparts measured in Hz. It can be seen that using different LPC model orders improves the performance on F_2 and F_3, and the performance on F_1 improves with the quasi-pitch-synchronous feature set. Results here are on the VTR test set.

Next we compare the results achieved by the feed forward network with LPC based features as input (DeepFormants) and the results achieved by the CNN using spectrograms as input. As a baseline we compared both our results to those of Praat, a popular tool in phonetic

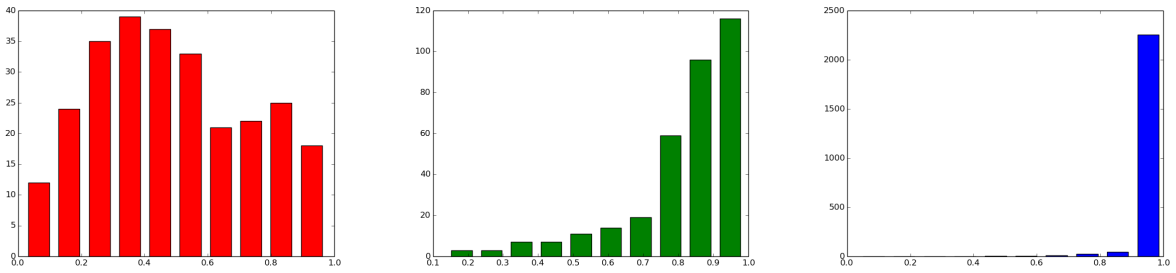

FIG. 4. Histograms of domain bias activations for the datasets Clopper (left), Hillenbrand (middle) and VTR (right).

Feature set	F_1	F_2	F_3
LPC, $p = 12$	59	123	179
LPC, $p = \{8 - 17\}$	60	86	110
quasi-pitch-sync	51	115	164
LPC, $p = \{8 - 17\}$ + quasi-pitch-sync	**48**	**83**	**109**

TABLE I. *The influence of different feature sets on the estimation of formant frequencies of whole vowels using deep learning.*

	Method	F_1	F_2	F_3
Mean	CNN	**45**	**65**	**94**
	DeepFormants	48	83	109
	Praat	75	115	151
Median	CNN	**35**	**48**	**63**
	DeepFormants	38	62	75
	Praat	48	78	88
Max	CNN	**444**	760	**1349**
	DeepFormants	528	**716**	1509
	Praat	1611	1711	1633

TABLE II. *Estimation of formant frequencies of whole vowels using feed forward networks, convolutional networks and Praat.*

research (Boersma and Weenink, 2002). Formants were extracted from Praat using Burg's method with a maximum formant value of 5.5 kHz, a window length of 30 msec and a pre-emphasis from 50 Hz. The results of our system and of Praat's on the test set are shown in Table II, where the loss is the mean absolute difference in Hz. As seen in the table, we have achieved better results across the board over Praat when comparing our respective estimations to the manually annotated reference with the CNN giving the best results in every category other than max loss for F2 over the LPC based system.

Analysis of the predictions with the largest inaccuracies show that they broadly fall into 3 categories: (i) There are annotation errors and the system indeed did classify them accurately; (ii) The vowel segment was very short (less than 35 ms); and (iii) Ambiguous spectrograms, where both the manual annotation and the predicted value can be considered as correct.

B. Adaptation to new data

Here we present the results of the domain adaptation network results on the three databases described in Section VI. We compared our network to the standard networks that was trained only on the VTR data and to the WaveSurfer program (Sjölander and Beskow, 2000), a popular tool in phonetic research. The reason we are not using Praat as a baseline is due to the fact that some of the databases used Praat to give initial estimations to the formant frequencies, and then fixed the mistakes so the annotations are biased towards Praat. The results of our system, DeepFormants and of WaveSurfers on the three datasets we used are shown in Table III, where the loss is the mean absolute difference in Hz. Note that the Clopper dataset was only annotated for the first and the second formants, hence the third formant was not evaluated.

As seen in the table, we achieved better results across the board over WaveSurfer when comparing our respective estimations to the manually annotated reference. The domain adaptation network shows improvement over DeepFormants in both the Clopper and Hillenbrand datasets with no significant drop off in accuracy on the VTR dataset. These results show the advantage of the proposed network architecture over standard networks based on fully connected layers.

When comparing these results to separate networks trained and tested on each of the databases, i.e. training a model with data from the Clopper dataset and then testing on the Clopper test set and another model trained and tested on the Hillenbrand dataset in the same manner and so on for the VTR dataset, we obtained comparable results. Hence, there is no need for multiple models for each domain, this single network can sepa-

Dataset	Method	F_1	F_2	F_3
VTR	WaveSurfer (Sjölander and Beskow, 2000)	70	96	154
	DeepFormants (Dissen and Keshet, 2016)	**48**	**83**	109
	Domain Adaptation	50	86	**104**
Hillenbrand	WaveSurfer	68	190	182
	DeepFormants	71	160	131
	Domain Adaptation	**36**	**100**	116
Clopper	Wavesurfer	128	181	–
	DeepFormants	228	168	–
	Domain Adaptation	**103**	**157**	–

TABLE III. *Estimation of formant frequencies using deep learning with and without domain adaptation and compared to WaveSurfer. Boldface indicates the best result in that category.*

Dataset	training method	F_1	F_2	F_3
VTR	joint training	96	279	283
	two step training	**50**	**86**	**104**
Hillenbrand	joint training	71	119	129
	two step training	**36**	**100**	**116**
Clopper	joint training	**70**	166	–
	two step training	103	**157**	–

TABLE IV. *Results of formant estimation for two possible training procedures of the proposed domain adaptation network. Boldface indicates the best result in that category.*

rate between the speaker and speech domains and adjust its estimations accordingly.

In the next experiment we demonstrate the need for the two step training procedure proposed in this study. Table IV shows the formant estimation results of two training strategies of the domain adaptation network. In the table we compare the results of the two step training to the results of a network with the same topology but trained jointly on all three datasets. As can be seen from Table IV, other than for the first formant in the Clopper dataset the accuracy is greatly diminished across all other data. Moreover adding the selection layer does not improve results over training a network identical to the VTR network but using all three datasets during training.

When using a CNN for estimation there is also some degree of overfitting but in this case there is no need for a two step training, training the model with a combination of all datasets gives it the ability to estimate accurately over all databases without corrupting the results for any other data domain. We can see in Table V that the results improve dramatically for the Clopper and Hillenbrand datasets without changing the VTR results as a result of

Dataset	training method	F_1	F_2
VTR	joint training	47	**65**
	VTR training	**45**	**65**
Hillenbrand	joint training	**22**	**54**
	VTR training	45	92
Clopper	joint training	**65**	**96**
	VTR training	193	190

TABLE V. *Results of formant estimation with training on the VTR set alone and with data from all three datasets. Boldface indicates the best result in that category.*

combined training over training with the VTR dataset alone.

C. Tracking

We now present the results for our tracking models. We evaluated the model on whole spoken utterances of VTR. We compared our results to Praat, to the results obtained in (Deng *et al.*, 2006) from WaveSurfer and from the MSR tracking algorithm. Table VI shows the accuracy in mean absolute difference in Hz for each broad phonetic class. The inter-labeler variation is also presented in this table for reference taken from (Deng *et al.*, 2006).

Both the LPC based RNN and the spectrogram based RCNN outperform Praat and WaveSurfer in every category, and compared to MSR our models show higher precision with vowels and semivowels while MSR reports higher precision with nasals, fricatives, affricates and stops. It is worth mentioning though that the phone class where formants are most indicative of speech phenomena is vowels. The higher precision reported by MSR in consonant phone classes is most likely due to the fact that the database obtained its initial trajectory labels from MSR and was then manually corrected (Deng *et al.*, 2006) so in phonemes without clear formants (i.e. consonants) there is a natural bias towards the trajectories labeled by MSR.

Also noted is the fact that the CNN performed better than th LPC based RNN only in vowels. This is most likely due to the fact that the LPC spectra forces the existence of clear peaks even in consonants whereas the spectrogram of a consonant will not have clearly defined peaks.

In addition, the observed mean differences between our automated measurements and the manually annotated measurements are comparable in size to the generally-acknowledged uncertainty in formant frequency estimation demonstrated on our dataset by the degree of inconsistency between different labelers in Table VI and to the perceptual difference limens found in (Mermelstein, 1978). Such that it is doubtful that higher accuracy can be achieved with automated tools seeing as manual annotation cannot.

	inter-labler			WaveSurfer			Praat			MSR (Deng *et al.*, 2004)			DeepFormants			CNN		
	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3
vowels	55	69	84	70	94	154	130	230	267	64	105	125	54	81	112	**53**	**72**	**108**
semivowels	68	80	103	89	126	222	136	295	334	83	122	**154**	**67**	114	168	68	**111**	160
nasal	75	112	106	96	229	239	219	409	381	67	**120**	**112**	**66**	175	151	69	191	158
fricatives	91	113	125	209	263	439	564	593	700	**129**	**108**	**131**	131	135	159	139	142	167
affricates	89	118	135	292	407	390	730	515	583	**141**	**129**	**149**	164	162	189	174	173	195
stops	91	110	116	168	210	286	258	270	351	130	**113**	**119**	131	135	168	**123**	135	170

TABLE VI. *Tracking errors of on broad phone classes measured by mean absolute difference in Hz.*

	WaveSurfer			Praat			MSR (Deng *et al.*, 2004)			DeepFormants			CNN		
	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3
CV transitions	156	192	273	169	225	261	**106**	**101**	**119**	110	142	165	108	142	166
VC transitions	59	88	157	344	355	495	**48**	92	120	53	**80**	111	54	81	**110**

TABLE VII. *Same as for Table VI except for the focus on temporal regions of CV transitions and VC transitions.*

Method	F_1	F_2	F_3	Overall
KARMA (Mehta *et al.*, 2012)	**114**	226	320	220
DeepFormants	118	**169**	**204**	**163**
RCNN	127	180	213	173

TABLE VIII. *Formant tracking performance of KARMA, and deep learning in terms of root-mean-square error (RMSE) per formant. RMSE is only computed over speech-labeled frames.*

We also examined the errors of the algorithms when limiting the error-counting regions to only the consonant-to-vowel (CV) and vowel-to-consonant (VC) transitions. The transition regions are fixed to be 6 frames, with 3 frames to the left and 3 frames to the right of CV or VC boundaries defined in the TIMIT database. The detailed results are listed in Table VII.

Results from other works on the VTR dataset include (Mehta *et al.*, 2012) and compared to his results seen in Table VIII in both models our precision is on par for the first formant but greatly improved for the second and third formants. Error is measured in root mean squared error (RMSE).

VIII. CONCLUSIONS

Accurate models for formant tracking and estimation were presented with the former surpassing existing automated systems accuracy and the latter within the margins of human inconsistencies. We proposed a formant estimation deep-learning architecture that achieves state-of-the-art results across several speech and speaker domains that are very different in nature. We also proposed a training scheme that validates the claim that each component of the network is indeed responsible for the task it was designed to do, either formant estimation or domain adaptation.

Deep learning has proved to be a viable option for automated formant estimation tasks and if more annotated data is introduced, we project higher accuracy models can be trained as analysis of the phonemes with the least accuracy on average seems to show that they were the ones that were represented the least in the databases.

In this paper we have demonstrated automated formant tracking and estimation tools that are ready to be added to the methods that socio-linguists use to analyze acoustic data. The tools are publicly available at `https://github.com/MLSpeech/DeepFormants`.

ACKNOWLEDGMENTS

This research was supported by the MAGNET program of the Israeli Innovation Authority. We would like to thanks Cynthia Clopper and T.N. Tamati for allowing us to use their dataset.

[1] In the context of estimation of the fundamental frequency we will use the term *pitch estimation*.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G. *et al.* (**2016**). "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, pp. 173–182.

Anand Joseph, M., Guruprasad, S., and Yegnanarayana, B. (**2006**). "Extracting formants from short segments of speech using group delay functions," in *Proceeding of Interspeech*.

Atal, B. S. (**1974**). "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification," the Journal of the Acoustical Society of America **55**(6), 1304–1312.

Birch, G. E., Lawrence, P., Lind, J. C., and Hare, R. D. (**1988**). "Application of prewhitening to ar spectral estimation of EEG," Biomedical Engineering, IEEE Transactions on **35**(8), 640–645.

Boersma, P., and Weenink, D. (**2002**). "Praat, a system for doing phonetics by computer," Glot international **5**(9/10), 341–345.

Brey, T., Jarre-Teichmann, A., and Borlich, O. (**1996**). "Artificial neural network versus multiple linear regression: predicting p/b ratios from empirical data," Marine Ecology Progress Series 251–256.

Cadzow, J. A. (**1982**). "Spectral estimation: An overdetermined rational model equation approach," Proceedings of the IEEE **70**(9), 907–939.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (**2016**). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, IEEE, pp. 4960–4964.

Clopper, C. G., Carter, A. K., Dillon, C. M., Hernandez, L. R., Pisoni, D. B., Clarke, C. M., Harnsberger, J. D., and Herman, R. (**2002**). "The indiana speech project: An overview of the development of a multi-talker multi-dialect speech corpus," Bloomington, Speech Research Laboratory, Indiana University, Research on Speech Perception Progress Report **25**, 367–380.

Clopper, C. G., and Tamati, T. N. (**2014**). "Effects of local lexical competition and regional dialect on vowel production," The Journal of the Acoustical Society of America **136**(1), 1–4.

Deng, L., Cui, X., Pruvenok, R., Chen, Y., Momen, S., and Alwan, A. (**2006**). "A database of vocal tract resonance trajectories for research in speech processing," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, IEEE, Vol. 1, pp. I–I.

Deng, L., and Geisler, C. D. (**1987**). "A composite auditory model for processing speech sounds," The Journal of the Acoustical Society of America **82**(6), 2001–2012.

Deng, L., Lee, L. J., Attias, H., and Acero, A. (**2004**). "A structured speech model with continuous hidden dynamics and prediction-residual training for tracking vocal tract resonances," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, IEEE, Vol. 1, pp. I–557.

Deng, L., Lee, L. J., Attias, H., and Acero, A. (**2007**). "Adaptive kalman filtering and smoothing for tracking vocal tract resonances using a continuous-valued hidden dynamic model," Audio, Speech, and Language Processing, IEEE Transactions on **15**(1), 13–23.

Deng, L., and Ma, J. (**2000**). "Spontaneous speech recognition using a statistical coarticulatory model for the vocal-tract-resonance dynamics," The Journal of the Acoustical Society of America **108**(6), 3036–3048.

Dissen, Y., and Keshet, J. (**2016**). "Formant estimation and tracking using deep learning.," in *INTERSPEECH*, pp. 958–962.

Duchi, J., Hazan, E., and Singer, Y. (**2011**). "Adaptive subgradient methods for online learning and stochastic optimization," The Journal of Machine Learning Research **12**, 2121–2159.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (**1993**). "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," NASA STI/Recon technical report n **93**.

Graves, A., Mohamed, A.-r., and Hinton, G. (**2013**). "Speech recognition with deep recurrent neural networks," in *Acoustics,*

Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, pp. 6645–6649.

Hernando, J., Nadeu, C., and Mariño, J. (**1997**). "Speech recognition in a noisy car environment based on lp of the one-sided autocorrelation sequence and robust similarity measuring techniques," Speech Communication **21**(1), 17–31.

Hillenbrand, J., Getty, L. A., Clark, M. J., and Wheeler, K. (**1995**). "Acoustic characteristics of american english vowels," The Journal of the Acoustical society of America **97**(5), 3099–3111.

Kopec, G. E. (**1986**). "Formant tracking using hidden markov models and vector quantization," Acoustics, Speech and Signal Processing, IEEE Transactions on **34**(4), 709–729.

Lee, M., Van Santen, J., Möbius, B., and Olive, J. (**2005**). "Formant tracking using context-dependent phonemic information," Speech and Audio Processing, IEEE Transactions on **13**(5), 741–750.

McCandless, S. S. (**1974**). "An algorithm for automatic formant extraction using linear prediction spectra," Acoustics, Speech and Signal Processing, IEEE Transactions on **22**(2), 135–141.

Medan, Y., and Yair, E. (**1989**). "Pitch synchronous spectral analysis scheme for voiced speech," IEEE Trans. on Acoustics, Speech and Signal Processing **37**(9), 1321–1328.

Mehta, D. D., Rudoy, D., and Wolfe, P. J. (**2012**). "Kalman-based autoregressive moving average modeling and inference for formant and antiformant trackinga)," The Journal of the Acoustical Society of America **132**(3), 1732–1746.

Mermelstein, P. (**1978**). "Difference limens for formant frequencies of steady-state and consonant-bound vowels," The Journal of the Acoustical Society of America **63**(2), 572–580.

Munson, B., and Solomon, N. P. (**2004**). "The effect of phonological neighborhood density on vowel articulation," Journal of Speech, Language, and Hearing Research **47**(5), 1048–1058.

Murthy, H. A., and Yegnanarayana, B. (**2011**). "Group delay functions and its applications in speech technology," Sadhana **36**(5), 745–782.

O'Shaughnessy, D. (**2007**). "Formant estimation and tracking," in *Springer handbook of speech processing*, edited by J. Benesty, M. M. Sondhi, and Y. Huang (Springer).

Ribas Gonzalez, D., Lleida Solano, E., de Lara, C., and Jose, R. (**2014**). "Zero phase speech representation for robust formant tracking," in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, IEEE, pp. 1462–1466.

Shadle, C. H., Nam, H., and Whalen, D. (**2016**). "Comparing measurement errors for formants in synthetic and natural vowels," The Journal of the Acoustical Society of America **139**(2), 713–727.

Sjölander, K., and Beskow, J. (**2000**). "Wavesurfer-an open source speech tool.," in *Interspeech*, pp. 464–467.

Toledano, D. T., Villardebó, J. G., and Gómez, L. H. (**2006**). "Initialization, training, and context-dependency in hmm-based formant tracking," Audio, Speech, and Language Processing, IEEE Transactions on **14**(2), 511–523.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (**2015**). "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, pp. 802–810.