DISCRIMINATIVE SEGMENTAL CASCADES
FOR FEATURE-RICH PHONE RECOGNITION

Hao Tang, Weiran Wang, Kevin Gimpel, Karen Livescu

Toyota Technological Institute at Chicago
{haotang,weiranwang,kgimpel,klivescu}@ttic.edu

ABSTRACT

Discriminative segmental models, such as segmental conditional random fields (SCRFs) and segmental structured support vector machines (SSVMs), have had success in speech recognition via both lattice rescoring and first-pass decoding. However, such models suffer from slow decoding, hampering the use of computationally expensive features, such as segment neural networks or other high-order features. A typical solution is to use approximate decoding, either by beam pruning in a single pass or by beam pruning to generate a lattice followed by a second pass. In this work, we study discriminative segmental models trained with a hinge loss (i.e., segmental structured SVMs). We show that beam search is not suitable for learning rescoring models in this approach, though it gives good approximate decoding performance when the model is already well-trained. Instead, we consider an approach inspired by structured prediction cascades, which use max-marginal pruning to generate lattices. We obtain a high-accuracy phonetic recognition system with several expensive feature types: a segment neural network, a second-order language model, and second-order phone boundary features.

Index Terms— segmental conditional random field, structured prediction cascades, phone recognition, segment neural network, beam search

1. INTRODUCTION

Segmental models have been considered for speech recognition as an alternative to frame-based models such as hidden Markov models (HMMs), in order to address the shortcomings of the frame-level Markov assumption and introduce expressive segment-level features. Segmental models include segmental conditional random fields (SCRFs) [1], or semi-Markov conditional random fields [2]; segmental structured support vector machines (SSVMs) [3]; and generative segmental models [4, 5]. Previous work comparing segmental model training algorithms has shown some benefits of discriminative segmental models trained with hinge loss (SSVM-type learning) [6], and we consider this type of model here.

Discriminative segmental models have allowed the exploitation of complex features, both at the word level [7] and at the phone level [8, 9, 6]. These powerful segmental features are a double-edged sword—on the one hand, the model becomes more expressive; on the other, it is computationally challenging to decode with and train such models. For this reason, SCRFs [10] and SSVMs [3] were initially applied to speech recognition in a multi-pass approach, where the segmental model considers only a subset of the hypothesis space contained in lattices generated by HMMs. Much effort has been devoted to removing the dependency on HMMs and instead developing first-pass segmental models [11, 9, 12]. However, working with the entire hypothesis space imposes an even larger burden on inference, especially when the features are computationally intensive or of high order.

If we wish to consider the entire search space in decoding, we can only afford features of low order or of specific types as in [9]. An alternative approach to the problem is to use approximate decoding. There are two widely used approximate decoding algorithms: beam search and multi-pass decoding. In the intuitive and popular beam search, the idea is to prune as we search along the graph representing the search space. It has been used for decoding in almost all HMM systems, and for generating lattices as well. Though popular, it offers no guarantees about its approximation. In the category of multi-pass decoding, lattice and n-best list rescoring [13] are commonly used alternatives.

We focus on a particular type of multi-pass approach based on structured prediction cascades [14], which we term discriminative segmental cascades. A cascade is a general approach for decoding and training complex structured models, using a multi-pass sequence of models with increasing order of features, while pruning the hypothesis space by a multiplicative factor to counteract the growth in feature computation. In this approach, the hypothesis space in each pass is pruned with max-marginals, which offers the guarantee that all paths with scores higher than the pruning threshold are kept.

Applying the discriminative segmental cascade approach to speaker-independent phonetic recognition on the TIMIT data set, we obtain a first-pass phone error rate of 21.4% with a unigram language model, and a two-stage cascade error rate of 19.9%, which includes a bigram language model, a segment neural network classifier, and second-order phone
For speech recognition, we formally define the hypothesis space \( \mathcal{Y} \) as the set of vertices, \( \mathcal{E} \) as the set of edges, and \( \mathcal{V} \) as the set of initial vertices. We equip each edge \((u, v)\) with a weight \( w(u, v) \) and a function \( \phi(u, v) \) that associates a weight to an edge.

A linear segmental model for input space \( \mathcal{X} \) and hypothesis space \( \mathcal{Y} \) is defined formally as a pair \((\theta, \phi)\), where \( \theta \in \mathbb{R}^d \) is the parameter vector and \( \phi: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d \) is the feature function. For an input \( x \in \mathcal{X} \), each hypothesis \( y \in \mathcal{Y} \) is associated with a score \( \theta^T \phi(x, y) \), and the goal of decoding is to find the hypothesis that maximizes the score.

For speech recognition, we formally define the hypothesis space \( \mathcal{Y} \) in terms of finite-state transducers (FST). Let \( \Sigma \) be the label set (e.g., the phone set in phone recognition), and \( \mathcal{S} = \Sigma \cup \{\epsilon\} \), where \( \epsilon \) is the empty label. Define a decoding graph as a standard FST \( G = (V, E, I, F, w, i, o) \), where \( V \) is the set of vertices, \( E \subseteq V \times V \) is the set of edges, \( I \subseteq V \) is the set of initial vertices, \( F \subseteq V \) is the set of final vertices, \( w: E \rightarrow \mathbb{R} \) is a function that associates a weight to an edge, \( i: E \rightarrow \mathcal{S} \) is a function that associates an input label to an edge, and \( o: E \rightarrow \mathcal{S} \) is a function that associates an output label to an edge. In addition to the standard definition of FSTs, we equip \( G \) with a function \( t: V \rightarrow \mathbb{R} \) that maps a vertex to a time stamp. For any edge \((u, v) \in E\), let \( \text{tail}((u, v)) = u \) and \( \text{head}((u, v)) = v \). For convenience, we will use subscripts to denote components of a particular FST, e.g., \( E_G \) is the edge set of \( G \).

For an input utterance, let \( x \) be the sequence of acoustic feature vectors. We construct a decoding graph \( G \) from \( x \), then define our hypothesis space \( \mathcal{Y} \subseteq 2^E \) to be the subset of paths that start at an initial vertex \( i \) and end at a final vertex \( o \) in \( F \). A path \( y \in \mathcal{Y} \) of length \( m \) is a sequence of unique edges \( \{e_1, \ldots, e_m\} \), satisfying \( \text{head}(e_i) = \text{tail}(e_{i+1}) \) for \( i \in [m] \). Given a model \((\theta, \phi)\), for each edge \( e \in E \), the weight \( w(e) \) is defined as \( \theta^T \phi(x, e) \). For convenience, for a path \( y \in \mathcal{Y} \), we overload \( \phi \) and \( w \) and define \( \phi(x, y) = \sum_{e \in y} \phi(x, e) \) and \( w(y) = \theta^T \phi(x, y) = \sum_{e \in y} w(e) \), where we treat a path \( y \) as a set of (unique) edges \( e \).

If the decoding graph is the full hypothesis space with all possible segmentations and all possible labels, for example the graph on the left in Figure 1, then the model is a first-pass segmental model. Otherwise, it is a lattice rescoring model. By the above definitions, inference (decoding) in the model (1) can be solved with a standard shortest-path algorithm.

The model parameters \( \theta \) can be learned by minimizing the sum of loss functions on samples \((x, y)\) in a training set. In general, the model can be trained with different losses. The model is an SCRF if we train it with log loss \(- \log p(y|x)\) where \( p(y|x) \propto \exp(\theta^T \phi(x, y)) \). It is a segmental structured SVM if we use the structured hinge loss:

\[
\ell_{\text{hinge}}(\theta) = \max_{y' \in \mathcal{Y}} \left[ \text{cost}(y, y') - \theta^T \phi(x, y) + \theta^T \phi(x, y') \right],
\]

where \( \text{cost}: \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty) \) measures the badness of a hypothesis path \( y' \) compared with the ground truth \( y \).

The loss can be optimized with first-order methods, such as stochastic gradient descent (SGD). The gradient (or subgradient, in this case) computation typically involves a forward-backward-like algorithm. For example, the subgradient of the hinge loss is

\[
\nabla \ell_{\text{hinge}}(\theta) = -\phi(x, y) + \phi(x, y'),
\]

where computing the cost augmented path

\[
y' = \max_{y' \in \mathcal{Y}} \text{cost}(y, y') + \theta^T \phi(x, y'),
\]

requires a forward pass over the graph. Compared to computing the gradient of other losses, which requires more forward passes and backward passes, hinge loss has computational advantages, and has been shown to perform well [6], so we will use hinge loss for the rest of the paper.

### 3. High-Order Features and Structured Composition

The order of a feature is defined as the number of labels on which it depends. A feature is said to be a first-order feature if it depends on a single label, a second-order feature if it depends on a pair of labels, and so on. Features with no label dependency are called zeroth-order features.

High-order features in sequence prediction can be extended from low-order ones by increasing the number of labels considered. Formally for any label set \( \Sigma \) and any feature vector \( \phi \in \mathbb{R}^d \), the feature vector \( \phi \) is **lexicalized** with a label \( s \in \Sigma \) as \( \phi \otimes s = \phi \otimes 1_s \), where \( 1_s \) is a one-hot vector of length \(|\Sigma|\) for the label \( s \) and \( \otimes: \mathbb{R}^d \otimes \mathbb{R}^{p \times q} \rightarrow \mathbb{R}^{mp \times nq} \) is the outer product. With a slight abuse of notation, we let \( \phi \otimes s = \phi \otimes 1_s \). The resulting vector is of length \(|\Sigma| d\).

Similarly, we can lexicalize a feature vector with pairs of labels, \( \phi \otimes s_1 \otimes s_2 = \phi \otimes 1_{s_1} \otimes 1_{s_2} \), giving a vector of length \(|\Sigma|^2 d\).

For example, a common type of zeroth-order segmental feature is of the form \( \psi(x, t_1, t_2) \) where \( x \) is the sequence of acoustic feature vectors, \( t_1 \) is the start time of the segment, and \( t_2 \) is the end time of the segment. To make it discriminative in a decoding graph \( H \), we can compute the first-order feature \( \phi_H(x, e) \) for any edge \( e \) by first computing \( \psi(x, t(\text{tail}(e)), t(\text{head}(e))) \) and then lexicalizing it with the label \( o_H(e) \).
Fig. 1. From left to right: An example of the full hypothesis space $H_1$ with four frames (five vertices) and three unique labels \{a, b, c\} (three edges between every pair of vertices) with segment length up to three frames (actual labels omitted for clarity); $H_2$, a pruned $H_1$; a graph structure corresponding to a bigram language model $L_2$ over three labels; and $H_2$ $\sigma$-composed with $L_2$, where $s_1s_2$ denotes the bigram $s_1s_2$.

To have a unified way of extending the order of features, we define the concept of FST structured composition, or $\sigma$-composition for short, as follows. For any two FSTs $A$ and $B$, the $\sigma$-composed FST is defined as

$$G = A \circ_\sigma B$$

where

$$V_G = V_A \times V_B$$
$$E_G = \{ (e_1, e_2) \in E_A \times E_B : \sigma_A(e_1) = \sigma_B(e_2) \}$$

and

$$i_G(e_1, e_2) = i_A(e_1)$$
$$o_G(e_1, e_2) = o_B(e_2)$$
$$\text{tail}_G(e_1, e_2) = \text{tail}_A(e_1), \text{tail}_B(e_2)$$
$$\text{head}_G(e_1, e_2) = \langle \text{head}_A(e_1), \text{head}_B(e_2) \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes a tuple. Unlike in classical composition, we only constrain the structure of $G$ and are free to define $w_G$ differently. In particular, we let

$$w_G(e_1, e_2) = \theta_G^T \phi_G(x, (e_1, e_2)),$$  \hspace{1cm} (12)

and $\phi_G$ is free to use $\phi_A$ and $\phi_B$, but is not constrained to do so. In other words, the weight function $w_G$ can extract richer features than $w_A$ and $w_B$.

With structured composition, we can easily convert low-order features to high-order ones. Continuing the above example, we can $\sigma$-compose the decoding graph $H$ with a bigram language model (LM) $L$ in its FST form [15] with a slight modification. We require the output labels of the LM FST to include the history labels alongside the current label. For example, the output labels of a bigram LM are of the form $s_1s_2 \in \Sigma \times \Sigma$, where $s_1$ is the history label (possibly $\epsilon$) and $s_2$ is the current label. Let $G = H \circ_\sigma L$. We can define $t_G((e_1, e_2)) = t_H(e_1)$. For an edge $e \in E_G$, we can compute first-order features $\varphi \otimes s_1$, and second-order features $\varphi \otimes s_1 \otimes s_2$ for $s_1s_2 = \sigma_G(e)$ and $s_1 \neq \epsilon$, where $\varphi = \psi(x, t_G(\text{tail}_G(e)), t_G(\text{head}_G(e)))$. If $s_1 = \epsilon$, everything falls back to the previous example. In general, by $\sigma$-composing with high-order $n$-gram LMs, we can compute high-order features by lexicalizing low-order ones.

4. DISCRIMINATIVE SEGMENTAL CASCADES

Our approach, which we term a discriminative segmental cascade (DSC), is an instance of multi-pass decoding, consisting of levels with increasing complexity of features and decreasing size of search space. We start with the full search space and a “simple” first-level discriminative segmental model using inexpensive features, and use the first-level model to prune the search space. We then apply a model using more expensive features, and optionally repeat the process for as many levels as desired. Rather than the typical beam pruning, we prune with max-marginals [16, 14], which have certain useful properties and turn out to be important for achieving good performance with our models. A max-marginal of an edge $e$ in $G$ is defined as

$$\gamma(e) = \max_{y \in \gamma} \theta^T \phi(x, y).$$ \hspace{1cm} (13)

In words, it is the highest score of a path that passes through the edge $e$. We prune the edge if its max-marginal is lower than a threshold, and keep it otherwise. In order to prune a multiplicative factor of edges at each level of the cascade, Weiss et al. [14] propose to use the threshold

$$\tau_\lambda = (1 - \lambda) \frac{1}{|E_G|} \sum_{e \in E_G} \gamma(e) + \lambda \max_{y \in Y} \theta^T \phi(x, y),$$ \hspace{1cm} (14)

which interpolates between the mean of the max-marginals and the maximum. If $\lambda$ is set to 1, we only keep the best path.

Lattice generation by max-marginal pruning guarantees that there is always at least one path left after pruning and that any $y$ satisfying $w(y) > \tau_\lambda$ is kept, because for every $e \in y$, $\gamma(e) \geq w(y) > \tau_\lambda$. In particular, if the ground truth has a score higher than the threshold, it will still be in the search space for the next level of the cascade.

Computing max-marginals in a specific level of the cascade requires a forward pass and a backward pass through the graph. Pruning with max-marginals thus takes twice the time as searching for the best path alone.

Learning the cascade of models is also done level by level. We start with the entire hypothesis space $H_1$ limited only by a maximum segment length. A first set of computationally inexpensive features up to first order is used for learning. Let
Table 1. A summary of results in terms of phonetic error rate (%) on the TIMIT test set, for prior first-pass segmental models, a speaker-independent HMM-DNN system given by a standard Kaldi recipe [18], and our models.

<table>
<thead>
<tr>
<th>Model</th>
<th>dev PER (%)</th>
<th>test PER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM-DNN</td>
<td>21.4</td>
<td></td>
</tr>
<tr>
<td>first first-pass SCRF [8]</td>
<td>33.1</td>
<td></td>
</tr>
<tr>
<td>our first-pass model ($H_1$)</td>
<td>22.15</td>
<td>21.73</td>
</tr>
<tr>
<td>DSC 2nd level with bigram LM</td>
<td>19.80</td>
<td></td>
</tr>
<tr>
<td>+ 2nd-order boundary features</td>
<td>19.22</td>
<td></td>
</tr>
<tr>
<td>+ 1st-order segment NN</td>
<td>18.86</td>
<td></td>
</tr>
<tr>
<td>+ 1st-order bi-phone NN bottleneck</td>
<td>18.77</td>
<td>19.93</td>
</tr>
</tbody>
</table>

The input to the network is a window of 15 frames of log-mel filter outputs. The network has five convolutional layers, with 64–256 filters of size 5 × 5 for the input and $3 \times 3$ for others, each of which is followed by a rectified linear unit (ReLU) [20] activation, with max pooling layers after the first and the third ReLU layers. The output of the final ReLU layer is concatenated with a window of 15 frames of MFCCs centered on the current frame, and the resulting vector is passed through three fully connected ReLU layers with 4096 units each. The network is trained with SGD for 35 epochs with a batch size of 100 frames. Fully connected layers and the concatenation layer are trained with dropout at a 20% and 50% rate, respectively. This classifier was tuned on the development set and achieves a 22.1% frame error rate (after collapsing to 39 phone labels) on the test set. We will use CNN($x, t$) to denote the log of the final softmax layer, corresponding to the predicted log probabilities of the phones, given as input $[x_{t-7}; \ldots; x_{t+7}]$.

5.1. First-order features

Below we list the features for each edge $(u, v)$. We will use $L = t(v) - t(u)$ for short.

**Average of CNN log probabilities** The log of the CNN output layer is averaged over all frames in the segment:

$$\frac{1}{L} \sum_{i=0}^{L-1} \text{CNN}(x, t(u) + i)$$

(15)

**Samples of CNN log probabilities** The log of the CNN output layer is sampled from the middle frames of three equally split sub-segments, i.e.,

$$\text{CNN} \left( x, t(u) + \left\lfloor \frac{k + (k + 1)L}{3 \cdot 2} \right\rfloor \right)$$

(16)

for $k = 0, 1, 2$.

**Boundary features** The log probabilities $i$ frames before the left boundary CNN($x, t(u) - i$) and $i$ frames after the right boundary CNN($x, t(v) + i$) are used as features. We use the concatenation of the boundary features for $i = 1, 2, 3$.

**Length indicator** $1_{L=\ell}$ for $\ell = 0, 1, \ldots, 30$.

**Bias** A constant 1.

We lexicalize all of the above features to first order, and include a zeroth-order bias feature. We minimize hinge loss with the overlap cost function introduced in [6] with AdaGrad for up to 70 epochs with step sizes tuned in

5. EXPERIMENTS

We experiment with segmental models in the context of phonetic recognition on the TIMIT corpus [17]. We follow the standard TIMIT protocol for training and testing. We use 192 randomly selected utterances from the complete test set other than the core test set as our development set, and will refer to the core test set simply as the test set. The phone set is collapsed from 61 labels to 48 before training. In addition to the 48 phones, we also keep the glottal stop ‘/q/’, sentence start, and sentence end so that every frame in the training set has a label. A summary of prior first-pass decoding results with segmental models, along with our results and one from a standard speaker-independent HMM-DNN, is shown in Table 1.

5.1. First-pass segmental model

First we demonstrate the effectiveness of our first-pass decoder. The first-pass search graph, denoted $H_1$, contains all possible labels and all possible segmentations up to 30 frames per segment. Like some prior segmental phonetic recognition models [11, 9], many of the features in our first-pass decoder are based on averaging and sampling the outputs of a neural network phonetic frame classifier, specifically a convolutional neural network (CNN) [19], which we describe next.
Fig. 2. Beam search on $H_1$ with different beam widths. Left: Hit rate on the development set. Right: PER on the development set. The dashed line is the PER of the exact search.

\{0.01, 0.1, 1\}. No explicit regularizer is used; instead we choose the step size and iteration that perform best on the development set (so-called early stopping). As shown in Table 1, our first-pass segmental model outperforms all previous segmental model TIMI results of which we are aware.

5.2. Higher-order features and segmental cascades

We next explore multi-pass decoding with beam search and with discriminative segmental cascades. In the second pass we include features of order two and a bigram LM $L_2$. Back-off is approximated with $\epsilon$ transitions in the bigram LM. Let $G = H \circ L_2$, where $H$ can be $H_1$ or $H_2$, the pruned $H_1$. We consider the following additional features on edges $e \in E_G$.

**Bigram LM score** The bigram log probability $\log p_{LM}(s_2|s_1)$, where $s_1 s_2 = o_G(e)$ We do not lexicalize this feature because it is naturally second-order.

5.2.1. Beam search

Before experimenting with the second-order features, we compare beam search and exact search on the best model for $H_1$ to give a sense of the approximation quality of beam search. We measure the quality of approximation via the “hit rate”, i.e., how often the exact best path is found. Results are shown in Figure 2. As expected, the hit rate decreases as the beam width decreases. However, the PER does not decrease significantly, which demonstrates that beam search is a good approximate decoding algorithm when the model is well-trained.

Judging from the decoding results, we use beam search with beam widths \{10, 20, 30\} for learning. Since the runtime of beam search is controlled by the beam width when the decoding graph is large, we can search directly on $H_1 \circ L_2$. The composition is done on the fly to avoid enumerating all edges in $H_1 \circ L_2$. We compare learning on both $H_1$ and $H_1 \circ L_2$. For $H_1$ we use the same features as the first-pass segmental model, while for $H_1 \circ L_2$ we add the bigram LM score and second-order boundary features. For consistency, we use the same beam width for decoding. Hinge loss is minimized with AdaGrad with step sizes tuned in \{0.01, 0.1, 1\}. Results are shown in Figure 3 for the step size that achieves the lowest development set PER. When we train the segmental model on $H_1$ (top of Figure 3), learning with beam search is successful when the beam width is large enough, while for $H_1 \circ L_2$ (bottom of Figure 3), learning completely fails.

5.2.2. Discriminative segmental cascades (DSC)

We next consider the proposed discriminative structured cascades (DSC) for utilizing the bigram LM and second-order features. We first prune $H_1$ with max-marginal pruning using our first-pass segmental model with weights $\theta_1$, resulting in $H_2$, and $\sigma$-compose $H_2$ with $L_2$. Recall that the larger the pruning parameter $\lambda$, the sparser the lattice. We measure the density of the lattice by the number of edges in $H_2$ divided by the number of ground-truth (gold) edges. The quality of $H_2$’s produced with different $\lambda$’s is shown in Figure 4 (left). For the DSC second level, we define an additional feature:

**Lattice score** Instead of re-learning all of the weights for the features in the first-pass model, we combine them into an additional feature from the first level of the cascade $\theta_1^T \phi_{H_1}(x, e_1)$, which is never lexicalized, where $e_1 \in H$ is such that $(e_1, e_2) \in E_G$.

To compare with beam search, we use the lattice score, the bigram LM score, second-order boundary features, first-order length indicators, and first-order bias as our features for the second level of the cascade. Hinge loss is minimized with AdaGrad for up to 20 epochs with step sizes optimized in \{0.01, 0.1\}. Again, no explicit regularizer is used except early stopping on the development set. Learning results on different lattices are shown in Figure 4 (right). We see that learning with the DSC is clearly better than with beam search.

5.2.3. Other expensive features

To add more context information, we use the same CNN architecture and training setup to learn a bi-phone frame classifier, but with an added 256-unit bottleneck linear layer before
We have presented discriminative segmental cascades (DSC), an approach for training and decoding with segmental models that allows us to incorporate high-order and complex features in a coarse-to-fine approach, and have applied them to the task of phone recognition. The DSC approach uses max-marginal pruning, which outperforms beam search for learning the second-pass model. Starting from a first-pass large-margin model that outperforms previous segmental model results and is competitive with HMM-DNNs, the DSC second pass improves the phone error rate by another 1.8% absolute.

Further analysis may be needed to understand precisely why learning with beam search is not successful in the context of our models. One issue is that \( \sigma \)-composing \( H_1 \) and \( L_2 \) introduces many dead ends (paths that do not lead to final vertices) in the graph because we have to do the composition on the fly. Minimizing \( H_1 \circ_\sigma L_2 \) might help, but we would need to touch the edges of \( H_1 \circ_\sigma L_2 \) at least once, which is itself expensive. Second, even if we reach the final vertices, the cost-augmented path might still have a lower cost+score than the ground-truth path, which leads to no gradient update. This issue has been studied recently, and one possible solution is “premature updates” [28], but these are intended for the perceptron loss. Third, the edge weights in our models are not strictly negative. Beam search would tend to go depth-first when encountering edges with positive weights. On the other hand, if the edge weights are negative, beam search would tend to go breadth-first, which may explain why greedy search like beam search may cause problems for segmental models but works for HMMs.

Additional future work includes considering even more expressive features, higher-order features and additional cascade levels. There is also much room for exploration with segment neural network classifiers. One concern with our segment classifiers is that they are trained only with ground truth segments, so it is unclear how they behave when the input is an incorrect hypothesized segment. Alternatives include training on all hypothesized segments and allowing the network to learn to classify non-phones, similarly to the anti-phone and near-miss modeling of [5].

7. ACKNOWLEDGEMENT

This research was supported by NSF grant IIS-1433485. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency.
8. REFERENCES


