HIERARCHICAL RNNS FOR WAVEFORM-LEVEL SPEECH SYNTHESIS

Qingyun Dou, Moquan Wan, Gilles Degottex, Zhiyi Ma, Mark J.F. Gales
Cambridge University Engineering Department, UK

ABSTRACT
Speech synthesis technology has a wide range of applications such as voice assistants. In recent years waveform-level synthesis systems have achieved state-of-the-art performance, as they overcome the limitations of vocoder-based synthesis systems. A range of waveform-level synthesis systems have been proposed; this paper investigates the performance of hierarchical Recurrent Neural Networks (RNNs) for speech synthesis. First, the form of network conditioning is discussed, comparing linguistic features and vocoder features from a vocoder-based synthesis system. It is found that compared with linguistic features, conditioning on vocoder features requires less data and modeling power, and yields better performance when there is limited data. By conditioning the hierarchical RNN on vocoder features, this paper develops a neural vocoder, which is capable of high quality synthesis when there is sufficient data. Furthermore, this neural vocoder is flexible, as conceptually it can map any sequence of vocoder features to speech, enabling efficient synthesizer porting to a target speaker. Subjective listening tests demonstrate that the neural vocoder outperforms a high quality baseline, and that it can change its voice to a very different speaker, given less than 15 minutes of data for fine tuning.

Index Terms— Waveform-level speech synthesis, hierarchical RNN, conditioning vector, flexible speech synthesis

1. INTRODUCTION
Speech synthesis systems are highly valuable. They can, for example, be used in voice assistants. Speech synthesis can be non-parametric [1, 2, 3] or parametric [4, 5, 6, 7]. Non-parametric speech synthesis is also known as concatenative speech synthesis; it generates speech by concatenating units of recorded speech, and requires a large corpus to cover the target domain. Parametric speech synthesis is also known as statistical speech synthesis; it generates speech by learning the mapping from text to speech, and is more flexible [8].

Parametric speech synthesis comes in two categories: feature-level speech synthesis [9, 10, 11] and waveform-level speech synthesis [12, 13, 14, 15]. In feature-level speech synthesis, also known as vocoder-based speech synthesis, a model generates a sequence of vocoder features, which is then mapped to a sequence of waveform samples by a vocoder [8]. Feature-level speech synthesis is hence limited by the quality of the vocoder. Typically the vocoder assumes stationarity in a sliding window, and uses explicitly engineered vocoder features that leads to lossy representations of waveforms. In contrast, waveform-level speech synthesis is not limited by these factors [12, 13].

A major challenge for waveform-level speech synthesis is to model a sequence of waveform samples, which is orders of magnitude longer than the corresponding sequence of frames. For sequence modeling, it is natural to use vanilla RNN [16, 17], or its variations which can model longer sequences, such as Gated Recurrent Unit (GRU) [18] and Long Short Term Memory (LSTM) [19]. However, due to the vanishing gradient problem [20], even LSTM and GRU cannot directly model waveforms. To deal with this problem, one approach is to use dilated Convolutional Neural Networks (CNNs) [12]. An alternative is to use hierarchical RNNs [17], which is computationally more efficient [21].

Another challenge for waveform-level speech synthesis is to represent text, typically as a sequence of feature vectors, also known as conditioning vectors. Several approaches have been proposed. One approach [13] uses features implicitly extracted from character sequences by a neural network, and uses attention mechanism. The extracted features can be powerful but are hardly interpretable, and the attention mechanism has the issue of attention drift [22]. In contrast, other approaches respectively use linguistic features [12] and vocoder features [23] from vocoder-based synthesis systems.

This paper investigates the performance of hierarchical RNNs for waveform-level speech synthesis, rather than dilated CNNs. The main contributions are as follows. First, it investigates different network conditioning, i.e. text representation, and finds that conditioning on vocoder features reduces the need of data and modeling power. Next, it develops a neural vocoder by conditioning the hierarchical RNN on vocoder features, and introduces some training techniques that can also be applied to other neural networks. Furthermore, this paper investigates the flexibility of the neural vocoder. Both
objective measures and subjective listening tests are used in the experiments. The results demonstrate that the neural vocoder outperforms a high quality baseline, and that it can change its voice to a very different speaker given less than 15 minutes of data for fine tuning.

2. WAVEFORM REPRESENTATION

The probability of a sequence of waveform samples \( x_{1:T} = \{x_1, \ldots, x_T\} \) given a sequence of text \( w_{1:M} = \{w_1, \ldots, w_M\} \) can be factorized as a product of conditional probabilities, as formulated in equation 1. \( \theta_{\text{wav}} \) denotes model parameters.

\[
p(x_{1:T} | w_{1:M}, \theta_{\text{wav}}) = \prod_{t=1}^{T} p(x_t | x_{1:t-1}, w_{1:M}, \theta_{\text{wav}}) \quad (1)
\]

The conditional distribution \( p(x_t | x_{1:t-1}, w_{1:M}, \theta_{\text{wav}}) \) of an individual waveform sample can be modeled by a mixture model [24, 25], or a categorical distribution [26]. This paper uses categorical distribution, because it is more flexible than mixture models. To apply categorical distribution to waveform samples, this paper uses \( \mu \)-law quantization [27], because it results in higher resolution for the samples close to the mean of the samples, which is preferable for speech waveforms. Previous research has demonstrated the advantages of categorical distribution and \( \mu \)-law quantization [12, 17].

3. TEXT REPRESENTATION

Text representation plays an important role in speech synthesis, and several different approaches have been proposed. One approach is to use raw text, i.e. a sequence of words or characters, as shown in figure 1 (d). The advantage of this approach is that there is no loss of information; the disadvantage is that mapping raw text to speech requires a complicated system, which is often difficult to train and adapt. In previous research adopting this approach, part of the system is a neural network with attention mechanism, which implicitly extracts a sequence of feature vectors from text. These feature vectors can be powerful but are hardly interpretable, and the attention mechanism has the issue of attention drift [22].

This paper uses two other approaches of text representation, where feature extraction is made explicit to reduce the need of modeling power and data. A sequence of feature vectors is used to represent text, and mapping feature vectors to speech usually requires a less complicated system. This can be viewed as introducing an intermediate variable \( c_{1:T} \) to equation 1 and approximating an integral by a point estimate \( c_{1:T}^* \), as formulated in equations 2 to 4.

\[
p(x_{1:T} | w_{1:M}, \theta_{\text{wav}}) = \int p(x_{1:T} | c_{1:T}, \theta_{\text{wav}}) p(c_{1:T} | w_{1:M}) dc_{1:T} \quad (2)
\]

\[
p(c_{1:T} | w_{1:M}) = \delta(c_{1:T} - c_{1:T}^*); c_{1:T} = f_{\text{wav}}(w_{1:M}) \quad (3)
\]

\[
p(x_{1:T} | w_{1:M}, \theta_{\text{wav}}) = p(x_{1:T} | c_{1:T}^*, \theta_{\text{wav}}) \quad (4)
\]

The feature vectors \( c_{1:T}^* \) are also known as conditioning vectors. The two types of conditioning vectors used in this paper are linguistic features and vocoder features. It should be noted that standard linguistic features and vocoder features need to be upsampled to match the sampling frequency of waveform samples. In this paper, upsampling is realized by linear interpolation.

Linguistic features \( l_{1:T} \) are vectors derived from text using Natural Language Processing (NLP) techniques. A linguistic feature vector corresponds to a phoneme, and contains information such as the identities of the current phoneme and the neighbour phonemes, the position of the phoneme in the syllable, the position of the syllable in the word, etc [9]. The derivation process is not trainable, and is shown in equation 5. Figure 1 (b) shows a synthesis system conditioned on linguistic features.

Vocoder features \( o_{1:T} \) can be extracted from waveforms by a vocoder, or generated by a frame-level speech synthesis system. In this paper, the vocoder features used for conditioning are generated; the generation process is trainable, as formulated in equation 6. \( \theta_{\text{voc}} \) denotes the parameters of the frame-level synthesis system, which are trainable. A vocoder feature vector contains information such as mel-cepstrum and fundamental frequency [28]. In this paper PML vocoder features are used, because a study comparing various vocoders shows that PML vocoder has the best overall performance across the performed tests [29]. Figure 1 (c) shows a synthesis system conditioned on acoustic features, which are generated by a synthesis system shown in figure 1 (a).
\begin{align*}
    c_{1:T} &= l_{1:T} = f_{ed}(w_{1:M}) \\
    c_{1:T}^* &= \arg \max_{c_{1:T}} p(a_{1:T}|l_{1:T}; \theta_{voc})
\end{align*}

Compared with linguistic features, conditioning on vocoder features has two advantages. First, it reduces the need of data and modeling power. A waveform-level speech synthesis system conditioned on vocoder features can be viewed as a neural vocoder, since it replaces the vocoder mapping vocoder features to waveform samples. In contrast, when conditioned on linguistic features, the system replaces a more complicated process of mapping linguistic features to vocoder features and then to waveform samples. Second, conditioning on vocoder features makes a waveform-level speech synthesis system flexible. Conceptually, a neural vocoder can map any sequence of vocoder features to speech, enabling efficient synthesizer porting to a target speaker.

4. HIERARCHICAL RECURRENT NEURAL NETWORKS

The conditional distribution \( p(x_t| x_{1:t-1}, c_{1:T}) \) of a waveform sample depends on the conditioning vectors as well as previous waveform samples. Even with a relatively low sampling rate of 16kHz, there are about 6000 waveform samples for each word. However, conventional sequence models such as LSTM and GRU can only cover about 100 time steps. To deal with this problem, this paper uses Hierarchical RNNs (HRNNs).

4.1. Model architecture

HRNN takes into account the fact that waveform samples contain structures at different time scales [17]. It uses a hierarchy of tiers, each operating at a different time scale, i.e. frequency. Figure 2 illustrates the structure of HRNN; a 3-tier model is shown but the configuration can be tuned to suit different tasks. The lowest tier operates at waveform-level frequency, and outputs distributions of waveform samples. Each higher tier operates at a lower frequency, and supervises the tier below it.

Except the lowest tier, all tiers operate below waveform-level frequency. These tiers are RNNs operating on non-overlapping frames of waveform samples, hence they are also known as frame-level tiers. Each frame-level tier summarizes the history of its inputs into a supervising vector for the next tier downward. These tiers can be formulated as equations 7 to 9. \( K + 1 \) is the number of tiers and \( k \) the tier index. \( c, f, s \) and \( W \) denote conditioning vector, frame vector, supervising vector and weight matrix; \( e \) and \( h \) denote input and out of RNN. To increase readability, unless necessary the superscript \((k)\) is not shown for \( e^{(k)}, h^{(k)}, f^{(k)}, \) and \( W^{(k)}\).

Table 1. Relation of time steps at different tiers

<table>
<thead>
<tr>
<th>time step</th>
<th>tier 0</th>
<th>tier 1</th>
<th>tier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>40</td>
</tr>
</tbody>
</table>

For each tier \( k \), a frame vector \( f^{(k)}_t \) includes \( FS^{(k)} \) previous waveform samples \( \{ x_{t+1}^{(k)}, \ldots, x_t^{(k)} \} \), where \( FS \) stands for frame size. The complete history, which has a growing length, is approximated by a fixed-length history vector \( h_t^{(k)} \). At each time step \( t^{(k)} \), the RNN makes a history update as a function of the previous history \( h_{t-1}^{(k)} \) and the current input \( e_t^{(k)} \), as shown in equation 8. For the top tier \( (k = K) \), the current input is a linear combination of a frame \( f^{(k)}_t \) and a conditioning vector \( e_t^{(k)} \). For intermediate tiers \((0 < k < K)\), it also includes a supervising vector from the next tier upward \( s_r^{(k+1)} \).

To condition the next tier downward, the history vector is upsampled into a series of \( R^{(k)} \) supervising vectors, where \( R^{(k)} \) is the ratio between the frame sizes of the two tiers. If \( k = 1 \), \( R^{(k)} = FS^{(k)} \); otherwise \( R^{(k)} = FS^{(k)}/FS^{(k-1)} \). This upampling is realized by a set of \( R^{(k)} \) different linear mappings, which has been demonstrated to work well [30]. Note that for each tier, time step \( t^{(k)} \) is related to a different frequency. Each time step \( t^{(k)} \) at tier \( k \) corresponds to \( R^{(k)} \) time steps in tier \( k - 1 \). Table 1 illustrates how time steps at different tiers are related, taking the configuration shown in figure 2 as example. In this configuration, \( FS^{(2)} = 80 \), \( FS^{(1)} = 20 \), \( FS^{(0)} = 20 \), \( R^{(2)} = 4 \), \( R^{(1)} = 20 \).

The lowest tier operates at waveform-level frequency, i.e. sampling frequency; hence it is also known as sample-level tier. This tier is a DNN with a softmax output layer, and can be formulated as equations 10 to 12.

\begin{align*}
    e_{t^{(0)}} &= W^{(0)} f^{(0)}_{t^{(0)}} + s_{t^{(0)}}^{(1)} \\
    y_{t^{(0)}} &= f_{\text{dnn}}(e_{t^{(0)})} \\
    x_{t^{(0)}} &= \text{Cat}(y_{t^{(0)})}
\end{align*}
considerably [17]. The output $y_t^{(0)}$ is a vector corresponding to a categorical distribution, each dimension showing the probability of one category. Each waveform sample is sampled from its categorical distribution.

During training, the complete sequence of waveform samples is available. During synthesis, this is not the case, so the model is run in an auto-regressive fashion: the sampled output at one time step will be included in the input at future time steps.

4.2. Training techniques

As a neural network, HRNN is trained with Stochastic Gradient Decent (SGD), which can be stuck in a local minimum that degrades performance. This section analyzes two such scenarios, and introduces training techniques that can also be applied to other neural networks. In our experiments, these training techniques considerably improved performance.

When using conditioning vectors with high dimensionality, the number of weights increases accordingly, and training can be problematic. SGD can be stuck in a local minimum where information from the conditioning vector overwhelms that from the previous waveform samples [31]. This problem can be solved by pretraining HRNN as an unconditional model. During pretraining, the weights for conditioning vectors are locked to zero. After pretraining, these weights are randomly initialized and normal training begins. During normal training, all the weights are trained together.

When using a deep HRNN with many tiers, training can be also problematic. Tier-wise training is a helpful technique in this scenario. To train a $K + 1$-tier HRNN, a $K$-tier HRNN is trained first; then its weights are used to initialize the $K + 1$-tier HRNN. When training the $K + 1$-tier HRNN, the weights of the extra tier, which are randomly initialized, are trained for a few epochs, while other weights are locked; after the tier-wise training phase, all the weights will be trained together.

5. EXPERIMENTS

5.1. Data configurations

The experiments in this paper are performed on two datasets with different size, namely Nick and Nancy. Nick dataset contains 2396 utterances from a male British speaker; each utterance is about 2 seconds so there is about 3 hours’ speech in total. Nancy dataset contains 12095 utterances from a female American speaker; each utterance is about 5 seconds so there is about 15 hours’ speech in total.Regardless of the dataset, 50 utterances are for validation, another 50 utterances are for testing, and the rest utterances are for training.

For waveforms, the sampling frequency is 16kHz, and the samples are quantized into 256 integer values. For conditioning vectors, both linguistic features and vocoder features have a frequency of 200Hz; when necessary they are upsampled as described in section 3. The linguistic features are 601-dimensional vectors; the first 592 dimensions are binary and the other dimensions are continuous. The vocoder features are 163-dimensional vectors; all dimensions are continuous.

5.2. Model configurations and training

The baseline synthesis model operates at feature-level, and is shown in figure 1 (a). It maps a linguistic feature sequence to a vocoder feature sequence, which is later used by a PML vocoder to generate speech. The mapping is realized by a neural network, and $\theta_{voc}$ denotes its parameters. It has three bidirectional LSTM (BLSTM) layers, and the dimension of each layer is 1024 for Nick dataset, and 512 for Nancy dataset.
Although called a baseline, this model is well tuned and can generate speech with high quality. After training, this model is used to generate vocoder features for all utterances of the two datasets; these generate vocoder features are used as conditioning vectors for neural vocoders.

For waveform-level synthesis, a 4-tier HRNN is used to map a conditioning vector sequence to speech, as shown in figure 1 (b) and (c). $\theta_{\text{av}}$ denotes the model parameters. Tier 0 is a 4-layer DNN, including three fully connected layers with ReLU activation and a softmax output layer; the dimension is 1024 for the first two fully connected layers, and is 256 for the other two layers. The other tiers are all 2-layer RNNs; GRU is used and the dimension is 1024 for all layers. The frequencies for tiers 0 to 3 are respectively 16000Hz, 3200Hz, 800Hz and 200Hz. The reason to use this HRNN configuration is as follows. In general, using many tiers operating at diverse frequencies leads to good performance. From experience, the effective history length of GRU is about 100 time steps. For tier 3, each time step corresponds to 5ms, so the effective history length is 500ms, which is at word-level. Similarly, for tier 2, the effective history length is 125ms, which is at phoneme-level. For tier 1, the effective history length is 31.25ms, which is at sub-phoneme-level and keeps the generated waveform from being too smooth. For each RNN tier, the number of layers to use should be consistent with its upsampling rate $R^{(k)}$. A tier with higher upsampling rate outputs more supervising vectors, and should have more layers [31]. In the above configuration, $R^{(3)} = R^{(2)} = 4$ and $R^{(1)} = 5$, therefore all RNN tiers have two layers.

During training, SGD is used to minimize the negative log-likelihood. Gradients are hard-clipped to remain in the range $(-1, 1)$. Adam optimizer [32] with an initial learning rate of 0.001 ($\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$) is used. The initial RNN state of all the RNN-based models is learnable. Weight normalization [33] is used for all the linear layers to accelerate training. Truncated back propagation through time is also used to accelerate training. Orthogonal weight matrices are used for initializing hidden-to-hidden connections and other weight matrices are initialized in a similar way to the way proposed by He et al. [34].

## 5.3. Performance metrics

For speech synthesis, objective metrics are only indicative, and subjective metrics are the gold-standard. In this paper, both objective and subjective metrics are used. When developing a speech synthesis system, we evaluate it with objective metrics, which are free and fast to use. When the system has a reasonable quality, we evaluate it with subjective listening tests.

Each listening test compares two systems, and is taken by more than 30 workers from Amazon Mechanical Turk. Participants are instructed to listen to pairs of sentences, and indicate which one they prefer in terms of overall quality. Each comparison includes 5 pairs of utterances randomly selected among all the test utterances.\(^1\)

For objective metrics, vocoder features are extracted from reference and generated waveforms, and root-mean-square error (RMSE) is computed between the feature trajectories. When computing RMSE, two trajectories are synchronized by phase shifting and zero-padding in a way that the resulting RMSE is minimal. To cover more aspects of speech, STRAIGHT vocoder as well as PML vocoder are used. For both vocoders, the dimensions of a vocoder feature vector can be separated into three streams. For STRAIGHT vocoder, the streams are aperiodicity (AP), mel-cepstrum (MCEP) and the log of fundamental frequency (F0). For PML vocoder, the streams are noise mask (NM), MCEP and log(F0). RMSE are computed separately for each stream. Within each stream, the RMSE of all dimensions are added. Although the energy level for each dimension is different, adding them up is sensible because the dimensions with higher energy are more important for the quality of generated speech.

## 5.4. Investigation on conditioning vectors

To investigate different conditioning vectors, we trained two HRNNs with Nick dataset, using linguistic features and vocoder features respectively. It is expected that conditioning on vocoder features yields better performance, because it requires less data and modeling power. Figure 3 shows the result of the listening test comparing linguistic features and vocoder features. Each number indicates the percentage of participants with a certain preference. It can be seen that most participants prefer conditioning on vocoder features. Table 2 compares linguistic features and vocoder features using objective metrics. It shows the added RMSE for MCEP, NM and log(F0). The vocoder analysis is performed with a PML vocoder. It can be seen that when conditioning on vocoder features, the model has better performance in all three aspects.

## 5.5. Investigation on high quality speech synthesis

For HRNNs trained with Nick dataset, a problem is that the generated utterances have obvious background noise, which

---

\(^1\)Utterances generated in the experiments are randomly selected and made available at [http://mi.eng.cam.ac.uk/~qd212/slt2018/](http://mi.eng.cam.ac.uk/~qd212/slt2018/)
Table 2. Added RMSE for different conditioning vectors; PML vocoder analysis; Nick test set

<table>
<thead>
<tr>
<th>Conditioning</th>
<th>MCEP</th>
<th>NM</th>
<th>log(F0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocoder</td>
<td>7.60</td>
<td>8.15</td>
<td>0.39</td>
</tr>
<tr>
<td>Linguistic</td>
<td>8.14</td>
<td>8.61</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Table 3. Added RMSE for different models; PML vocoder analysis; Nancy test set

<table>
<thead>
<tr>
<th>Model</th>
<th>MCEP</th>
<th>NM</th>
<th>log(F0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRNN</td>
<td>6.75</td>
<td>10.82</td>
<td>0.51</td>
</tr>
<tr>
<td>Baseline</td>
<td>7.27</td>
<td>8.93</td>
<td>0.42</td>
</tr>
</tbody>
</table>

is largely due to the amount of data being small. In our internal listening test, most participants prefer the BLSTM baseline to HRNNs, and state that an important reason is the background noise. To reduce the background noise and improve the quality of generated utterances, we trained a HRNN conditioned on vocoder features with Nancy dataset, which is 5 times as large as Nick dataset. As expected, the generated utterances have a much lower level of noise. Figure 4 shows the result of the listening test comparing the HRNN with the corresponding BLSTM baseline. It can be seen that most participants prefer HRNN.

Table 3 shows the added RMSE of the HRNN and the corresponding LSTM baseline, and PML vocoder is used for this analysis. It can be seen that the HRNN is better than the baseline in mel-cepstrum, but is worse in log(F0) and noise mask. As discussed in section 5.3, objective metrics are only indicative. Table 4 shows the results of the same analysis, except that STRAIGHT vocoder is used. The HRNN is still better than the baseline in mel-cepstrum, and it is comparable in log(F0). Meanwhile, the HRNN is much better in terms of aperiodicity.

Table 4. Added RMSE for different models; STRAIGHT vocoder analysis; Nancy test set

<table>
<thead>
<tr>
<th>Model</th>
<th>MCEP</th>
<th>AP</th>
<th>log(F0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRNN</td>
<td>12.54</td>
<td>209.54</td>
<td>2.82</td>
</tr>
<tr>
<td>Baseline</td>
<td>13.46</td>
<td>250.35</td>
<td>2.81</td>
</tr>
</tbody>
</table>

Fig. 4. Result of the listening test comparing HRNN and baseline; Nancy test set

Fig. 5. Results of the listening tests comparing HRNN, HRNNFT100% and HRNNFT10%; Nick test set

is not fine tuned, it can map a sequence of vocoder features from the test set of Nick data (male, British) to speech that sounds like a British male.

Figure 5 shows the results of listening tests comparing three systems: HRNN, HRNNFT10% and HRNNFT100%. For HRNN, \( \theta_{\text{wav}} \) and \( \theta_{\text{voc}} \) are trained with all the Nick training data. For HRNNFT10% and HRNNFT100%, first \( \theta_{\text{wav}} \) and \( \theta_{\text{voc}} \) are trained with all the Nancy training data, then \( \theta_{\text{voc}} \) is trained with all the Nick training data, but \( \theta_{\text{wav}} \) is fine tuned with respectively 10% and 100% of Nick training data. It can be seen that HRNN and HRNNFT100% have very similar quality. More importantly, 10% of Nick training data, which is about 10 minutes, is enough to fine tune the model to be comparable with the HRNN trained with 100% of Nick training data.

6. CONCLUSION

This paper investigates hierarchical RNNs for waveform-level speech synthesis. First, the form of network conditioning is discussed, comparing linguistic features and vocoder features from a vocoder-based synthesis system. Conditioning on vocoder features is found to require less data and modeling power than linguistic features. Next, this paper develops a neural vocoder by conditioning the hierarchical RNN on vocoder features. This neural vocoder is capable of high quality synthesis and efficient synthesizer porting to a target speaker. Both objective measures and subjective listening tests are used in the experiments. The results demonstrate that the neural vocoder outperforms a high quality baseline, and that it can change its voice to a very different speaker given less than 15 minutes of data for fine tuning.
7. REFERENCES


