BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

SEMI-SUPERVISED TRAINING OF DEEP NEURAL NETWORKS FOR SPEECH RECOGNITION

'SEMI-SUPERVISED' TRÉNOVÁNÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ PRO ROZPOZNÁVÁNÍ ŘEČI

PHD THESIS (EXCERPTS)
TEZE DISERTAČNÍ PRÁCE - AUTOREFERÁT

AUTHOR — **Ing. KAREL VESELÝ**
AUTOR PRÁCE

SUPERVISOR — **Doc. Ing. LUKÁŠ BURGET, Ph.D.**
ŠKOLITEL

BRNO 2017

Abstract

In this thesis, we first present the theory of neural network training for the speech recognition, along with our implementation, that is available as the 'nnet1' training recipe in the Kaldi toolkit. The recipe contains RBM pre-training, mini-batch frame Cross-Entropy training and sequence-discriminative sMBR training. Then we continue with the main topic of this thesis: semi-supervised training of DNN-based ASR systems. Inspired by the literature survey and our initial experiments, we investigated several problems: First, whether the confidences are better to be calculated per-sentence, per-word or per-frame. Second, whether the confidences should be used for data-selection or data-weighting. Both approaches are compatible with the framework of weighted mini-batch SGD training. Then we tried to get better insight into confidence calibration, more precisely whether it can improve the efficiency of semi-supervised training. We also investigated how the model should be re-tuned with the correctly transcribed data. Finally, we proposed a simple recipe that avoids a grid search of hyper-parameters, and therefore is very practical for general use with any dataset. The experiments were conducted on several data-sets: for Babel Vietnamese with 10 hours of transcribed speech, the Word Error Rate (WER) was reduced by 2.5%. For Switchboard English with 14 hours of transcribed speech, the WER was reduced by 3.2%. Although we found it difficult to further improve the performance of semi-supervised training by means of enhancing the confidences, we still believe that our findings are of significant practical value: the untranscribed data are abundant and easy to obtain, and our proposed solution brings solid WER improvements and it is not difficult to replicate.

Abstrakt

V této dizertační práci nejprve prezentujeme teorii trénování neuronových sítí pro rozpoznávání řeči společně s implementací trénovacího receptu 'nnet1', který je součástí toolkitu s otevřeným kódem Kaldi. Recept se skládá z předtrénování bez učitele pomocí algoritmu RBM, trénování klasifikátoru z řečových rámců s kriteriální funkcí Cross-entropy a ze sekvenčního trénování po větách s kriteriální funkcí sMBR. Následuje hlavní téma práce, kterým je semi-supervised trénování se smíšenými daty s přepisem i bez přepisu. Inspirováni konferenčními články a úvodními experimenty jsme se zaměřili na několik otázek: Nejprve na to, zda je lepší konfidence (t.j. důvěryhodnosti automaticky získaných anotací) počítat po větách, po slovech nebo po řečových rámcích. Dále na to, zda by konfidence měly být použity pro výběr dat nebo váhování dat – oba přístupy jsou kompatibilní s trénováním pomocí metody stochastického nejstrmějšího sestupu, kde jsou gradienty řečových rámců násobeny vahou. Dále jsme se zabývali vylepšováním semi-supervised trénování pomocí kalibrace kofidencí a přístupy, jak model dále vylepšit pomocí dat se správným přepisem. Nakonec jsme navrhli jednoduchý recept, pro který není nutné časově náročné ladění hyperparametrů trénování, a který je prakticky využitelný pro různé datové sady. Experimenty probíhaly na několika sadách řečových dat: pro rozpoznávač vietnamštiny s 10 přepsanými hodinami (Babel) se chybovost snížila o 2.5%, pro angličtinu se 14 přepsanými hodinami (Switchboard) se chybovost snížila o 3.2%. Zjistili jsme, že je poměrně těžké dále vylepšit přesnost systému pomocí úprav konfidencí, zároveň jsme ale přesvědčení, že naše závěry mají značnou praktickou hodnotu: data bez přepisu je jednoduché nasbírat a naše navrhované řešení přináší dobrá zlepšení úspěšnosti a není těžké je replikovat.

Contents

Acronyms

ASR	Automatic speech recognition
WER	Word error rate (evaluation metric)
PER	Phone error rate (evaluation metric)
AM	Acoustic model
LM	Language model
WFST	Weighted finite state transducer (automaton representing a graph with costs on arcs)
GMM	Gaussian mixture model
EM	Expectation maximization (training algorithm for GMM)
NN	Neural network
DNN	Deep neural network (NN with 'several' hidden layers)
RBM	Restricted Boltzmann Machine (an auxiliary model for DNN pre-training)
DBN	Deep Belief Network (a stack of RBMs, initializiation of hidden layers)
SGD	Stochastic gradient descent (training algorithm for NNs)
CE	per-frame cross-entropy (objective function for NN training)
MMI	Maximum mutual information (objective function)
MPE	Minimum phone error (objective function)
BMMI	Boosted maximum mutual information (objective function)
sMBR	State minimum Bayes risk (objective function)
MFCC	Mel-frequency cepstral coefficients (input features)
FBANK	Log Mel-filterbank output (input features)
PLP	Perceptual linear predictive analysis (input features)
LDA	Linear discriminant analysis
MLLT	Maximum likelihood linear transform (feature projection, also called Semi-tied covariance method)
fMLLR	Feature-based maximum-likelihood linear regression (speaker adaptation technique)

Notation

t, T	time indexing
k, K	NN-output class indexing
w_i, W	single word, word sequence
s, s_i, S	tied-state, tied-state from a sequence, a sequence of tide states
$\mathcal{L}, \pi$	lattice, a path from lattice
κ	acoustic scale applied to the per-frame likelihoods from the acoustic model
ϱ	graph scale applied to graph-costs in WFST graphs, representing lattices or recognition network
$\mathbf{x}$	single data-point (vector) of input features
$\mathbf{X}$	matrix with input features (composed from single vectors)
$\mathbf{h}$	hidden vector from neural network
$\mathbf{y}$	output vector from neural network
$\mathbf{W}$	matrix with synapses from single layer of neurons
$W_{i,j}$	single element from the matrix of synapses
$\mathbf{b}$	bias vector from single layer of neurons
σ	logistic sigmoid
$\mathbf{w}$	all neural network parameters reshaped in one big vector
M	size of mini-batch
N	number of inputs from neuron or neural network
L	identifier of specific neural network layer

$\alpha_{\mathsf{src}(a_j)}$, $\beta_{\mathsf{tgt}(a_j)}$	accumulation statistics in Forward-backward algorithm, which produces lattice-posteriors $\gamma(a_j)$
$\gamma(a_j)$	posterior probability of traversing through lattice-link a_j
$\gamma(t, s)$	posterior probability of being in tied-state s at time t
$\gamma(t, \bar{s}_t)$	posterior of a particular tied-state $\bar{s}_t$ from the best path of lattice
$\gamma(q, w)$	posterior probability of word w being at position q in decoder output
$\gamma(q, \bar{w}_q)$	posterior of a particular word $\bar{w}_q$ from the best path of lattice
λ	the lattice-scale, applied to re-scale the lattice posteriors by multiplying the link scores before the forward-backward algorithm is started
α	the exponential scale, applied to the confidences which are already extracted from the lattices

Chapter 1

Introduction

The lack of space becomes a limiting factor in the case of small devices, like smart-phones or tablets. These are usually equipped with touch-screens, however writing on them is not comfortable. Then, a good example of a cognitively loaded person would be the driver, who would like to make a call or to control the GPS-navigation. In both cases the interaction using natural speech is a good alternative to the traditional input methods.

Other successful uses of speech recognition are in the fields that traditionally involve dictating for documentary purposes. This is the case of medicine, courts, state administrative and parliamentary talks. Speech recognition might also be interesting for companies, where it can be used to keep track of internal meetings or it can be integrated into customer care systems.

Nowadays, speech recognition is commonly used to search the Internet from mobile devices. The speech is also an input interface to the personal assistants like Siri, Cortana or Google Now. Slowly, speech recognition is finding its way to our home gadgets like TV-sets, light switches, etc. A recent product is the question-answering machine Amazon Echo[1]. It is equipped with a speaker and a microphone array for improved robustness on distant speech. It can be asked about news or weather, it can read a Wikipedia page or play a music on demand. Technically, it is a hardware client, while the speech recognition and the dialogue management is running in a cloud server.

Although there was a tremendous research progress in the last few years, especially after the Deep Neural Networks were introduced, the current Automatic Speech Recognition (ASR) systems are not perfect. Although the recognition of formal talks, conversational telephone speech and meetings does have an acceptable performance ranging between 10-25% of word error rate, a big challenge remains the far-field speech recognition of informal speaking style, here the error rate ranges 35-50%. Then, an unresolved problem remains the ASR of recordings with overlapped speakers, recorded by 1 microphone. Another challenge might be the limited amount of noisy training data, the WER can increase up to 50-70%, which we observed while working on Babel program.

Another limitation of current ASR systems comes from the nature of its supervised machine-learning. To develop an ASR system for a new language, we need to carefully transcribe at least few hours of the training recordings. Then, for the language model training, we need a text corpus, ideally with vocabulary and speaking style similar to the target domain, in which the recognizer will be deployed. We also need a linguist to design a phone-set and create a pronunciation lexicon. Finally, the acoustic conditions in target

[1] https://www.youtube.com/watch?v=KkOCeAtKHIc

domain need to be similar to those in the training data, otherwise a mismatched acoustic model will cause performance degradation.

Fortunately, many of the problems are partially solved: The robustness is improved by various acoustic feature normalizations or by capturing the signal by microphone array and processing the multi-channel input (beam-forming, de-noising, source separation, ...). The pronunciation lexicon can be replaced by a graphemic one at the cost of small degradation in performance (typically few percents of WER). The text corpus can be prepared by cleaning downloaded web resources.

And finally, in the case of having only a little amount of the transcribed training data, we can improve the acoustic model by using the untranscribed data in the semi-supervised training, which is the main topic of this thesis.

1.1 Motivation

The state-of-the-art Automatic Speech Recognition (ASR) systems need carefully transcribed and thus expensive data to be trained on. It is therefore in the interest of the research community to search for such techniques that will help to reduce this 'cost barrier', and make the ASR technology more accessible both for commercial and non-commercial use.

The idea of semi-supervised training is to improve the system by using non-transcribed data. This is done by generating automatic transcripts along with their confidences, i.e. the probability of being correct. Then the performance is improved because of better acoustic models trained on more data. We are using a 'self-training' scenario, where a small part of data is transcribed manually. This allows us to train a 'seeding' system, which we use to generate the automatic transcripts, that we later use for the self-training.

The self-training of ASR systems has been studied extensively. However, at that time, the dominant acoustic models were GMMs (Gaussian Mixture Models) [Wessel and Ney, 2005, Wessel et al., 2001]. These are typically trained *generatively* by EM algorithm. In this model, each acoustic unit is described by a GMM, i.e. a probability density function estimated on its associated data-points. In the EM training, the models do not partition the feature space exclusively, so two acoustic units can have similar distributions. A minor part of wrong labels in the training data, may not have too bad influence on the final model.

However, the current state-of-the-art acoustic modeling is based on neural networks, which are usually trained *discriminatively* to classify feature frames into a closed set of acoustic units. Here the classification is exclusive, the posterior probability is sub-divided among the acoustic units, which makes the training potentially less robust to wrong labels.

For this reason, it is interesting to re-visit some of the older techniques and to use them as an inspiration for developing a self-training recipe for current state-of-the-art ASR systems.

1.2 Scope of the thesis

In this thesis is presented a systematic study of acoustic model self-training. The acoustic model is a feed-forward Deep Neural Network and I searched for answers to these crucial questions:

- What is the most suitable confidence granularity for the semi-supervised DNN training. Should we extract confidences: per-sentence, per-word, per-frame?
- How should we use the confidence, for data selection or for weighted training?

- What is the 'ideal confidence'? What should be its role in self-training?
- Is confidence calibration important?
- Do we need to further post-process the self-trained model by using the correctly transcribed data?
- Can we build a simple generic recipe that is applicable to different data-sets?

In the thesis I worked with feed-forward neural networks with sigmoid units. It is likely that the observations will generalize to other types of networks, although we did not particularly investigate this in detail.

The Kaldi 'nnet1' recipe

Before the experiments with semi-supervised training were started, I have developed and made publicly available the DNN training recipe *'nnet1'* as part of the toolkit *Kaldi*[2]. The design of this implementation was partially inspired by my previous project *TNet*[3]. Both tool-kits are used by other researchers in various laboratories or companies from all over the world.

The 'nnet1' recipe consists of Restricted Boltzmann Machine pre-training, the mini-batch frame classification training and the sequence-discriminnative sMBR training. The important aspects of the recipe are covered in chapter 3, which sources mainly from my own publications. This recipe represents a solid basis upon which the semi-supervised experiments are performed.

1.3 Original claims

1. In this thesis is performed an extensive study of semi-supervised training of DNN in which we use confidences extracted from a single ASR system.
2. I have carefully compared the scenarios in which the confidences are extracted per-sentence, per-word or per-frame. Along the way are also compared other state-of-the-art confidence measures, and it is shown that our preferred confidence is better.
3. From the results is apparent that standard 'sentence selection' approach provides only limited performance improvements. Better results are achieved either with selecting smaller units (words, frames) or from the use of weighted training with some appropriate scaling mechanism.
4. I also identified a simple rule for setting an optimal threshold in word-selection, which generalizes both for Babel Vietnamese and Switchboard English. The amount of words added in self-training is determined by word accuracy from development set. Such simple system is not far from our best recipe, which involves a time-consuming grid search over a hyper-parameter.

[2] http://kaldi-asr.org/doc/dnn1.html
[3] http://speech.fit.vutbr.cz/software/neural-network-trainer-tnet

Chapter 2

Introduction to Neural Network based speech recognition

In this chapter, we introduce the theory of speech recognition, the models that are used in the recognizer and the derivation of the back-propagation algorithm for neural network training. If the reader is already familiar with these topics, he/she may consider skipping this chapter.

2.1 The problem of speech recognition

According to the theory of Automatic Speech Recognition (ASR), the problem is to correctly recognize the sequence of words that corresponds to the 'observed' acoustic signal.

As illustrated in figure 2.1, the input is a speech signal, while the output is the recognized text. The processing is subdivided into 3 stages.

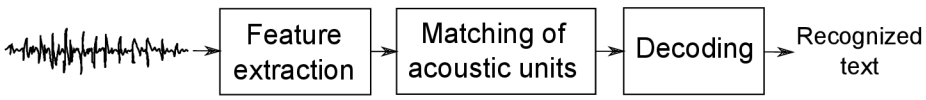


Figure 2.1: *Architecture of speech recognizer*

The purpose of **feature extraction** is to compress the waveform in a sequence of fixed-length vectors of low dimension. Usually, we extract one vector per a 10 ms step from 25 ms long chunks of speech signal, i.e. the *speech frames*. The encoding must preserve the information relevant for recognition and suppress the irrelevant information. For example, in the case of speech recognition, we try to suppress the differences across speakers, genders, dialects, microphones etc. The typical feature extraction is based on signal processing techniques such as filtering and Discrete Fourier Transform. Usually, Fourier spectrum is post-processed to obtain a representation convenient for the machine-learning models. Typically, the short-term spectrum is projected into a set of triangular filters, 'sitting' at different frequency ranges, and some further processing steps are performed. The most popular feature extraction methods for ASR are FBANKs (log-Mel filterbanks), MFCCs (Mel-frequency cepstral coefficients) and PLPs (Perceptual Linear Prediction).

In **Matching of acoustic units**, we 'convert' the features into scores of some closed set of *acoustic units*. For illustration, one can think of phonemes as units. The acoustic scores are computed by an *acoustic model*, usually a Gaussian Mixture Model (GMM) or a Deep Neural Network (DNN). Formally, each score is a likelihood $P(\mathbf{x}|s)$, i.e. the density

function value for the feature vector $\mathbf{x}$ given the identity of acoustic unit s. The acoustic models in general are trained on a set of speech recordings, the training is usually supervised by manual transcriptions. Naturally, better models are obtained when more training data is used. For a poor system, we need to have at least few transcribed hours, while up to hundreds of thousands of hours are used in some companies. For the training, the feature vectors need to be assigned to acoustic units. This is not done manually, but by using a *forced-alignment* to reference transcripts with some existing model. If there is no model yet, equal lengths are assigned to all acoustic units in an utterance.

In **Decoding**, we search for the most likely word sequence $\hat{W}$ that corresponds to the 'observed' sequence of feature vectors $\mathbf{X}$. This is done by a search in a huge graph of all the possible hypothesis, where we combine the scores from the acoustic model, language model and lexicon. A typical decoding algorithm is based on two ideas: *token passing* and *beam search*. The idea of token passing is a frame-by-frame cycle advancing in time over the input features, and for the current frame we have a stack of tokens with partial recognition paths. In the next frame, each token is expanded into many new tokens with the possible continuations of the hypothesis. To avoid having too many tokens, some of them are discarded. Only the tokens with scores within some margin from the best token survive; this is the idea of beam search. This local and greedy heuristic makes the speech recognition fast enough for practical use, however it can lead to a *search error*, if a more accurate path is discarded because its token 'fell-out' of the beam. Practically, the beam-width is the distance of the scaled log likelihoods of partial recognition hypotheses, and we should make sure the beam is large enough, so that its further extension does not improve the recognition results.

2.2 Models in speech recognition and decoding

Mathematically, the decoding is formulated as finding the word string $\hat{W}$ with the maximal *a posteriori* probability given the sequence of input feature vectors $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$:

$$\hat{W} = \arg\max_W P(W|\mathbf{X}) \ . \tag{2.1}$$

By using Bayes rule we can rewrite this expression as:

$$\hat{W} = \arg\max_W \frac{P(\mathbf{X}|W)P(W)}{P(\mathbf{X})} \ , \tag{2.2}$$

where the prior term $P(W)$ corresponds to the probability of the word sequence without using any acoustic information. Practically, this score is obtained from a *language model* (e.g. an n-gram) trained on a large text corpus. Then we have the likelihood term $P(\mathbf{X}|W)$ from the *acoustic model*, which is the score of the feature vector $\mathbf{X}$ given the word sequence W. The likelihood term $P(\mathbf{X}|W)$ involves the sum over all state sequences corresponding to our word-string W, while $P(\mathbf{x}|s)$ are the likelihoods of the acoustic units in those state-sequences. The normalization term $P(X)$ can be ignored as it is constant for any word/state sequence.

For practical reasons, the formulation in Kaldi is simplified to the search of the most likely state sequence S. This is mapped to the corresponding word sequence by the mapping function 'wrds':

$$\hat{W} = \text{wrds}\left(\arg\max_{S} \frac{P(\mathbf{X}|S)P(S)}{P(\mathbf{X})}\right) . \tag{2.3}$$

In other words, instead of getting the score of a word-string by marginalizing over all its possible state-sequences, only the best state sequence is considered in the decoding process. The decoder becomes simpler and faster.

2.2.1 Hidden Markov Model

Each acoustic unit from the pronunciation lexicon is expanded into HMM model as in figure 2.2. Because the durations of acoustic units differ and their pattern changes over time, we represent each of them by 3 state Hidden Markov Model (HMM), where the 3 states model the beginning, the middle part and the end of the acoustic unit. During the

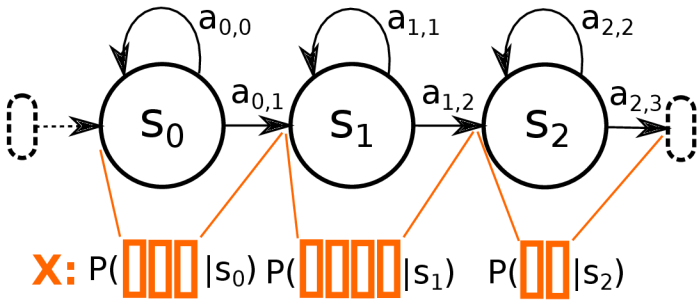


Figure 2.2: *3-state Hidden Markov Model, which models acoustic unit in ASR systems*

decoding, the token is passed through all the states, while it is allowed to stay in the state for several frames by traversing the self-loop transitions. Also note that the transitions have associated transition probabilities $a_{i,j}$, and the outgoing transition probabilities from each state sum up to 1. A score of a path through the simple HMM in figure 2.2 is the product of all the acoustic likelihoods $P(\mathbf{x}_t|s_i)$ and the transition probabilities $a_{i,j}$. In a more complicated HMM called *recognition network*, the scores on HMM-links will be a product of *transition probabilities*, *lexicon scores* and *language model scores*.

2.2.2 Hybrid acoustic model, context dependency, prior trick

As can be seen in a spectrogram (figure 2.3), speech is a continuum where one phoneme changes into another one without a clear boundary between the phonemes. Moreover, the realizations of phonemes are influenced by preceding and following phonemes, this influence is called *coarticulation*.

Inspired by this, a more precise modeling in ASR system is achieved by having context dependent phonemes, usually triphones. Each such unit is labeled as a triplet composed of the preceding, current and the following phoneme. With phone set size n, we get n^3 triphones. This can be a lot. For example, with 40 phonemes there are 64k units. Moreover, each triphone is described by a 3-state HMM, which further increases the overall number of context-dependent states. In practice, not all triphone combinations exist, and some may be very rare. Therefore, it is better to cluster the HMM-states corresponding to similar sounds into so called *tied-states*, which leads to a model that is easier to train. The tied-state clustering [Young and Woodland, 1994] is obtained by training a decision tree. It is

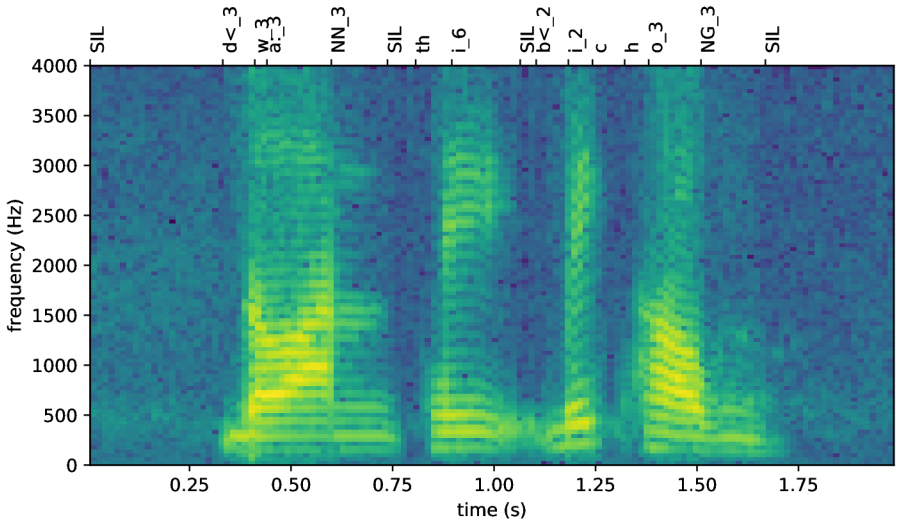


Figure 2.3: *Spectrogram of a sample Vietnamese expression. Note the phone-alignment on the top (from DNN).*

trained by a top-down greedy splitting, where, in each step, we add a split that maximally increases the likelihood of data.

Coming back to the decoding formula (2.2), the term $P(\mathbf{X}|S)$ is given as follows:

$$P(\mathbf{X}|S) = \Pi_{t=1}^{T} P(\mathbf{x}_t|s_t) \ , \tag{2.4}$$

where s_t is the HMM state generating the feature frame $\mathbf{x}_t$.

In case of 'hybrid setup' (i.e. HMM decoding of NN outputs), the deep neural network produces posterior probabilities of tied states $P(s|\mathbf{x})$, while the formulation of *maximum a posteriori* decoding expects the acoustic scores as likelihoods $P(\mathbf{x}|s)$. To convert the posteriors into pseudo-likelihoods, we use:

$$P(\mathbf{x}|s) = \frac{P(s|\mathbf{x})}{P(s)} \ , \tag{2.5}$$

where $P(s)$ is the prior probability of acoustic unit s. The $P(s)$ can be estimated as relative frequency of s in the set of training labels [Bourlard and Morgan, 1993]. Or alternatively by marginalizing $\mathbf{x}$ from the posteriors by $P(s) = E_{\mathbf{x}}[P(s|\mathbf{x})]$ on a representative set of training data [Zhang et al., 2014b].

2.2.3 Lattices in WFST format

As we will be generating confidences based on lattice-posteriors, we will describe them together with some characteristics arising from the 'exact lattice' generation [Povey et al., 2012]. The lattice is a graph for representing alternative hypothesis of speech recognition that is in Kaldi represented by two types of WFSTs, which can be mutually converted.

The type `Lattice` is a trellis with per-frame arcs, here one HMM transition corresponds to one arc in the lattice. The input symbol is *transition-id*, output symbol is *word-id* or ϵ, and the WFST weight is a tuple of acoustic score and graph score. This type is used mostly as an internal representation in the C++ code.

For storing lattices there is the type `CompactLattice`, it is a 'deterministic acyclic weighted acceptor' in which one arc corresponds to one word. Here, the input and output symbols are identical words, and the WFST 'weight' consists of: acoustic score, graph score and a sequence of transition-id's that is obtained over the word's duration.

Due to the determinization algorithm described in 'Generating exact lattices' [Povey et al., 2012], each distinct word sequence is present in lattice only once (i.e. lattice is deterministic), and with its best score. As a side effect of weight pushing, the positioning of scores and word-boundaries on a lattice path is not always properly synchronized in time with the original signal. The timing can be fixed by Kaldi tool `lattice-align-words-lexicon`.

2.2.4 Forward-backward algorithm for lattice

The forward-backward algorithm is used later for obtaining per-frame confidences of tied-states from `Lattice`. To get this, we first need to compute the *'responsibility'* $\gamma(a_j)$ representing a conditional probability of being in lattice-arc a_j, given some lattice $\mathcal{L}$. The identity of lattice-ark a_j encodes implicitly timing by length of any path leading to the arc, as in `Lattice`, each arc corresponds to one HMM transition. The *tied-state* of the arc is identified from *transition-id* in its input symbol.

The lattice link a_j is defined in Kaldi type `Lattice` by following elements:

$\mathsf{src}(a_j)$	source state in lattice from which the arc points out,
$\mathsf{tgt}(a_j)$	target state in lattice into which the arc is pointing,
$S_{input}(a_j)$	input symbol (transition id),
$S_{output}(a_j)$	output symbol (word id),
$(P_G(a_j), P_{AM}(a_j))$	WFST weight, tuple consisting of graph score and acoustic score (already scaled with graph scale ϱ and acoustic scale κ).

The posterior probability $\gamma(a_j)$ is then the *total probability* of crossing the arc a_j, computed as a ratio of score-sum of all paths that cross the arc (illustrated in figure 2.4) over the score-sum α_Ω of all paths in the lattice:

$$\gamma(a_j) = \frac{\alpha_{\mathsf{src}(a_j)}\, a_j\, \beta_{\mathsf{tgt}(a_j)}}{\alpha_\Omega} \,, \tag{2.6}$$

where $\alpha_{src(a_j)}$ are forward statistics computed as the sum of scores on all paths π in sub-lattice $\mathcal{L}_{\mathcal{I},\mathsf{src}(a_j)}$ that spans between initial state $\mathcal{I}$ and the source state of our arc a_j:

$$\alpha_{\mathsf{src}(a_j)} = \sum_{\pi \in \mathcal{L}_{\mathcal{I},\mathsf{src}(a_j)}} P(\pi) \,, \tag{2.7}$$

where a_j corresponds to the WFST weights on our link a_j:

$$a_j = P_G(a_j) P_{AM}(a_j) \,, \tag{2.8}$$

and where $\beta_{tgt(a_j)}$ are backward statistics computed as the sum of scores on all paths π in sub-lattice $\mathcal{L}_{\mathsf{tgt}(a_j),\Omega}$ that spans between the target state of our arc a_j and the final super-state Ω[1]:

$$\beta_{\mathsf{tgt}(a_j)} = \sum_{\pi \in \mathcal{L}_{\mathsf{tgt}(a_j),\Omega}} P(\pi) \,. \tag{2.9}$$

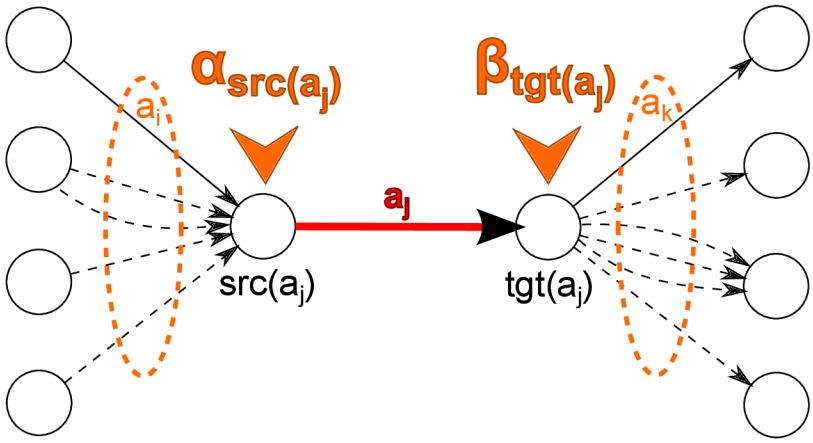


Figure 2.4: *Calculation of an arc-posterior $\gamma(a_j)$ in a lattice. Dynamic programming is used to obtain the statistics $\alpha_{\mathsf{src}(a_j)}, \beta_{\mathsf{tgt}(a_j)}$ calculated for all nodes in the lattice. The statistics represent the scores from all partial-paths leading into/from arc a_j.*

The efficient calculation of statistics $\alpha_{\mathsf{src}(a_j)}$, $\beta_{\mathsf{tgt}(a_j)}$ is done for each state in the lattice by means of *dynamic programming* according to recursions, which sum over all incoming arcs a_i or outgoing arcs a_k:

$$\alpha_{\mathsf{src}(a_j)} = \sum_{a_i \in \{a_i | \mathsf{tgt}(a_i) = \mathsf{src}(a_j)\}} \alpha_{\mathsf{src}(a_i)} P_G(a_i) P_{AM}(a_i) \ , \tag{2.10}$$

$$\beta_{\mathsf{tgt}(a_j)} = \sum_{a_k \in \{a_k | \mathsf{src}(a_k) = \mathsf{tgt}(a_j)\}} \beta_{\mathsf{tgt}(a_k)} P_G(a_k) P_{AM}(a_k) \ . \tag{2.11}$$

Recall that, with Kaldi type `Lattice`, the duration of each arc is exactly one data-point of input features, hence the time info can be computed from number of traversed arcs. The initial conditions for recursions are $\alpha_{\mathcal{I}} = 1$ for initial state $\mathcal{I}$, and $\beta_{\Omega} = 1$ for terminal super-state Ω. The corresponding final alpha is:

$$\alpha_{\Omega} = \sum_{f \in \mathcal{F}} \alpha_f \, P_G(f) \tag{2.12}$$

where $\mathcal{F}$ is set of all final states in the lattice, these have state weights $P_G(f)$. The α_{Ω} is our normalizer term from (2.6).

A similar algorithm is used in Baum-Welch training in HTK-book [Young et al., 2002]. However, Baum-Welch algorithm considers a trellis of many HMM paths through a word, while in our algorithm, we consider only the best HMM path of each word, as we use 'exact lattice' generation [Povey et al., 2012]. Other difference is that we defined the algorithm for calculating posteriors of arcs, while the original HTK definition of Forward-Backward algorithm produced posteriors of HMM-states.

For the semi-supervised experiments we are primarily interested in posterior probability of *acoustic units* (tied-states) denoted as $\gamma(t, s)$, meaning a posterior of tied-state s at time t. We use *transition model* to convert the posteriors of arcs $\gamma(a_j)$ to the posteriors of tied-states $\gamma(t, s)$. If several arcs are mapped to same tied-state at time t, we sum the posterior probabilities of all such arcs.

[1] Because a WFST lattice can have more final states, we added final super-state Ω.

In later chapters , we will use lattice-scale λ. Its purpose is to have a control over the 'uncertainty/sharpness' of the resulting lattice-posteriors. In our practical implementation, we use λ to simultaneously multiply the *acoustic scale* κ and *graphs scale* ϱ. This effectively exponentiates the scores of whole lattice-paths, making them closer or farther from the score of the best path, which translates to a change of 'sharpness' of posteriors probabilities.

2.3 The Backpropagation training algorithm

The backpropagation algorithm is the standard algorithm for neural network training. It can be found in the literature [Bishop, 2007], but we would like to explain it in our own illustrative way.

The principle of backpropagation algorithm is shown on the computation of the update (gradient) for a neural network with 1 hidden layer, while the extension to deeper networks is simple. We consider a network with sigmoid non-linearity and the classification output layer with softmax. The gradient is computed for single datapoint represented by an input feature vector $\mathbf{x}$ and its target vector with 1-of-K encoding $\mathbf{t} = [0\ 1\ 0\ \ldots\ 0]^T$, where the element '1' identifies the NN output of the 'correct' class.

Neural network as a feed-forward function

At first, we will define the neural network as a structured function with trainable parameters. Being a function, it maps the multidimensional inputs to the outputs. It is organized into 'layers', and the typical feed-forward network consists of several alternating linear and non-linear transformations. The smallest processing unit of a neural network is one *neuron*:

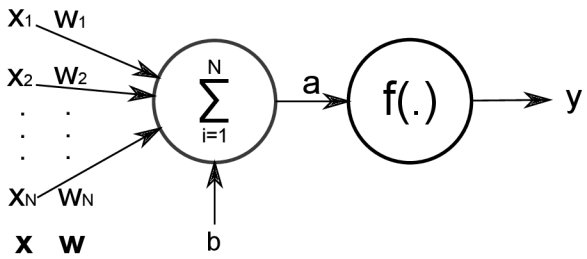


Figure 2.5: *Functional scheme of one neuron*

As illustrated in figure 2.5, the neuron has vector of inputs $\mathbf{x}$, and trainable parameters $\mathbf{w}$ and b. Functionally, it computes the activation a as a weighted combination of the inputs $\mathbf{x}^T\mathbf{w}$ plus bias term b. The output of the neuron y is obtained by transforming *activation* a with a differentiable non-linear *activation function* $f(.)$, which can be defined in many ways. The single neuron with logistic sigmoid activation function is capable of binary classification. The *neural network* is then composed of neurons going both into the width (parallel neurons in single layer) and the depth (serially connected layers of neurons).

Now let's return to our example network. As mentioned above, the activation of j-th *neuron* in the first layer is given as:

$$a_j = \mathbf{w}_j^T\mathbf{x} + b_j, \tag{2.13}$$

each input feature x_i has its weight $w_{i,j}$, and a bias b_j is added. For convenience, we can group all the neurons in the first layer and form a weight matrix $\mathbf{W}^{(1)}$, in which j-th row

is our weight vector $\mathbf{w}_j$. The biases are grouped into a vector $\mathbf{b}^{(1)}$. Then, the activation vector for all neurons in first layer is:

$$\mathbf{a}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \ . \tag{2.14}$$

Here we should emphasize that the input vector $\mathbf{x}$ is the same for all neurons in the first layer.

The key element which gives the neural network higher representative power than simple logistic regression is the nonlinearity in the hidden layers. Without the non-linearities, we could simply multiply all the weight matrices together to obtain the multi-class logistic regression with no hidden layers. A very popular nonlinearity is *logistic sigmoid*, which converts the numbers from interval $(-\infty, \infty)$ to probability-like numbers $[0, 1]$. The sigmoid is denoted with sigma σ :

$$h_j^{(1)} = \sigma(a_j^{(1)}) = \frac{1}{1 + \exp(-a_j^{(1)})} \tag{2.15}$$

The output of the sigmoid is the *hidden vector* $\mathbf{h}^{(1)}$, it can be seen as an intermediate encoding of the input features, on the way towards the output of the neural network. Usually we are not interested in the actual values there, which is why we call the layers 'hidden'. It is important that we can calculate the output of the neural network from these values.

Analogically to (2.14), the hidden vector $\mathbf{h}^{(1)}$ is transformed with another affine transform to the second layer activations $\mathbf{a}^{(2)}$:

$$\mathbf{a}^{(2)} = \mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)} \ . \tag{2.16}$$

From these, the posterior probabilities of classes y_i are computed using the softmax function:

$$y_i = \frac{\exp(a_i^{(2)})}{\sum_j \exp(a_j^{(2)})}, \quad y_i \in \mathcal{R}_0^+, \ \sum_i y_i = 1 \tag{2.17}$$

As illustrated in (2.17), the softmax function ensures that we always obtain positive quantities which sum-up to one.

To illustrate that our example neural network is one big structured function, we show the forward-propagation formula in which we compute the output $\mathbf{y}$ from the input vector $\mathbf{x}$:

$$\mathbf{y} = \text{softmax}\left(\mathbf{W}^{(2)}\ \sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right) \tag{2.18}$$

An alternative look at formula (2.18) is shown in figure 2.6, where the nodes represent individual neurons and arcs the interconnections. The first layer of neurons produces the hidden vector $\mathbf{h}$, while the second layer produces the output $\mathbf{y}$. The trainable parameters are weight matrices $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ and bias vectors $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}$. In this example, all the functions we used are differentiable, so we can calculate the partial derivatives for the backpropagation algorithm. In fact, the functions are also smooth, which is however not strictly required for the training algorithm.

A related interesting question is: What is the set of functions that can be represented by a neural network? The studies from Cybenko [Cybenko, 1989] and Barron [Barron, 1993] show that theoretically, they can approximate any continuous function with arbitrary precision, if

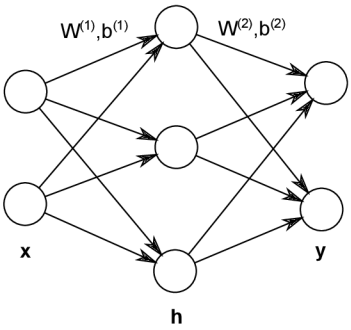


Figure 2.6: *Structure of neural network, the nodes represent neuron outputs.*

the neural network with one hidden layer has sufficient number of sigmoid neurons. Although this work does not say how to train them, the statement is very promising.

From the practical experience, it is commonly agreed that it is advantageous to use many hidden layers, which is the origin of the very frequently used term *'Deep Neural Networks'*. In our recipes, we usually use 6 hidden layers. A popular explanation is that with more layers we better smooth out the feature variability that is irrelevant to the classification task, and that the layers near the network output encode more complex features. However, this intuitive view is not easy to be validated practically.

Loss function, multi-class cross-entropy

Before we can derive the backpropagation training, we need to define a loss function to optimize. For the classification of n-th data-point into 1-of-K mutually exclusive classes, the natural loss function is the multi-class cross-entropy (CE):

$$E_n(\mathbf{w}) = -\sum_{k=1}^{K} t_k \ln y_k \ . \tag{2.19}$$

Recall that vector $\mathbf{t}$ has 1-of-K encoding $\mathbf{t} = [\,0\ 1\ 0\ \ldots\ 0\,]^T$, so that the sum picks-up the k-th element, which corresponds to the sole non-zero target. The loss value is always positive and reaches zero minimum, if the posterior vector $\mathbf{y}$ exactly matches the target vector $\mathbf{t}$. (note that in this section we denote time-id with subscript n to avoid confusion with training labels $\mathbf{t}$, in other chapters time is denoted with t subscript)

In the more general case, when the 'soft' probabilistic targets are used (the 1-of-K encoding of $\mathbf{t}$ is replaced by a vector with positive values, which sum-up to one), it is convenient to replace the cross-entropy with KL-divergence:

$$E_n(\mathbf{w}) = D_{KL}(\mathbf{t}||\mathbf{y}) = -\sum_{k=1}^{K} t_k \ln \frac{y_k}{t_k} \tag{2.20}$$

the difference is that the KL-divergence subtracts the entropy of the target labels and reaches its zero minimum when $\mathbf{y} = \mathbf{t}$, while the cross-entropy would have a non-zero value if $t_k \notin \{0, 1\}$. With targets $\mathbf{t}$ in 1-of-K encoding, both functions become the same. The derivatives with respect to neural network outputs are the same for both functions, and in Kaldi 'nnet1' we implemented (2.20).

When training on a data-set composed of T data-points, the overall loss is the sum of the per-frame values:

$$E = \sum_{n=1}^{T} E_n \ . \tag{2.21}$$

An interesting value for the log-prints is the per-frame average $\bar{E} = E/T$, which can be converted to the geometrical-average posterior value of the correct class by $\bar{p}_{corr} = \exp(-\bar{E})$, if we assume to have the 1-of-K targets.

Stochastic gradient descent, update rule

The most popular training algorithm for neural networks is *Stochastic Gradient Descent* (SGD). The other frequent term 'backpropagation' refers to the way how the SGD gradient is computed from the neural network.

The idea behind *gradient descent* training is to greedily minimize the loss by doing small parameter steps in the direction of the opposite gradient, i.e. the direction of steepest descent of the loss.

Stochastic Gradient Descent training is a gradient descent, in which the model is updated on-line after processing a single or a small group of randomly selected training data-points. It is possible that, while the loss decreases on some samples, we can see a loss increase for other samples. The overall trend 'steers' the model towards the regions, where the loss is low. Hence, the training progress is noisier and more 'exploratory', and this lowers the risk of converging to a poor local minimum.

Due to practical reasons, we usually do mini-batch SGD training, in which we calculate the gradients for M data-points together (default $M = 256$). The data-points are grouped into matrices where, for example, the matrix-vector multiplication in (2.18) becomes matrix-matrix multiplication. This can better employ hardware by reusing data elements in cache and better caching accelerates the passes through training data (i.e. *epochs*) considerably.

The update formula for the mini-batch SGD is:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \sum_{n=1}^{M} \nabla E_n(\mathbf{w}^{(\tau)}) \ , \tag{2.22}$$

where $\mathbf{w}^{(\tau)}$ is a vector with all the trainable parameters, and $\nabla E_n(\mathbf{w}^{(\tau)})$ is the gradient of E_n w.r.t. $\mathbf{w}^{(\tau)}$ calculated on a single data-point $\mathbf{x}_n$. Note that we sum the gradients from M data-points, and that η is *learning-rate*, i.e. a scalar controlling step-size during training. The suitable learning-rate value needs to be tuned carefully and depends on many factors (network-type, non-linearity type, loss type, size of mini-batch, per-utterance training, etc). By the nature of SGD training, the updates of the parameters are 'noisy' with a 'correct' global trend. Therefore, we can see the learning-rate as the temperature of simulated annealing. By controlled 'cooling' (i.e. decreasing) of the learning-rate, we can reduce the exploration while approaching the end of the training and let the network converge to a better solution.

Due to limited space, we skip the deriving of gradient ∇E_n. For those who are interested, it is presented in the full manuscript of the thesis ...

Chapter 3

Kaldi 'nnet1' DNN training recipe

After the introduction, we proceed with the description of the DNN training recipe that we implemented into the open-source toolkit Kaldi. In this chapter, we first cover the general experimental setup and continue with sections describing RBM pre-training, mini-batch frame cross-entropy training and sequence-discriminative sMBR training. The chapter ends with notes on the scalability of the mini-batch training.

Acoustic units

As mentioned earlier, the DNN acoustic model provides the posterior probabilities $P(c|\mathbf{x})$ for a closed set of acoustic units. The typical acoustic units are the clustered states from 3-state HMMs, which model the context-dependent (CD) phonemes. Such units are often referred to as *tied-states*, *CD-states* or *senones*. The clustering is determined by a decision tree, which is built by a greedy rule that adopts the splits with the best increase of the data likelihood. Depending on the size of training set there are usually thousands of CD-states.

The neural network is trained to classify them exclusively, the Softmax function from equation (2.17) in the output layer ensures that the posteriors of all the acoustic units are non-negative and sum to one. For decoding, the posteriors are converted to pseudo-likelihoods $P(\mathbf{x}|c)$ by dividing them with priors $P(c)$ as illustrated earlier in equation (2.5).

Input features

There are many ways how to prepare the input features for a neural network, and new feature extraction methods are published at every conference. Usually, the feature vectors on NN input cover time period 150-300ms, and they are assembled from short-term feature vectors. The short-term features are typically computed from 25ms frames of speech extracted with 10ms steps.

The features we use are the PLP-fMLLR speaker-adapted features. To produce them, we need an initial GMM-HMM system, which is used to estimate the speaker specific linear transform fMLLR.

The feature extraction pipeline in figure 3.1 shows that we begin from the PLP+pitch short-term features. Then, there are two stages, the first with a GMM model and the second with DNN model.

The GMM-HMM features are obtained by splicing 9 frames of the 'short-term' feature vectors (4 on each side from the 'current' frame). The short-term features are 13-dimensional PLPs (including C0) extended by 3 Kaldi-pitch features [Ghahremani et al., 2014] (probability of voicing, pitch, delta pitch), which are both mean-variance normalized by CMVN.

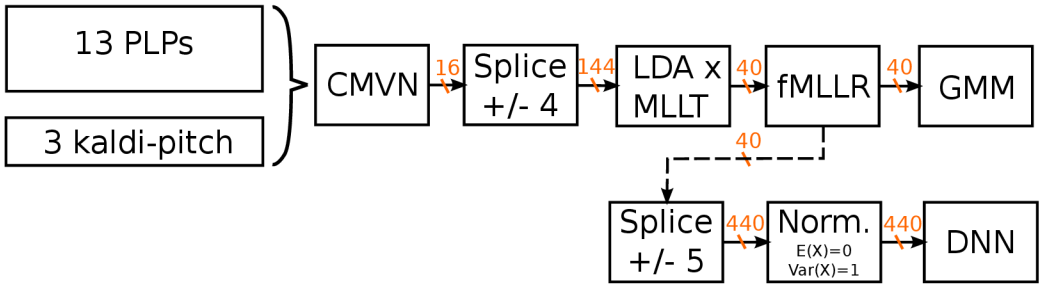


Figure 3.1: *Input features of the DNN.*

Our telephone speech has 8kHz sampling frequency, so the PLPs are computed from the frequency range 125-3800Hz and we use dithering. The spliced features were projected down to 40 dimensions using a global LDA-MLLT linear transform and per-speaker fMLLR linear transform. The fMLLR transform is obtained with a GMM trained in the adapted feature-space. For test data, the transformation is obtained by the multi-pass decoding with the GMM model.

Then, for the DNN input, we splice 11 frames of the 40-dimensional fMLLR features (5 on each side of the current frame), and we rescale them globally to have zero mean and unit variance. The total dimension of DNN input is 11 x 40 = 440.

This setup is used in most of the experiments, the exceptions are described locally in the text. Eventually, we can obtain a small improvement in recognition accuracy by replacing the LDA+MLLT+fMLLR linear transform with a non-linear bottleneck network and fMLLR. However this would make the analysis of semi-supervised DNN training more difficult, so we keep using this simpler setup.

NN topology, initialization

The typical neural network we use in this thesis has 6 hidden layers of 2048 Sigmoid neurons. The neural network has 440 inputs (the spliced fMLLR features) and thousands of Softmax outputs (for example for Vietnamese 4599 outputs).

There are three ways to initialize a neural network, either with a) small random numbers or by b) unsupervised pre-training with RBMs or by c) supervised discriminative pre-training. In our recipe, we **use the RBM pre-trainig**, which allows us to use an *'universal'* topology and obtain good results for many training sets (except the really small ones for which we need to use smaller network).

Data randomization, mini-batch training

The idea of Stochastic Gradient Descent training (SGD) is to randomly draw samples from the distribution of the training data and perform small updates of model parameters in the opposite direction of the gradient of a loss function, which decreases the loss.

Having a training data-set of finite size, the *'sampling'* is usually understood as randomizing the order in which the data is used for training. Hence, we typically shuffle the list of training sentences. Then, for the mini-batch training as introduced in equation (2.22) and also for the RBM pre-training, there is an additional frame-level shuffling mechanism inside the training tools. We can use the fixed 'random' order for all the epochs, usually it has little effect on the final results and the experiments become replicable.

For the recurrent networks, we need a different approach. We need to keep the continuity

of the sentence, so we cannot shuffle the speech frames. Instead, we can process several sentences in parallel, which accelerates the training as more frames are processed in one step.

3.1 Training the DNN acoustic model

The training recipe consists of RBM pre-training, frame classification training with cross-entropy loss function and sequence discriminative training with sMBR loss function. All the three steps are described in the following sections.

3.1.1 Pre-training with Deep Belief Network (Restricted Boltzmann Machines)

The Deep Belief Networks (DBN) were a hot topic in the DNN based speech recognition in 2010. The model and its theory were developed in the laboratories of Geoffrey Hinton [Hinton et al., 2006] and Yoshua Bengio [Bengio et al., 2007]. We can see the DBN pre-training as one of the regularization methods as it both reduces over-fitting and improves the results, when compared to the randomly initialized network. A great source of DBN related information is the 'Practical guide' from Geoff Hinton [Hinton, 2012], which we used as a basis for our implementation.

3.1.2 Frame classification mini-batch training

After the pre-training of the DBN, we append to it a randomly initialized output layer and continue with the frame classification training with *multi-class cross-entropy* (CE) loss function from eq. (2.20) (actually, the cross-entropy is replaced with KL-divergence, which has the same derivative and even loss value in case of 1-of-K targets).

Although we have already described the mathematical core of the mini-batch frame classification training in section 2.3, some practical parts were not yet covered and will be presented in this section. As mentioned earlier, the idea behind mini-batch stochastic gradient descent is to reduce the value of a loss function by updating the model parameters with small noisy steps, which are taken in the direction in which the loss decreases the most according its first order derivative (i.e. the opposite gradient). The gradient is each time computed from a small group of randomly selected data-points (i.e. the mini-batch). The individual updates are noisy, while we assume that the overall trend of the updates will 'steer' the model in a good direction.

This supervised learning trains the model to classify the speech-frames (data-points) into the correct classes (usually triphone states). For input vector $\mathbf{x}$, it provides its posterior probability $p(s|\mathbf{x})$. Each data-point is considered as an independent classification trial with an equal weight, regardless of the prior frequency of the classes.

3.1.3 Sequence-discriminative training, sMBR

This section is based on [Veselý et al., 2013a], where we studied the sequence-discriminative DNN training with various objective functions.

Neural networks (NNs) for speech recognition are typically trained to classify individual frames based on a cross-entropy criterion, equation (2.19). Speech recognition, however, is inherently a sequence classification problem. As such, speech recognizers using the Gaussian

mixture model (GMM) as the emission density of an HMM achieve the state-of-the-art performance when trained using the sequence-discriminative criteria like maximum mutual information (MMI) [Bahl et al., 1986], boosted MMI (BMMI) [Povey et al., 2008], minimum phone error (MPE) [Povey, 2003] or minimum Bayes risk (MBR) [Kaiser et al., 2000, Gibson and Hain, 2006, Povey and Kingsbury, 2007]. It is possible to efficiently estimate the parameters based on any of these criteria using the statistics collected from lattices [Povey, 2003].

The theory for sequence-discriminative training of neural networks was also developed in the early literature [Bridle and Dodd, 1991, Krogh and Riis, 1999]. In fact, the 'clamped' and 'free' posteriors described in [Bridle and Dodd, 1991] are the same as the numerator and denominator occupancies used in discriminative training of GMM-HMM systems [Povey, 2003]. The idea to use this lattice-based framework for sequence-discriminative training of NNs was explored in [Kingsbury, 2009]. It was shown that the sequence-discriminative training can improve upon networks trained using the cross-entropy. Subsequent results reported in [Wang and Sim, 2011, Kingsbury et al., 2012, Jaitly et al., 2012] have also shown consistent gains from sequence-discriminative training of NNs. However, there is some disagreement about which of the criteria is suitable: [Kingsbury, 2009, Kingsbury et al., 2012] suggest using a state-level minimum Bayes risk (sMBR) criterion, while [Wang and Sim, 2011] finds MMI to work better than MPE, and [Jaitly et al., 2012] only provide results using MMI.

Needless to say, such empirical observations depend on the choice of the dataset and specific details of the implementation. In our work, we presented a comparison of the different training criteria for DNNs on the standard 300-hour Switchboard conversational telephone speech task, which has also been used in [Seide et al., 2011, Kingsbury et al., 2012].

The networks are trained to optimize a given training objective function using the standard *error backpropagation* procedure [Rumelhart et al., 1986], and the optimization is done through stochastic gradient descent (SGD). For any given objective, the important quantity to calculate is its gradient with respect to the activations at the output layer. The gradients for all the parameters of the network can be derived from this one quantity based on the back-propagation procedure described in section 2.3.

Maximum mutual information, MMI

The MMI criterion used in ASR [Bahl et al., 1986] is the mutual information between the distributions of the observation and word *sequences*. With $\mathbf{O}_u = \{\mathbf{o}_{u1}, \ldots, \mathbf{o}_{uT_u}\}$ as the sequence of all observations, and W_u as the reference word-sequence for utterance u, the MMI criterion is:

$$\mathcal{F}_{MMI} = \sum_u \log \frac{p(\mathbf{O}_u|S_u)^\kappa P(W_u)}{\sum_W p(\mathbf{O}_u|S)^\kappa P(W)}, \tag{3.1}$$

where $S_u = \{s_{u1}, \ldots, s_{uT_u}\}$ is the sequence of states corresponding to W_u; and κ is the acoustic scaling factor. The sum in the denominator should be evaluated over all possible word-sequences W, but practically, it is computed from all paths through a *denominator lattice* generated for utterance u. Differentiating (3.1) w.r.t. the log-likelihood $\log p(\mathbf{o}_{ut}|r)$

for state r, we get:

$$\begin{aligned}\frac{\partial \mathcal{F}_{MMI}}{\partial \log p(\mathbf{o}_{ut}|r)} &= \kappa\delta_{r;s_{ut}} - \frac{\kappa \sum_{W:s_t=r} p(\mathbf{O}_u|S)^\kappa P(W)}{\sum_W p(\mathbf{O}_u|S)^\kappa P(W)}, \\ &= \kappa(\delta_{r;s_{ut}} - \gamma_{ut}^{DEN}(r)),\end{aligned} \tag{3.2}$$

where $\delta_{r;s_{ut}}$ is the Kronecker delta function, which equals 1 for state r at reference state sequence s_{ut}, and $\gamma_{ut}^{DEN}(r)$ is the posterior probability of being in state r at time t, computed over the denominator lattices for utterance u. The required gradient w.r.t. the activations is obtained as:

$$\begin{aligned}\frac{\partial \mathcal{F}_{MMI}}{\partial a_{ut}(s)} &= \sum_r \frac{\partial \mathcal{F}_{MMI}}{\partial \log p(\mathbf{o}_{ut}|r)} \frac{\partial \log p(\mathbf{o}_{ut}|r)}{\partial a_{ut}(s)}, \\ &= \kappa(\delta_{s;s_{ut}} - \gamma_{ut}^{DEN}(s)).\end{aligned} \tag{3.3}$$

Note that, in this work, we have assumed that the reference state labels are obtained through a forced alignment of the acoustics with the word transcript. More generally, one may use forward-backward over the word reference to obtain the numerator occupancies $\gamma_{ut}^{NUM}(s)$ instead of using $\delta_{s;s_{ut}}$ in equation (3.3).

Minimum phone error, MPE / State minimum Bayes risk, sMBR

While minimizing $\mathcal{F}_{CE}$ (2.19) minimizes expected frame-error, maximizing $\mathcal{F}_{MMI}$ minimizes expected sentence error. The MBR family of objectives are explicitly designed to minimize the expected error corresponding to different granularity of labels [Gibson and Hain, 2006]:

$$\mathcal{F}_{MBR} = \sum_u \frac{\sum_W p(\mathbf{O}_u|S)^\kappa P(W) A(W, W_u)}{\sum_{W'} p(\mathbf{O}_u|S)^\kappa P(W')}, \tag{3.4}$$

where $A(W, W_u)$ is the raw accuracy, representing the number of correct phone labels (for MPE) or state labels (for sMBR). The raw accuracy is counted for a path from some word sequence W that is compared with a path from the reference transcripts W_u. By differentiating (3.4) w.r.t. $\log p(\mathbf{o}_{ut}|r)$, we get:

$$\begin{aligned}\frac{\partial \mathcal{F}_{MBR}}{\partial \log p(\mathbf{o}_{ut}|r)} &= \kappa\, \gamma_{ut}^{DEN}(r) \left\{ \bar{A}_u(s_t = r) - \bar{A}_u \right\}, \\ &= \kappa\, \gamma_{ut}^{MBR}(r),\end{aligned}$$

where $\bar{A}_u(s_t = r)$ is the average accuracy of all paths in the lattice for utterance u that pass through state r at time t; $\bar{A}_u$ is the average accuracy of all paths in the lattice; and $\gamma_{ut}^{MBR}(r)$ is the MBR 'posterior' as defined for approximate MPE in [Povey, 2003]. Like before for $\mathcal{F}_{MMI}$, we get:

$$\frac{\partial \mathcal{F}_{MBR}}{\partial a_{ut}(s)} = \kappa \gamma_{ut}^{MBR}(s). \tag{3.5}$$

Finally, table 3.1 summarizes the results of the different systems trained on the entire 300 hour training set. The results are presented on both the development set (Hub5 '00) and the test set (Hub5 '01) and their respective subsets. We see that the CE trained DNN

Table 3.1: *Results (% WER) of the DNNs trained on the full 300 hour training set using different criteria. The input features are always the same: MFCCs transformed by LDA+MLLT+fMLLR.*

	Hub5 eval'00			Hub5 eval'01			
System	SWB	CHE	Total	SWB	SWB2P3	SWB-Cell	Total
GMM	21.2	36.4	28.8	-	-	-	-
GMM BMMI	18.6	33.0	25.8	18.9	24.5	30.1	24.6
DNN CE	14.2	25.7	20.0	14.5	19.0	25.3	19.8
DNN MMI	12.9	24.6	18.8	13.3	17.8	23.7	18.4
DNN sMBR	12.6	24.1	**18.4**	13.0	17.7	22.9	**18.0**
DNN MPE	12.9	24.1	18.5	13.2	17.7	23.4	18.2
DNN BMMI	12.9	24.5	18.7	13.2	17.8	23.5	18.3

models are better than the discriminatively trained GMM BMMI models. Then, the use of the sequence-discriminative training criteria (incl. lattice re-generation after first epoch) led to performance improvements within the range of $1.2 - 1.8\%$, and a little better results were achieved with the sMBR objective. The sMBR training was subsequently adopted as the default sequence-discriminative objective in the 'nnet1' training recipes in Kaldi.

3.2 Accelerating the DNN training

The acceleration of NN training was the main topic of my Master thesis [Veselý, 2010], which was later summarized in [Veselý et al., 2010]. Although already six years passed and the project TNet was abandoned, the gained experience was important for designing the 'nnet1' training tools in Kaldi.

In 2016, we compared again the training speeds of 1 CPU-core and 1 GPU with the current hardware. This time, we used Kaldi 'nnet1' training of a DNN with 8 million parameters on 10k sentences from AMI corpus. The GPU model was GTX980 and the CPU was Intel Xeon E5-2670. We used the OpenBLAS library for CPU training.

From table 3.2, we see that the speedup from using a GPU instead of 1 CPU-core is much higher than what we measured in 2012 (60x vs. 14x). This practically shows that the 'computation capabilities' of GPUs grew faster than those of CPUs. It is true, that we did the comparison with slightly different conditions (different NN topology, front-end, training toolkit), on the other hand the typical neural networks we train now are larger than those in the past.

Table 3.2: *Comparing the speeds of NN training with 1 CPU-core and 1 GPU. The reported time is the average duration of 13 epochs with 10k sentences.*

CPU-core	GPU	Ratio
264 min	4.4 min	60x

Chapter 4

Data-sets

In this chapter we provide a brief description of the databases we will later use for the experiments with the semi-supervised DNN training: Babel Vietnamese, some other Babel languages (Assamese, Bengali, Haiti, Lao, Zulu) and Switchboard English. We also mention some details about the experimental setups: language models, lexicons, phone-sets, OOV rates.

4.1 Babel Vietnamese

Most of the experiments with semi-supervised training were done with the Vietnamese dataset[1] as provided within the IARPA Babel program, release babel107b-v0.7. The training data consist of a large portion of conversational telephone speech and a small part of prompted speech. For training, we used both types of data. The development set consists of conversational speech only. The data come from various telephone channels: landlines, different kinds of cellphones, or phones embedded in vehicles. The sampling rate is 8000 Hz.

Two scenarios are defined – Full Language Pack (FullLP), in which all the collected data is transcribed; and Limited Language Pack (LimitedLP), in which only a subset of the data is transcribed, while the remaining part of the FullLP data can be used as 'untranscribed' data for the semi-supervised training.

The overview of the data (i.e. numbers of speakers and amounts of speech data after re-segmenting) is in table 4.1. We generated our segmentation, using our own MLP-based Voice activity detection (VAD) with Viterbi smoothing [Ng et al., 2012]. The speech segments were extended by 300 milliseconds on both ends.

The provided Vietnamese lexicon uses 54 phonemes. There are 25 consonants and 29 vowels, while for Vietnamese, we distinguish 6 tones.

The corpus is composed of 4 dialects, the pronunciation of some graphemes is different

[1] Collected by Appen Butler Hill: http://www.appenbutlerhill.com

Table 4.1: *Data analysis, numbers of speakers, amounts of annotated speech data after resegmentation by VAD*

Dataset	FullLP	LimitedLP	dev
speakers	991	121	120
size in hours (reseg.)	84.8	10.8	9.8

between dialects, a single grapheme can have 2-3 different vocalizations. Also, some of the phonemes can be translated into graphemes in several different ways.

For the purpose of ASR training, the phone set consists of 29 phonemes, which are marked with six different tones. The under-represented phones were merged manually. For the triphone-tree clustering, we introduced a 'position in a word' feature, which leads to the final phone-set with 350 items. We allow state sharing across phonemes.

The original syllabic lexicon provided by Appen was modified by reducing the number of pronunciation variants. The FullLP lexicon contains 6k syllables and the LimitedLP lexicon contains 3k syllables.

The ASR outputs are syllables, which is natural for Vietnamese and which conveniently avoids eventual errors from inconsistent word-segmentation. Also, there are no phonological processes that cross syllable boundaries, such as consonantal assimilation, tone sandhi or wordlevel stress. The consequence is that the OOV rate is very small, 0.21% for the FullLP condition and 1.19% for the LimitedLP condition (LimitedLP is used in semi-supervised training).

We used a trigram language model with Kneser-Ney smoothing built on the syllabic training transcripts, with 100k 3-grams and 200k 2-grams for FullLP, and with 12k 3-grams and 47k 2-grams for LimitedLP.

4.2 Switchboard

The Switchboard database consists of Conversational Telephone Speech. The training set is Switchboard-1 Release 2 (LDC97S62), a collection of about 2,400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from all areas of the United States. We used the Mississippi State transcripts and lexicon. The language model is built by interpolating two 3-gram language models trained on Switchboard and Fisher transcripts respectively.

Semi-supervised experiments For the semi-supervised experiments, the LM was built purely on the Fisher transcripts. Note that we generate automatic transcripts for the training data, so the true transcripts of the Switchboard data have to be removed from the LM corpus.

For the semi-supervised experiments, we split the Kaldi `'train_100k_nodup'` 100hour set, from which we randomly selected 186 conversation sides as the transcribed set (14 hours), while the remaining 1165 conversation sides (96 hours) are the untranscribed data.

Evaluation set: Hub5-2000 (eval2000) The evaluation set consists of: a) 20 conversations from the CallHome corpus, b) 20 conversations that were collected for the Switchboard Corpus but not included in the original release. Most of the speakers in these conversations, appeared in the released Switchboard Corpus for the training.

In this thesis we report the performance on both subsets together, while the conference articles from other laboratories usually report results for the b) subset. For more information see: http://www.itl.nist.gov/iad/mig/tests/ctr/2000/h5_2000_v1.3.html

Chapter 5

Semi-supervised training

This chapter is a gentle introduction to the semi-supervised training. It explains the basic pattern of improving the system with unlabeled data. We give an overview of the main design questions that we address in the following chapters. We also show the principle of frame-weighted mini-batch SGD training. The chapter is closed with a survey of the relevant literature.

The practical value of semi-supervised training is that it allows us to build better systems with an inexpensive untranscribed data, while we need to find a way how to use such data efficiently.

5.1 Definition

The *semi-supervised learning* is a type of supervised learning, where both the labeled and the unlabeled data are used. The goal of semi-supervised learning is to improve the system performance by adding the unlabeled data into the training process (compared to the case when only the labeled data are used).

We are using the heuristic approach called *self-learning*. In this case, a 'seed system' is built on the labeled data. The 'seed system' is then used to guess the labels for the unlabeled data. Next, a new system is built using the augmented dataset, while we typically add only the data where we are confident about the guessed labels.

Applied to ASR, we focus on semi-supervised training of Deep Neural Network acoustic models, which was not yet studied extensively. The process of semi-supervised system building is shown in figure 5.1, and is described as follows: We use the transcribed data to train the seed system. Then we generate the automatic transcripts and their confidences for the untranscribed data by decoding it with the seed system. The data with more reliable automatic transcripts are selected for the system re-training, where the confidences can be calculated in many ways. Lastly, the process of decoding and re-training can be iterated until no further improvements are obtained. This was done for GMM-HMMs in [Wessel and Ney, 2005].

5.2 The key questions of semi-supervised DNN training

When thinking about the semi-supervised training for the DNN models, we first aim to identify the questions, which help us establish the search space for finding a good semi-supervised recipe.

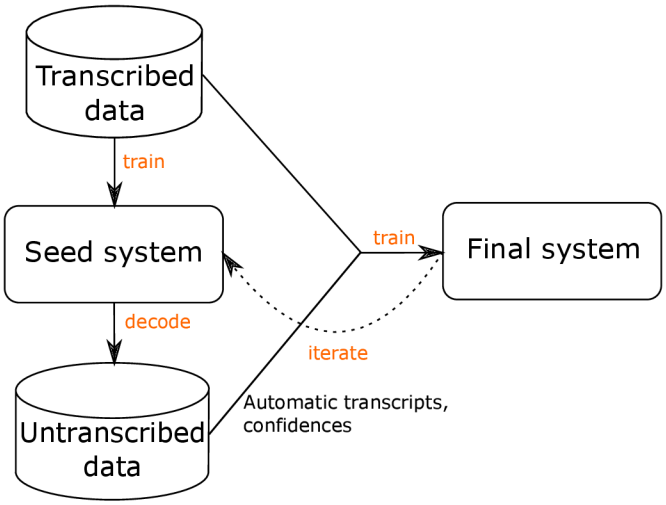


Figure 5.1: *General paradigm of Semi-supervised training for ASR.*

5.2.1 Granularity of confidence units

The first question is: „What should be the size of the unit for which we calculate the confidence?“ It can be a *sentence*, a *word* or a *feature-frame* (i.e. the smallest unit).

Per-word confidence $c_{\bar{w}_q}$

There are many methods how to extract the per-word confidences. For example, C_{max} [Wessel et al., 2001] is based on summing the posteriors of lattice-links that both correspond to the same word and overlap in time. Within the word-link, we take the value from such time-slice for which the sum of posteriors is the highest. However, this seems to be less necessary because we use 'exact lattice' generation [Povey et al., 2012]. The lattice is 'deterministic': each distinct word string is present in lattice only once and with its best score.

The method that we are using in our experiments is the calculation of statistics $\gamma(q, w)$ taken from the Minimum Bayes Risk (MBR) decoding [Xu et al., 2011, section 7.1]. The quantity $\gamma(q, w)$ is the posterior probability of the word symbol w being aligned with the position q in a word sequence, given lattice $\mathcal{L}$. In our case, we purposely fix word sequence to be from the best path in lattice $\bar{\pi}$: $\bar{W} = \text{wrds}(\bar{\pi}) = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_M)$, and the confidence score is $c_{\bar{w}_q} = \gamma(q, \bar{w}_q)$. This MBR confidence is the default word confidence implemented in Kaldi.

Yet another method to obtain word-confidence is based on the averaging of the neural network log-posteriors selected with the one-best state-sequence corresponding to the recognized word [Zhang et al., 2014a].

Ideally, a well calibrated word-confidence should correspond to the probability that the word is correctly recognized. The experiments with per-word confidences are in section 6.3.

Per-sentence confidence c_{sent}

In works of other authors, the sentence confidences are usually computed as arithmetic mean of the per-word confidences [Novotney et al., 2009, Novotney and Schwartz, 2009, Thomas et al., 2013, Zhang et al., 2014a]. It is better to think about it as the estimate of the *word*

accuracy in the sentence, rather than the correctness of the whole sentence. The supporting arguments for this interpretation are:

- a long sentence with one incorrect word can still be valuable for SST
- the word accuracy is closely related to the word error rate, which is the main evaluation metric for ASR systems

The experiments with per-sentence confidences are in section 6.2.

Per-frame confidence $c_{\bar{s}_t}$

In our work [Veselý et al., 2013b], we advocated for using frame-level confidence to do the frame-selection in mini-batch SGD training.

The per-frame confidence $c_{\bar{s}_t}$ is taken from the lattice posterior $\gamma(t, s)$, which is obtained by the forward-backward algorithm (see section 2.2.4). The posterior $\gamma(t, s)$ corresponds to the probability of being at time t in the tied-state s. Supposing that we have a sequence of tied-states for the best-path $\bar{S} = (\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_N)$, the confidence value for frame t is extracted with its associated state $\bar{s}_t$ as:

$$c_{\bar{s}_t} = \gamma(t, \bar{s}_t) \tag{5.1}$$

The forward-backward algorithm over a lattice is the same as for calculating the denominator posteriors in MMI discriminative training, which was discussed in chapter 3.1.3.

An ideally calibrated frame-confidence should correspond to the probability that the frame-label $\bar{s}_t$ is correct.

What is best? It is very hard to predict which of the three types of confidence will be more useful. In some situations, it might be better to use the less specific sentence-confidences, while with the per-word and per-frame confidences we can locally decide about processing sub-chunks of utterances, which should be good as well. Any guess at this point would be a pure speculation, and we will search for the answer experimentally.

5.2.2 The concept of 'ideal' confidence

By its nature, an optimally calibrated confidence corresponds to the probability that the label is correct. For word-confidence, it is the probability that the recognized word matches its reference. In the case of frame-confidence, it is the probability that the hypothesized tied-state is the same as the element in the forced-alignment.

Only the per-sentence confidence is an exception following a different pattern. Usually we are not interested in the probability that the whole sentence is correct. Instead, we can replace the 'ideal' confidence value by the *word accuracy* in the sentence.

This applies to confidences in general, however, when used in semi-supervised training, it is not clear if these 'ideally calibrated' confidences also lead to the best results. It can be the case that additional processing of confidences is beneficial.

5.2.3 Use of confidences in SGD

Selecting data by confidence

The simplest approach is to select the automatically transcribed training data by setting a threshold on the confidence, or alternatively by setting the target fraction of data to accept.

The more reliable data are accepted (higher confidence), the less reliable data are discarded. For data selection, we don't need calibration of confidences. What matters is the ordering of data according to confidences, the actual values of confidence are not important.

Confidence-weighted training

An alternative approach is to weight the data by the confidence, where the weights c_n are applied to the NN gradients of the individual data-points ∇E_n. The update rule of mini-batch SGD (2.22) is slightly modified to:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \sum_{n=1}^{M} c_n \nabla E_n(\mathbf{w}^{(\tau)}) \ . \tag{5.2}$$

The simple way to implement the gradient scaling is to use c_n to scale the partial derivative of loss function E_n w.r.t. network output $\mathbf{y}$:

$$\frac{\partial E'_n}{\partial \mathbf{y}} = c_n \frac{\partial E_n}{\partial \mathbf{y}} \ . \tag{5.3}$$

5.2.4 What we believe to be interesting

To summarize, there are many interesting ideas in the literature. Here is a list of important topics:

- different methods to obtain per-word confidences
- per-frame confidences
- calibration of confidences
- re-training with transcribed data
- importance sampling
- iterative semi-supervised training
- realigning automatic transcripts
- two-softmax semi-supervised training
- entropy minimization semi-supervised training
- multi-system automatic transcripts and confidences

Clearly, we cannot re-explore all these ideas. Instead, in the spirit of the Occam's razor, we will try to build the simplest possible system that will work well in practice. With this literature survey, we broadened our know-how of semi-supervised training and we also created the context to situate our experiments into.

Chapter 6

What is the best granularity of confidences?

Previously, in section 5.2, we 'layed-out' the area for our semi-supervised training experiments. The key questions were: A) "Should we work with per-sentence, per-word or per-frame confidences?" B) "Should we use data selection or data weighting?" C) "Should we calibrate the confidence?" In this chapter we explore these questions systematically. We begin with 'oracle' experiments, followed by three sections with confidence-types from question A). We also introduce re-training with manually transcribed data, which is necessary to better decide which semi-supervised setup is preferable. Here, we returned to our Babel Vietnamese setup (LimitedLP) from section 4.1

Seed system The seed system is built with full recipe as described in chapter 3. It includes the training of initial GMM-HMM system to produce PLP-pitch-fMLLR features. Then the recipe continues with RBM pre-training, frame cross-entropy training and sMBR training. The seed system is built with 10 hours of transcribed Babel Vietnamese LimitedLP data, the performance of the seed system is 59.6%. The performance of intermediate stages is in table 6.1.

Table 6.1: *Performance of seed system we use in this chapter for semi-supervised experiments. The initial fMLLR-GMM system was used to produce input features, the CE-DNN was trained with frame cross-entropy loss, then it was re-trained with sMBR objective.*

	fMLLR-GMM	CE-DNN	**sMBR-DNN**
WER	66.1	61.7	**59.6**

Semi-supervised training As mentioned earlier, we do frame-CE semi-supervised training with inter-mixed automatically and manually transcribed data. The automatic labels for NN training are the tied-states $\bar{S} = (\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_T)$ from the best-path in lattice $\mathcal{L}$, which was generated with sMBR-DNN seed system. The specific confidences and the way they are used is described later in the chapter.

6.1 Oracle experiments

To better understand the effect of calculating confidences for smaller or larger units, we designed the oracle experiments which use the 'ground truth' transcripts of the untranscribed data. The transcripts are used to *mark the correct words* in the recognized text by using the word-alignment from scoring, or to *mark the correct frames* in the best state-sequence $\bar{S}$ by comparing it with the forced-alignment to transcripts. We create the 'oracle' confidences in the following way:

Per-sentence confidence is related to the estimate of word accuracy in the sentence, rather than the correctness of the whole sentence. The oracle value is the word-accuracy in the automatically transcribed sentence. The DNN self-training is done with the sentences scaled by their word-accuracies, which are post-processed by tuning the exponential scale α (see table 6.2).

Per-word confidence is related to the probability that the word is correct. Here, according to the word-alignment from scoring, the correct words get oracle confidence 1 and incorrect 0, this confidence is then assigned to the frames of the word. The confidences for silence frames on sentence ends or between words are filled with linear interpolation (35% of all frames).

Per-phone confidence would be the probability of labeling a frame with the correct phone. For the oracle confidence, we compare the phones on the best-path from lattice with the phones in forced-alignment. The frames where the phones match get confidence 1, otherwise 0. The self-training is done with these per-frame weights. For the sake of comparison, we simplified the Vietnamese phone-set not to consider the tones or the position in the word.

Tied-state confidence is similar to the per-phone confidence, with the difference that we are comparing the sequence of acoustic model classes (i.e. pdfs, tied-states). Again, we compare the lattice best-path with the forced-alignment. The frames with the same tied-states have confidence 1, otherwise 0. We train with weighting the frames by these confidences, which in fact selects the 'correct' frames from the forced alignment of the untranscribed data-set. This oracle confidence is thus very optimistic.

Table 6.2: *Per-sentence oracle confidences, the sentences are weighted by word accuracy (scaled exponentially by α: $c' = c^{\alpha}$). Although the WER of seed systems with sMBR training was 59.6%, we should also compare to CE-DNN baseline with WER 61.7%. The results in table are from frame-CE training.*

Scale α	0.5	1.0	1.5	2.0	**2.5**	3.0
WER	59.4	59.0	58.9	58.8	**58.7**	59.2

The results

As can be seen in table 6.3, the oracle confidences bring nice improvements. The first two lines in the table show the performance of the baseline CE-DNN system and the sMBR-DNN, which is our seed system. The next two lines compare the systems with no untranscribed data added, or the case when we simply add all the untranscribed data without using any

confidences. The system 'No untranscribed' is better than 'Baseline system', because we replaced the training labels from GMM-alignment with a 'Seed system' alignment. The group of four lines with the proposed *oracle confidences* of different granularity is in the middle. The last line in the table is the best possible oracle, where we trained with the correct transcripts of the 'untranscribed' data.

Table 6.3: *DNN self-training with oracle confidences (optimizing frame Cross-Entropy)*

	WER	WER recovery
Basline system (CE-DNN)	61.7	-
Seed system (with sMBR)	59.6	-
No untranscribed	60.8	0
All untranscribed, no confidence	60.1	8
Oracle confidences:		
Per-sentence confidence	58.7	25
Per-word confidence	57.7	36
Per-phone confidence	56.1	55
Tied-state confidence	55.3	64
With correct transcripts	52.3	100

From the results in table 6.3 we see that:

1. all the oracle confidences are better than using no confidence
2. the results improve as the confidence unit becomes smaller (sentence → word → frame)

This oracle experiment suggests that the tied-state confidences are the most promising. However, the method did not consider the actual confidence values that will be available. In the real world scenario we do not know which frames are correct. Inevitably, the training set will contain some wrongly labeled frames, while we will miss some of the correct frames. The actual results will be certainly worse than in this oracle experiment.

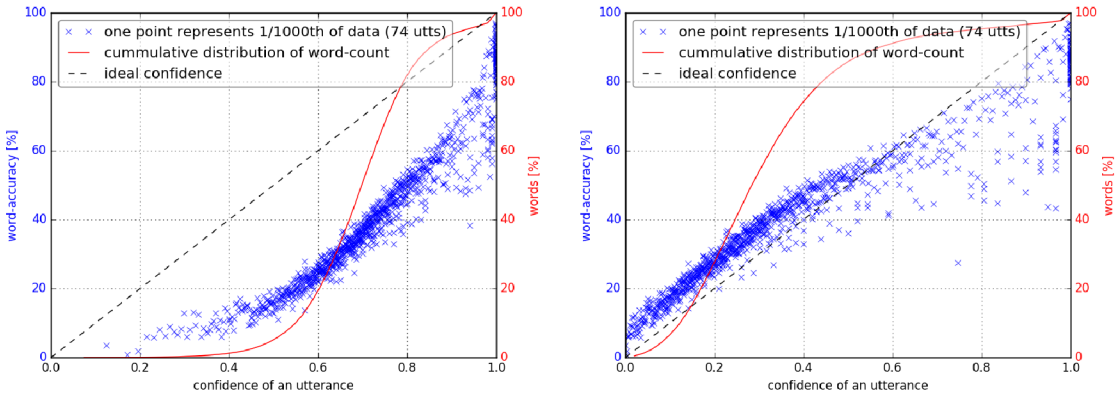
Also, we have to perform the real experiments with the sentence-level and word-level confidences. Only a careful comparison across techniques can verify if the frame confidences are the best.

6.2 Per-sentence confidences

In this section we will experiment with the per-sentence confidences based on the MBR statistics or based on the NN-posteriors. We will address data selection, weighted training and confidence calibration. Finally we show that it is beneficial to further re-train the acoustic model with the correctly transcribed data.

6.2.1 Minimum-Bayes risk confidence

As mentioned earlier in section 5.2.1, the sentence-level confidence c_{sent} is calculated as the average word confidence within a sentence: $c_{sent} = \frac{1}{M} \sum_{i=1}^{M} c_{\bar{w}_q}$. The per-word probabilities



(a) Scale $\alpha = 1.0$ (b) Scale $\alpha = 3.5$

Figure 6.1: *Sentence-confidence from MBR statistics for weighted training.*

$c_{\bar{w}_q}$ are the posteriors $\gamma(q, \bar{w}_q)$ from the Minimum Bayes Risk decoding [Xu et al., 2011, section 7.1], where we fixed the word sequence $\bar{W} = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_T)$ to the best path in lattice $\bar{W} = \text{wrds}(\bar{S})$. The confidences $c_{\bar{w}_q}$ were extracted with the scales that were used to generate the lattices (i.e. the acoustic scale $\kappa = 0.1$, graph scale $\varrho = 1.0$).

Data-selection, no weighting

The simplest experiment, which can be done with the per-sentence confidences, is the data-selection, in which we gradually add sentences ranked by the confidence.

Table 6.4: *Data selection by per-sentence confidence.*

Added sentences	0%	30%	**50%**	**70%**	90%	100%	Oracle
WER	60.9	60.1	**59.8**	**59.8**	60.0	60.1	58.7

From table 6.4, we see that it is good to leave out 30-50% of sentences, which brings a 0.3% WER improvement compared to adding 100% sentences. The oracle WER 58.7 achieved by weighting the sentences with their re-scaled true word accuracy indicates that there is a space for further improvement.

Weighting the sentences by confidence

To make our setup more similar to the oracle experiment, we should weight the sentences in the SGD training. Before doing it, it is good to show how well the confidence matches the word accuracy.

The scatter plot of confidence and accuracy in figure 6.1a revealed that the confidence is more optimistic than the actual word-accuracy, as the blue cloud is far beneath the dashed line representing the ideal match. The steep section of red curve indicates that majority of words comes from sentences with confidence ranging from 0.6 to 0.8.

If the confidence gets closer to the word accuracy, we should also get closer to the oracle performance. For a better match, we can warp the confidence by an exponential scale α as

follows: $c'_{sent} = c^{\alpha}_{sent}$. By a grid search over α in table 6.5, we found the best $\alpha = 3.5$. From the results, we see that the distance to Oracle shrank from 1.1% to 0.6% WER.

Table 6.5: *Weighted training with sentence-confidence from MBR statistics (scaled exponentially by α, keeping all the sentences)*

Scale α	1.0	2.0	2.5	3.0	**3.5**	4.0	Oracle
WER	59.8	59.6	59.6	59.5	**59.3**	59.5	58.7

Using the tuned $\alpha = 3.5$, we regenerated the confidence-accuracy scatter plot. In figure 6.1b, we clearly see that large part of mismatch is removed, as the blue cloud nearly overlaps with the dashed line.

We also combined the weighted-training with data-selection, the grid search of scale α was repeated while we added the top 90% 70% and 50% of the automatically transcribed sentences. In table 6.6, we found a further drop by 0.1% WER, in case of adding 90% with confidence scale $\alpha = 4.0$.

Table 6.6: *Weighted training with sentence-confidence from MBR statistics (scaled exponentially by α, adding portion of top N% of untranscribed sentences)*

Scale α	1.0	2.0	2.5	3.0	3.5	**4.0**	4.5	Added sentences
	59.8	59.6	59.6	59.4	59.3	**59.2**	59.3	**90%**
WER	59.6	59.5	59.5	59.4	59.5	**59.2**	59.3	**70%**
	59.6	59.6	59.5	59.6	59.5	59.5	59.5	50%

6.2.2 Calibration of confidences

A common approach to calibrate confidences is to train a logistic regression on the annotated development data, which are disjoint both from the training set of ASR seed-model and the untranscribed data [Yu et al., 2011]. In this section we verify if such calibration of the per-word MBR statistics can improve the efficiency of semi-supervised training with weighted sentences.

Table 6.7: *Weighted training with per-sentence confidence from the calibrated word-confidences (derived from 'raw' MBR statistics, scaled exponentially by α, keeping all the sentences)*

Scale α	1.0	1.5	2.0	2.5	**3.0**	3.5	4.0	4.5	Oracle
WER	59.5	59.5	59.4	59.3	**59.3**	59.3	59.4	59.6	58.7

As we achieved the same best performance in tables 6.5 and 6.7, we can conclude that the *calibration did not lead to a performance improvement* in the semi-supervised training with the weighted sentences.

6.2.3 Summary

In this section, we compared three types of per-sentence confidence based on 1) MBR-decoding statistics, 2) calibrated MBR decoding-statistics, 3) neural network posteriors.

First, we found that weighted training leads to better results than simple data selection (WER% 59.8 → 59.3). However, for weighted training, we need a 'probabilistic' confidence, which we further tune by an exponential scale α, and this tuning is time consuming.

Table 6.8: *Weighted SGD training with various sentence confidences.*

Confidence method	WER
MBR statistics	59.3
Calibrated MBR statistics	59.3
NN-posteriors	59.4
Oracle	58.7

The best results for the three different confidence methods are summarized in table 6.8, where we see that the calibration of MBR statistics by logistic regression did not improve the system and the confidence from NN-posteriors [Zhang et al., 2014a] was a little worse than the MBR based confidences.

6.2.4 Re-training with transcribed data

In literature, we can find that it is beneficial to post-process the model trained with the inter-mixed transcribed and untranscribed data. In [Thomas et al., 2013], the DNN output layer was discarded and trained again from random initialization, by using only the correctly transcribed data.

We also tried an alternative scenario in which we keep the output layer 'as-is' and continue training with the 10 hours of correctly transcribed data, while using a smaller learning rate (initial learning rate 0.001 instead of the original value 0.008). The model post-processing in this spirit was described in [Grézl and Karafiát, 2014] as 'fine-tuning'.

To verify that the per-sentence confidences are helpful, we post-process these two 'initial models':

1. no confidence: the initial model was built by adding all the automatically labeled sentences, no confidences are used

2. best per-sentence confidence: the initial model is built by training with weighted sentences ($\alpha = 4.0$), while 90% sentences are added (the best result from table 6.6).

The difference of these two scenarios is the improvement from using the per-sentence confidences. The results in table 6.9 show the post-processing by both the 'frame CE' training and the subsequent 'sMBR' training. Both is done with the 10 hours of correctly transcribed data.

If we focus on 'frame CE' numbers, we see that re-training with the correctly transcribed data improves the results in all the cases, while we were picking the best initial learning rate from 0.008, 0.001 and 0.0001.

Then, if we (A) discard the output layer, there is no difference whether the initial model was built with confidences or not, see the 'frame CE' line.

If we (B) re-train the initial model, it is better to start from the model trained with the best per-sentence confidences, see the 'frame CE' line.

This improvement in (B) persists also after the sMBR training (last row in table 6.9, 3rd column). Based on this sMBR result, we can conclude that it is good to introduce the

Table 6.9: *Re-training the 'initial model' with 10 hours of correctly transcribed data.*

WER	no confidence (added 100% sentences)	best per-sentence confidence (added 90% sentences, $\alpha = 4.0$)	
initial model (semi-supervised)	**60.1**	**59.2**	frame CE
	58.3	57.6	sMBR
(A) discard last layer	58.7 (lrate 0.008)	58.7 (lrate 0.008)	frame CE
	57.5	57.6	sMBR
(B) re-train the initial model	58.7 (lrate 0.008)	**58.3 (lrate 0.001)**	frame CE
	57.6	**57.2**	sMBR

'frame CE' re-training before the sMBR training. By skipping it, the performance would degrade by 0.4%, from 57.2 to 57.6.

Also, we can conclude that the use of per-sentence confidences is beneficial. The best result without confidences is 57.5, while with the confidences we obtained 57.2.

6.3 Per-word confidences

In this section, we will use the Minimum Bayes Risk statistics $c_{\bar{w}_q} = \gamma(q, \bar{w}_q)$ directly as the word-confidences. Primarily, we are interested to decide if the confidences are more efficient when used per-sentence or per-word. The literature presents both approaches (per-word [Wessel and Ney, 2005], per-sentence [Novotney et al., 2009]), but we have not seen a direct comparison. Per-sentence confidences are used more frequently, probably because of their simpler use. The oracle experiment in chapter 6.1 was favoring the per-word confidences over the per-sentence confidences.

The experiments will be conducted following a similar pattern as in the previous section 6.2. The behavior can be different if the processing decisions are done on the word-level basis.

6.3.1 Minimum Bayes Risk confidence

We start from the non-calibrated MBR statistics that were introduced on page 27; we'll use them for word-selection or word-weighting. Previously, with the per-sentence confidences, 'weighting' was better than 'selecting'.

Word-selection, no weighting

In this experiment, we are adding words into the *frame CE* training, starting from the highest confidence. The frames at the selected words have training weight 1, while the frames of rejected words have weight 0. The weights in eventual silence gaps between words are filled by linear interpolation.

In table 6.10, we tune the fraction of added words. We directly got to WER 59.1, which is a little better than the best performance achieved with per-sentence confidences (see table 6.6). This indicates that the per-word confidences are at least as good as the per-sentence ones.

Table 6.10: *Data selection by per-word confidence, Babel Vietnamese.*

	Added words [%]								Word	Seed
	0	20	30	**40**	50	60	70	100	oracle	system
WER	60.9	59.5	59.2	**59.1**	59.2	59.3	59.6	60.1	57.7	59.6

Previously, in table 6.6, we used training with weighted sentences, while table 6.10 was prepared with simple hard-selection of words. It is also remarkable that the optimal amount of added words seem to coincide with the word-accuracy of the seed system, which is $(100 - 59.6 = 40.4)$. We will re-visit this later in chapter 7 with a different experimental setup.

Weighting words by confidence

Next, we replace the data selection with weighted SGD training. The confidence of a word is used to scale the gradients over its time-span. Again, the silences are bridged by linear interpolation of the confidences from the adjacent words. We extend the *frame CE* training set with all the untranscribed data, while we tune the exponential scale α applied to word-confidences: $c'_{\bar{w}_q} = (c_{\bar{w}_q})^{\alpha}$. The 'raw' word confidences were generated with the lattice-scale $\lambda = 1.0$ (i.e. with default scaling from lattice generation by $\kappa = 0.1$, $\varrho = 1.0$).

Table 6.11: *Weighted training with MBR statistics (uncalibrated per-word confidence, scaled exponentially by α, all words were added)*

Scale α	1.0	2.0	3.0	4.0	5.0	6.0	7.0	Oracle
WER	59.5	59.2	59.2	59.1	59.0	59.0	59.1	57.7

Scale α	8.0	9.0	10.0	11.0	**12.0**	13.0	14.0	Oracle
WER	58.9	59.0	59.0	58.9	**58.8**	58.9	59.0	57.7

In table 6.11, we see that the best alpha 12.0 leads to WER 58.8%, which is by 0.3% better than we had with the word-selection in table 6.10. The WERs seem to fluctuate between 59.1 and 58.8. The best exponent value 12.0 is relatively high, we can see it as a soft version of data selection. For example, the words with confidence 0.68 get a training weight $c_{\bar{w}_q} = 0.68^{12.0} \doteq 0.01$, which in our case almost removed 43% words from the training.

6.3.2 Weighting words by calibrated confidence

In section 6.2.2, we calibrated the MBR statistics by logistic regression. We re-used the same calibrated per-word confidences for experiment in table 6.12. The best alphas 2.0, 3.0 lead to WER 58.8. Here also, the calibration of confidence did not bring a performance improvement (same best result in tables 6.11 and 6.12).

Table 6.12: *Weighted training with calibrated word-confidence, (calibration by logistic regression, exponential scaling by α, keeping all the words)*

Scale α	0.5	1.0	1.6	**2.0**	2.5	**3.0**	4.0	5.0	6.0	Oracle
WER	59.6	59.4	59.1	**58.8**	59.0	**58.8**	58.9	59.1	59.3	57.7

6.3.3 What happens in data selection?

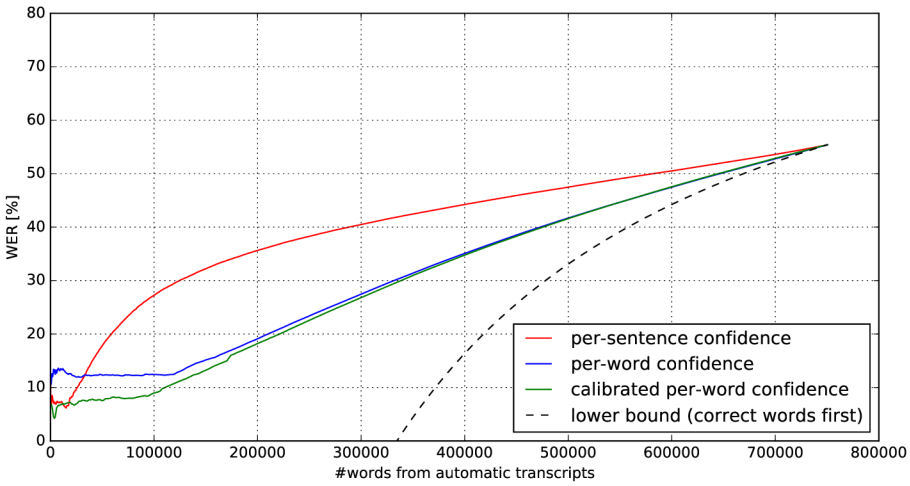


Figure 6.2: *WER in subset of data selected by the confidence, starting from the highest. The per-word confidences (blue curve) allow us to select data with lower WER, than the per-sentence confidences (red curve). The calibration by logistic regression further improves the data selection (green curve). However, the better calibration did not translate into improvement of the semi-supervised training. All the confidences are based on Minimum Bayes Risk decoding statistics. The WER on this plot is without deletions and we normalized by the length of the hypothesis, as we cannot select a 'deleted' word from the automatic transcripts.*

To demonstrate that the data selection done per-word is better than per-sentence, we created figure 6.2, where we compare the WER as function of number of selected words. We see that by selecting the individual words, we can create a subset with lower WER (blue curve) than if we select whole sentences (red curve). With the 'un-weighted' data selection, the WER was the following: 59.8 for per-sentence confidences (table 6.4), 59.1 for per-word confidence (table 6.10). With the weighted training, the difference shrank from 0.7 to 0.4, as we had 59.2 with per-sentence confidences and 58.8 with per-word confidences.

6.3.4 Re-training with transcribed data

Similarly to the previous section, we take the best *'initial model'* and post-process it by re-training with the 10 hour set of the correctly transcribed data. As we found earlier, we first re-train by 'frame CE' training and continue with 'sMBR' training. As a sanity check we also compare with the sMBR done directly from the initial model.

The results in table 6.13 show that the final sMBR WER 56.9 is by 0.3% better than we previously obtained with the per-sentence confidences. This makes the per-word confidences better than the per-sentence confidences. We believe that this explicit comparison has not been shown in any study yet.

6.4 Tied-state confidences

In this section, we return to the frame confidences $c_{\bar{s}_t} = \gamma(t, \bar{s}_t)$, The lattice posteriors for states from best path can be used as frame confidence in weighted mini-batch SGD training.

Table 6.13: *Re-training the best 'initial models' with 10 hours of correctly transcribed data. The 'initial models' are obtained by three variants of semi-supervised training: a) no confidences at all, b) best per-sentence confidences and c) best setup with per-word confidences (weighted training with uncalibrated per-word confidences, exponential scale $\alpha = 12.0$, added all the words.*

WER	a) no confidence	b) best sentence confidence	c) best word confidence	
initial model	**60.1**	**59.2**	**58.8**	frame CE
(semi-supervised)	58.3	57.6	57.4	sMBR
re-train the initial CE model	58.7 (lrate 0.008)	58.3 (lrate 0.001)	**58.2** (lrate 0.001)	frame CE
	57.6	57.2	**56.9**	sMBR

We observed that it is beneficial to tune the 'lattice-scale' λ by rescaling both the acoustic and graph scores in the lattice, which shifts the confidences closer to the ideal confidence (probability that the label is correct). In this section, we will extend this approach in two directions:

- we further introduce the exponential scale α that is applied to the extracted frame confidences (i.e. the frame posterior from lattice at states on the best path). The ideal confidence may not be the best weight for the SGD training, so we 'bend' the distribution of the confidences by exponential scaling $c'_{\bar{s}_t} = (c_{\bar{s}_t})^{\alpha}$.
- we introduce the per-pdf calibration, assuming that the confidences behave differently across tied-states. The calibration helps to select data-points with less annotation errors than if we used the global calibration.

In this section we want to see if the tied-state confidences lead to better results than we obtained with the per-word confidences.

6.4.1 Tuning confidence scale α in frame-weighted SGD training

Table 6.14: *Weighted training with per-frame confidences (the tied-state posteriors from lattice for the state on the best path). The confidences were extracted with the lattice scales $\lambda = \{1.0, 0.3, 0.02\}$ (applied both to acoustic and graph scores in the lattice). After the extraction, the frame confidences were re-scaled by exponential scale α.*

Scale α	1.0	2.0	3.0	4.0	5.0	6.0	Lattice scale λ
	59.4	59.1	59.0	59.1	**58.9**	59.0	1.0
WER	59.2	**59.0**	**59.0**	59.2	59.3	59.3	0.3
	59.1	59.1	59.2	59.4	59.4	59.5	0.02

In table 6.14, we see that for any lattice scale, we can always tune such alpha for which the WER becomes similar regardless of the initial lattice scale. We decided to fix the lattice-scale to $\lambda = 0.3$ for further experiments, as it reasonably pre-calibrates the confidences and the scale is not too far from $\lambda = 1.0$.

6.4.2 Per-phoneme analysis of the frame confidences

So far, we have been analyzing the confidences for the whole untranscribed data-set. To get an idea how the confidences differ for individual phonemes, we show them separately in figure 6.3 (frame-accuracy as function of confidence).

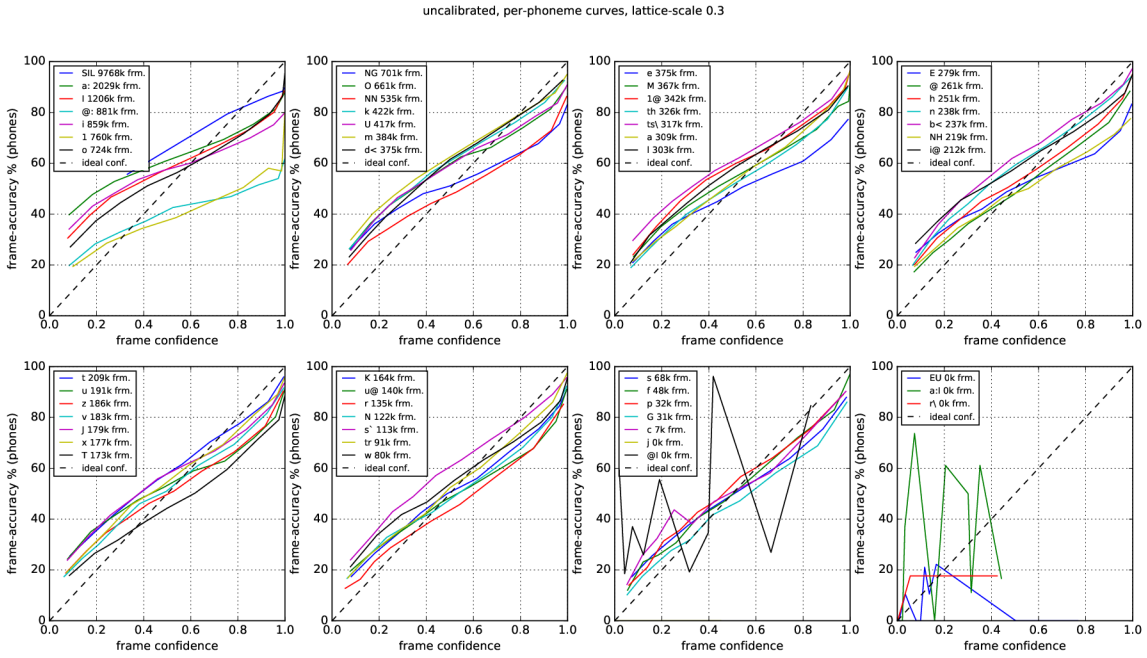


Figure 6.3: *Per-phoneme frame-accuracy as function of confidence (uncalibrated), from lattice-posteriors with scaling* $\lambda = 0.3$. *Here, a frame is considered correct when the phoneme label matches both in the alignment and the best-path from lattice. The curves are sorted by frame-count of phonemes, which are marked in the legend.*

In the graphs, we see that the curves of phonemes with similar frame counts look similar. The curves of highly represented phonemes are smooth, while the curves for low-represented phonemes are noisy. Then we see that the curves of the 5 most represented phonemes (vowels *a, e, i, o* and *n*) begin with higher frame-accuracies than the other phonemes. We also see that silence, which is the most represented label, has both high confidence and high accuracy. This is understandable, it is harder to confuse the silence with a phone than to confuse two phones.

It would be good to use the plots for suggestions to the experiments, however it is not clear how exactly. We can confirm that there definitely are differences among the phoneme curves, these will be reduced by our calibration.

6.4.3 Calibration by logistic regression

Because we saw in figure 6.3 that there are differences among the accuracy curves of the individual phonemes, we can apply a pool of calibration models to reduce the differences. Each model is a simple logistic regression with 2-parameters (scale, bias) for each class of acoustic model (pdf, tied-state), the set of models is trained on the transcribed development set. Yes, we do not build a per-phoneme model, but a per-tied-state one (i.e. we have a model per neural network class). Note that the oracle results in table 6.3 were the best for

the per-pdf confidences.

The pdf's are represented by various amounts of data. For some pdf's, there is too little data for reliable estimation of calibration model parameters. To solve this, we adopted a three-level hierarchy of models: a) per-pdf models, b) per-phoneme models and c) a global model. In case, there is not enough data for a given unit, we switch to higher-level model a) $\rightarrow$ b) $\rightarrow$ c). By observing their loss function values, we found that for a minimum of 100 data-points, the logistic regressions are already trained reasonably. The per-pdf models 'a)' were used for 91.5% frames, the per-phoneme models 'b)' for 8% frames, the the global model 'c)' for 0.5% frames. We started from the lattice posterior frame confidences with the lattice-scale $\lambda = 0.3$.

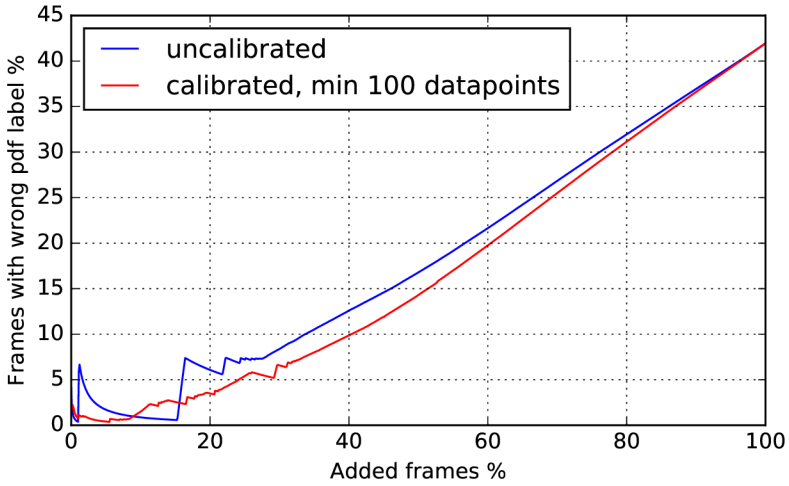


Figure 6.4: *Frame selection according to frame-confidence with or without the calibration (hierarchical three-level calibration). The minimum number of data-points for training a calibration model (logistic regression) was 100. We see that the calibrated confidences allow us to select frames with less errors (red curve) than without the calibration (blue curve). The benefit comes from training the calibration models per-pdf (tied-state), which changes the overall ranking of the frames.*

In figure 6.4, we see how our three-level calibration changes the gradual selection of frames, starting from the highest confidence. The gap between the curves is promising as it means that with the calibration, we can select data-points containing less wrong labels.

Frame selection experiments

In the first set of experiments, we perform the 'data-selection', starting from the highest confidence: the weights are either 0 (dropped frame) or 1 (selected frame).

Table 6.15: *Data selection by per-frame confidence, WER.*

Added frames:	0%	20%	*40%*	50%	60%	70%	80%	100%
Uncalibrated conf.	60.9	61.0	*60.1*	59.4	**59.1**	59.3	59.2	60.1
three-level calibration	60.9	60.3	*59.4*	59.1	58.9	**58.8**	59.0	60.1

In table 6.15, we see that the calibration helped a lot when selecting 40% of data, where

the gap between curves in figure 6.4 was large. The best results are obtained when selecting 60% or 70% frames, where the confidence calibration brought a WER improvement of 0.3%.

Frame weighted experiments

In the next experiment, we use the calibrated confidences for frame-weighting:

Table 6.16: *Weighted training with calibrated per-frame confidences. The calibration is a three-level hierarchy of calibration models. The calibrated confidences are scaled exponentially by α.*

Scale α	0.5	1.0	1.5	**2.0**	**2.5**	3.0	4.0	5.0	6.0	Oracle
WER	59.6	59.1	59.0	**58.6**	**58.6**	59.0	59.0	59.2	59.4	55.3

In table 6.16 we obtained the best result so far for semi-supervised training with mixed data (transcribed and untranscribed).

The hierarchical three-level calibration improved the results by 0.3% WER. Previously, the calibration of the per-word confidences, was not bringing improvements (see sections 6.2.2 and 6.3.2).

The performance obtained with the frame-weighted training is by 0.2% WER better than the frame-selection, both for the calibrated and uncalibrated confidences (see the tables 6.14 and 6.16).

6.4.4 Re-training with transcribed data

As we did in the previous sections, we further re-train the *'initial model'* that was trained with a mix of transcribed and untranscribed data. The re-training is done with the 10-hour set of correctly transcribed data. We first re-train by 'frame CE' training to continue with 'sMBR' training. As a sanity check we also compare with sMBR done directly from the initial model.

Table 6.17: *Re-training the best 'initial model' with 10 hours of correctly transcribed data, after 'frame CE' training with the mixed data (transcribed and untranscribed). We compare four scenarios of semi-supervised training: a) no confidences at all, b) best per-sentence confidences, c) best setup with per-word confidences and d) best setup with per-frame confidences (frame confidences extracted with lattice-scale $\lambda = 0.3$, re-calibrated by three-levels of logistic regressions and post-processed by exponential scale $\alpha = 2.0$, weighted SGD training). The 'frame-CE' re-training is followed by 'sMBR'*

WER	a) no confidence	b) best sentence confidence	c) best word confidence	d) best frame confidence	
initial model	**60.1**	**59.2**	**58.8**	**58.6**	frame CE
(semi-supervised)	58.3	57.6	57.4	57.1	sMBR
re-train the initial CE model with true transcripts	58.7 (lrate 0.008)	58.3 (lrate 0.001)	58.2 (lrate 0.001)	**58.0** (lrate 0.001)	frame CE
	57.6	57.2	**56.9**	57.0	sMBR

The re-training result in table 6.17 for 'frame CE' objective shows that the calibrated per-frame confidences are the best, with WER of 58.0. From the sMBR results, we see that

the best model remained the word-confidence DNN with WER 56.9. The sMBR results from the 'best frame confidence' column reveal that, now, there is almost no difference between the sMBR training from the 'initial model' 57.1 and the re-trained model 57.0, which we did not see in other columns.

6.5 Summary

The observations from this chapter, which was focused on granularity of confidences, can be summarized as follows:

- We found it important to **re-tune the model with a small amount of correctly transcribed data**. The re-tuning is done with smaller learning rate 0.001 (table 6.9, otherwise the initial learning rate is 0.008).

- Also, it is beneficial to **go beyond simple sentence selection**, which can often be seen in the literature (see table 6.18). The very simple, but still powerful approach is to select frames corresponding to the N% words with the best confidence.

- If we compare the results in 'data-selection' table 6.18 with the results in the first column from the 'data-weighting' table 6.19, we see that **'data-weighting' leads to better results than 'data-selection'**. Along the way, we also noticed that data **'weighting' and 'selection' are not complementary**. However, the **data-selection** is more **straightforward to tune-up**, as we can use 'raw' confidences in it.

- For getting better results, we had to **introduce a hyper-parameter that is expensive to tune**: the N% amount of added data for data selection, or the exponential scale α for approximate calibration of 'raw' confidences in role of frame weights.

- For 'proper' confidence calibration with logistic regression, we introduced dependency on the correctly transcribed development set. In some cases, there was no improvement from the confidence calibration (per-sentence, per-word confidences). With the per-frame confidences, we obtained 0.3% WER improvement, this was however absorbed by the re-tuning with frame-CE training and then by sMBR training. Based on this evidence, we can form a conclusion that the **confidence calibration is not necessary for semi-supervised training**.

- Despite the initial optimism from the oracle results (table 6.3), where the tied-state confidence oracle dropped deeply to 55.3% (but this oracle completely ignored the confidence values, as it just selected the frames with correct labels), the final gains from semi-supervised training were more modest, leading to 58.0% WER after re-training with the correctly transcribed data. The final best WER-recovery was 33%.

- In the end, there were **small differences between using the per-sentence, per-word or per-frame confidences**, especially after re-tuning with the 'frame CE' and 'sMBR' objectives using the correctly transcribed data. Still, it is **clearly beneficial to use the confidences in the semi-supervised training** (see the last line in table 6.19).

In our case, the results for word-confidences and frame-confidences are very similar. Hence, to decide which approach is the best, we will use the Occam's razor. The setup with

Table 6.18: *Data selection, summary. Each time we added the optimal amount of data, the amount of data is the only hyper-parameter we needed to tune manually. The two results for the 'frame-selection' refer to 'uncalibrated/calibrated' confidences.*

	WER	taken from
Sentence selection	59.8	table 6.4
Word selection	59.1	table 6.10
Frame selection	59.1/58.8	table 6.15

Table 6.19: *Data weighting, summary. Each time we re-scaled the confidences by tuning the exponential scale α. The frame confidences were calibrated, the sentence and word confidences were better without calibration. The results are from table 6.17.*

	WER	(re-tuned)	(re-tuned + sMBR)
Sentence weighting	59.2	58.3	57.2
Word weighting	58.8	58.2	**56.9**
Frame weighting	**58.6**	**58.0**	57.0
no confidence	60.1	58.7	57.6

per-word confidences is simpler as it requires less storage for confidence values. The system without 'proper' calibration of confidences does not depend on development set. The only hyper-parameter we need to tune is the exponential scale α for weighted training or the N% of added words for data-selection.

Chapter 7

Finding generic semi-supervised training approach

Ideally, we are interested in finding such semi-supervised training recipe, that will be efficient for a broad range of scenarios. Until now, we explored the behavior for Babel languages (mainly Vietnamese) and one scenario (10 hours are transcribed, ≈70 hours are untranscribed).

In section 6.3.1, we saw that the best percentage of added words seems to correspond to the word accuracy of the seed system (in table 6.10 it was good to add 40% words, while the WER of the seed system was 59.6%, i.e. the word accuracy was 40.4%).

A closer look on word-selection is in figure 7.1. We split the automatically transcribed Vietnamese words into 10 bins with same amount of words, while the words with similar confidence are in the same bin. We see that in the first 4 bins there are more correct words than wrong ones, while by adding the fifth and next bins, we would introduce more incorrectly labeled words than correct ones. In the selection of 40% words, there is 27% WER as can be read from blue curve in figure 6.2 on page 38.

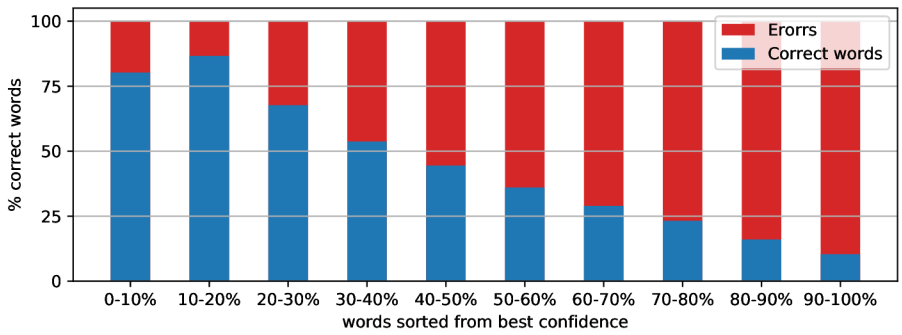


Figure 7.1: *Automatic transcripts of Vietnamese sorted from best word-confidence, split into 10 bins with same word-count. See the correspondence of the bars with results in table 6.10 on page 37 and blue curve in figure 6.2 on page 38.*

Now, let's check if our *word selection rule* based on word accuracy W_{acc} of seed system generalizes to other databases. From results of Babel Bengali in table 7.1 and Switchboard in table 7.2 we see that the same rule holds.

For example, for Switchboard in table 7.2, the W_{acc} of the seed system was 73.1%, while the optimal amount of added words was 70%, so the word accuracy of the seed system was

Table 7.1: *Data selection by per-word confidence, babel Bengali.*

Added words	0%	20%	30%	**40%**	**50%**	60%	70%	100%	**Seed $\mathbf{W_{acc}}$**
WER	64.2	62.9	62.5	**62.3**	**62.3**	62.4	62.5	63.2	**37.1**

Table 7.2: *Data selection by per-word confidence. Our Switchboard setup has 14 hours transcribed, 95 hours are untranscribed. The LM is trained on Fisher transcripts. The results are for HUB5-2000 (Switchboard + CallHome), further description of the setup is in section 4.2.*

Added words	0%	50%	60%	**70%**	80%	90%	100%	**Seed $\mathbf{W_{acc}}$**
hub5 WER	28.0	25.3	25.1	**24.4**	24.7	24.5	24.8	**73.1**

the right amount of words to add. Such simple rule works reasonably well for such different setups as the Babel Vietnamese, Babel Bengali or even for Switchboard data.

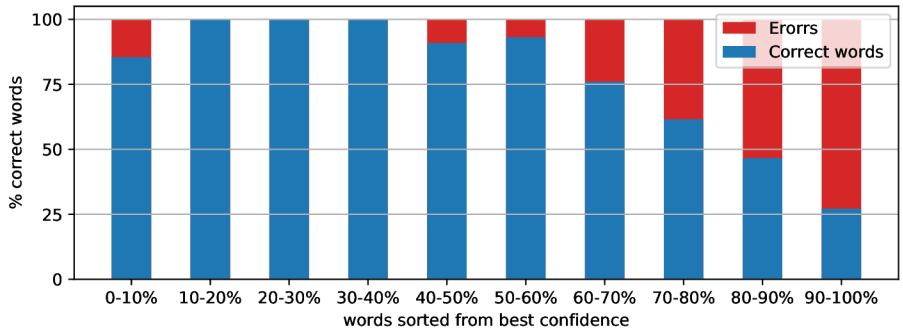


Figure 7.2: *Automatic transcripts of Switchboard sorted from best word-confidence, split into 10 bins with same word-count. See the correspondence of the bars with results in table 7.2.*

The bar-graph showing word-selection for Switchboard is in figure 7.2. We see that there is certain amount of incorrect words with high confidence, while the biggest portion of wrong words is located in the last three bins. The selection of top 70% words has a WER of 7.4%.

The only external information we needed for our *word selection rule* is the WER of the seed system calculated on some development set, while we assume that the untranscribed data and development data are similar. We are aware that it is not rigorously 'guaranteed' that it will always lead to best possible results, in the same time, it will often be a good fit for its simplicity.

7.1 Re-tuning with correctly transcribed data, Switchboard

As, for Switchboard, we found word-selection to be as good as word-weighting, we take these two systems and proceed with re-tuning. For the final comparison, we re-tune the 'initial models' with the small 14hour set of the correctly transcribed data. We re-tune first by training with 'frame CE' objective and then with 'sMBR'. For comparison, we also run the sMBR training directly from the initial model.

Table 7.3: *Switchboard, re-training the 'initial models' with 14 hours of correctly transcribed data. The 'initial models' are obtained by three variants of semi-supervised training: a) no confidences at all, b) best training with weighted words ($\alpha = 7.0, \lambda = 1.0$) and c) best word selection (selected top 70% of words)*

WER, seed 26.9	a) no confidence	b) best word weighting	c) best word selection	
initial model (semi-supervised)	**24.8**	**24.4**	**24.4**	frame CE
	24.0	**23.6**	23.9	sMBR
re-train the initial CE model	24.3 (lrate 0.001)	**24.1** (lrate 0.001)	24.2 (lrate 0.001)	frame CE
	23.7	**23.5**	23.7	sMBR

For Switchboard (table 7.3), the final sMBR result for word-selection c) was 23.7, which is by 0.2% WER worse than with the word-weighting b). And the sMBR result of c) is the same as if no confidences were used in a). We see that the word-selection did not bring an improvement, while it also was not harmful. The WER recovery of the system c) is 63%, which is much higher than we had for Vietnamese (33%), which can be explained by lower WER in automatic transcripts from Switchboard seed system.

Table 7.4: *Babel Vietnamese, re-training the 'initial models' with 10 hours of correctly transcribed data. The 'initial models' are obtained by three variants of semi-supervised training: a) no confidences at all, b) best per-word weighted training ($\alpha = 12.0, \lambda = 1.0$) and c) best word selection (selected 40% words)*

WER, seed 59.6	a) no confidence	b) best word weighting	c) best word selection	
initial model (semi-supervised)	**60.1**	**58.8**	**59.1**	frame CE
	58.3	**57.4**	57.6	sMBR
re-train the initial CE model	58.7 (lrate 0.008)	**58.2** (lrate 0.001)	58.4 (lrate 0.001)	frame CE
	57.6	**56.9**	57.1	sMBR

The same set of experiments done for Babel Vietnamese is in table 7.4; here the degradation between word-selection c) and the word-weighting b) is $57.1 - 56.9 = 0.2$, while the word-selection c) is better than the system without confidences a) by 0.5% WER.

If the 'simple word-selection' causes either an improvement or no harm, it is still a preferable technique. It it is much faster than the careful tuning of the exponential scale α by a grid search of NN trainings, while such careful tuning brought only a small 0.2% WER improvement.

7.2 Final summary, simple word-selection

The final summary of the WER improvements we obtained with our preferred *simple word-selection* technique is in table 7.5. We see that the overall absolute WER improvement between the seed system and the final sMBR systems is 2.5% for Babel Vietnamese, 2.3%

for Babel Bengali and 3.2% for Switchboard. For Bengali and Vietnamese the WER in the automatic transcripts was higher, so the absolute WER improvement from the semi-supervised training is smaller than in the case of Switchboard. At the same time, the use of confidences was more important for Vietnamese (with higher WER), than for Switchboard (with smaller WER).

Table 7.5: *Final WER performance of the semi-supervised training based on 'simple word-selection'. The initial model is trained with the mixed data (transcribed and untranscribed), the re-training is done with the smaller set of correctly transcribed data.*

WER	Vietnamese	Bengali	Switchboard
seed system (sMBR)	59.6	62.9	26.9
initial-model (mixed data)	59.1	62.3	24.4
+ re-trained (frame CE)	58.4	61.6	24.2
+ re-trained (sMBR)	57.1	60.6	23.7
abs. WER improvement	2.5	2.3	3.2

Chapter 8

Final remarks

Initial chapters In this thesis we initially presented a quick introduction to the theory of speech recognition and neural network training, along with our NN-training implementation available as the 'nnet1' training recipe in Kaldi. The recipe is composed of RBM pre-training, mini-batch frame Cross-Entropy training, and sequence-discriminative sMBR training.

Semi-supervised training, what we searched for? Then we switched to the main topic, which is the semi-supervised training of DNN-based ASR systems. Inspired by the literature survey and our initial experiments, we investigated several questions: Firstly, whether the confidences are better to be calculated per-sentence, per-word or per-frame. Then, if the confidences should be used for the data-selection or the data-weighting, which is both compatible with the weighted mini-batch SGD training. It was also not clear whether the confidence calibration can improve the performance of the semi-supervised training. We also investigated how the model should be re-tuned with the correctly transcribed data. And finally we searched for a simple recipe, avoids a grid search over hyper-parameter and that is practical for general use with any dataset.

What we found out The performance differences of the systems with various confidences were relatively small, while it was easier to obtain good results with word-confidences and frame-confidences. The data-weighting (with a tuned scale α as exponent) led to a little better results than the data-selection. The confidence calibration led to minimal or no performance improvements. And the re-training with correctly transcribed data is better to be done first with the 'frame Cross Entropy' objective and then with the 'sMBR' objective.

Finally a **practical recipe** without a computationally expensive hyper-parameter tuning is following: *Use the best-path from lattice as NN training targets. From the best-path, select the words whose confidence is in top N%, where the N% is given by word-accuracy of seed system in the development set. The word confidences are extracted as the 'posteriors' from the MBR decoding, in which the word-sequence is fixed to the words obtained from best-path in lattice.*

Final conclusion We found it quite difficult to further improve the performance of the semi-supervised training. Still, we believe, that our findings will be perceived to have practical value. The untranscribed data are abundant and easy to obtain, while our proposed solution brings solid WER improvements (see table 7.5 on page 48) and is not difficult to replicate.

The main results from this thesis were recently published in [Veselý et al., 2017].

Bibliography

[Bahl et al., 1986] Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE ICASSP*, volume 1, pages 49–52.

[Barron, 1993] Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Montréal, U. D., and Québec, M. (2007). Greedy layer-wise training of deep networks. In *In NIPS*. MIT Press.

[Bishop, 2007] Bishop, C. M. (2007). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 1st ed. 2006. corr. 2nd printing edition.

[Bourlard and Morgan, 1993] Bourlard, H. A. and Morgan, N. (1993). *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA, USA.

[Bridle and Dodd, 1991] Bridle, J. S. and Dodd, L. (1991). An Alphanet approach to optimising input transformations for continuous speech recognition. In *Proc. IEEE ICASSP*, volume 1, pages 277–280.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *MCSS*, 2(4):303–314.

[Ghahremani et al., 2014] Ghahremani, P., BabaAli, B., Povey, D., Riedhammer, K., Trmal, J., and Khudanpur, S. (2014). A pitch extraction algorithm tuned for automatic speech recognition. In *Proceedings of ICASSP*.

[Gibson and Hain, 2006] Gibson, M. and Hain, T. (2006). Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition. In *Proc. INTERSPEECH*, pages 2406–2409.

[Grézl and Karafiát, 2014] Grézl, F. and Karafiát, M. (2014). Combination of multilingual and semi-supervised training for under-resourced languages. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*.

[Hinton, 2012] Hinton, G. E. (2012). A Practical Guide to Training Restricted Boltzmann Machines. In *Neural Networks: Tricks of the Trade - Second Edition*.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

[Jaitly et al., 2012] Jaitly, N., Nguyen, P., Senior, A., and Vanhoucke, V. (2012). Application of pretrained deep neural networks to large vocabulary speech recognition. In *Proc. INTERSPEECH*.

[Kaiser et al., 2000] Kaiser, J., Horvat, B., and Kačič, Z. (2000). A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. ICSLP*, volume 2, pages 887–890.

[Kingsbury, 2009] Kingsbury, B. (2009). Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In *Proc. IEEE ICASSP*, pages 3761–3764.

[Kingsbury et al., 2012] Kingsbury, B., Sainath, T. N., and Soltau, H. (2012). Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization. In *Proc. INTERSPEECH*.

[Krogh and Riis, 1999] Krogh, A. and Riis, S. K. (1999). Hidden neural networks. *Neural Computation*, 11(2):541–563.

[Ng et al., 2012] Ng, T., Zhang, B., Nguyen, L., Matsoukas, S., Zhou, X., Mesgarani, N., Veselý, K., and Matějka, P. (2012). Developing a Speech Activity Detection System for the DARPA RATS Program. In *Proc. of INTERSPEECH 2012*.

[Novotney and Schwartz, 2009] Novotney, S. and Schwartz, R. M. (2009). Analysis of low-resource acoustic model self-training. In *Proc. of INTERSPEECH*, pages 244–247.

[Novotney et al., 2009] Novotney, S., Schwartz, R. M., and Ma, J. Z. (2009). Unsupervised acoustic and language model training with small amounts of labelled data. In *Proc. of IEEE ICASSP*, pages 4297–4300.

[Povey, 2003] Povey, D. (2003). *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, Cambridge, UK.

[Povey et al., 2012] Povey, D., Hannemann, M., Boulianne, G., Burget, L., Ghoshal, A., Janda, M., Karafiát, M., Kombrink, S., Motlícek, P., Qian, Y., Riedhammer, K., Veselý, K., and Vu, N. T. (2012). Generating exact lattices in the WFST framework. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*.

[Povey et al., 2008] Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., and Visweswariah, K. (2008). Boosted MMI for model and feature-space discriminative training. In *Proc. IEEE ICASSP*, pages 4057–4060.

[Povey and Kingsbury, 2007] Povey, D. and Kingsbury, B. (2007). Evaluation of proposed modifications to MPE for large scale discriminative training. In *Proc. IEEE ICASSP*, volume 4, pages IV–321–IV–324.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[Seide et al., 2011] Seide, F., Li, G., Chen, X., and Yu, D. (2011). Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proc. IEEE ASRU*, pages 24–29.

[Thomas et al., 2013] Thomas, S., Seltzer, M. L., Church, K., and Hermansky, H. (2013). Deep neural network features and semi-supervised training for low resource speech recognition. In *Proceedings of ICASSP*, pages 6704–6708.

[Veselý et al., 2017] Veselý, K., Burget, L., and Černocký, J. H. (2017). Semi-supervised DNN training with word-selection for ASR. In *Proceedings of INTERSPEECH 2017*.

[Veselý, 2010] Veselý, K. (2010). *Parallel Training of Neural Networks for Speech Recognition*. Brno University of Technology (master thesis).

[Veselý et al., 2010] Veselý, K., Burget, L., and Grézl, F. (2010). Parallel training of neural networks for speech recognition. In *Proc. INTERSPEECH'10*, pages 2934–2937.

[Veselý et al., 2013a] Veselý, K., Ghoshal, A., Burget, L., and Povey, D. (2013a). Sequence-discriminative training of deep neural networks. In *Proc. of INTERSPEECH'13*.

[Veselý et al., 2013b] Veselý, K., Hannemann, M., and Burget, L. (2013b). Semi-supervised training of deep neural networks. In *Proceedings of ASRU*, pages 267–272.

[Wang and Sim, 2011] Wang, G. and Sim, K. C. (2011). Sequential classification criteria for NNs in automatic speech recognition. In *Proc. INTERSPEECH*, pages 441–444.

[Wessel and Ney, 2005] Wessel, F. and Ney, H. (2005). Unsupervised training of acoustic models for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(1):23–31.

[Wessel et al., 2001] Wessel, F., Schlüter, R., Macherey, K., and Ney, H. (2001). Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288–298.

[Xu et al., 2011] Xu, H., Povey, D., Mangu, L., and Zhu, J. (2011). Minimum Bayes Risk decoding and system combination based on a recursion for edit distance. *Computer Speech & Language*, 25(4):802–828.

[Young et al., 2002] Young, S., Jansen, J., Odell, J., Ollason, D., and Woodland, P. (2002). *The HTK book*. Entropics Cambridge Research Lab., Cambridge, UK.

[Young and Woodland, 1994] Young, S. J. and Woodland, P. C. (1994). State clustering in hidden markov model-based continuous speech recognition. *Computer Speech & Language*, 8(4):369–383.

[Yu et al., 2011] Yu, D., Li, J., and Deng, L. (2011). Calibration of confidence measures in speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2461 – 2473.

[Zhang et al., 2014a] Zhang, P., Liu, Y., and Hain, T. (2014a). Semi-supervised DNN training in meeting recognition. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 141–146.

[Zhang et al., 2014b] Zhang, X., Trmal, J., Povey, D., and Khudanpur, S. (2014b). Improving deep neural network acoustic models using generalized maxout networks. In *Proc. ICASSP 2014*.

Životopis

Ing. Karel Veselý

Vzdělání:

- 2010+, FIT VUT v Brně, doktorské studium (řečové technologie, neuronové sítě)
- 2008-2010, FIT VUT v Brně, magisterské studium (Počítačová grafika a multimédia)
- 2007-2008, Erasmus Avignon (předměty se zaměřením na 'řečové technologie')
- 2004-2007, FIT VUT v Brně, bakalářské studium
- 2000-2004, Gymnázium Dr. Karla Polesného, Znojmo

Profesionální kariéra:

- 2010+, Člen core-teamu vývoje open-source toolkitu Kaldi (vesis84),
- https://github.com/kaldi-asr/kaldi/graphs/contributors

Stáže, workshopy:

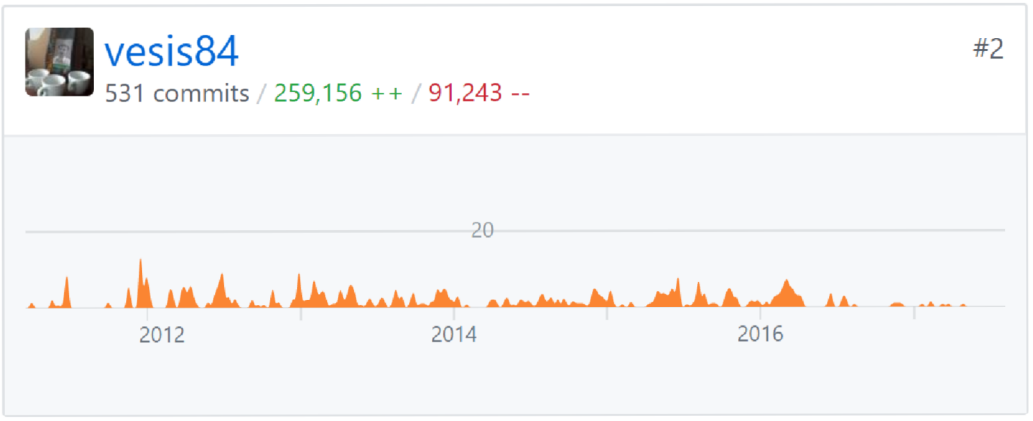
- 2015, Podzimní stáž, CLSP JHU Baltimore (2.5 měsíce),
- 2015, Letní workshop, JSALT, Seattle (1.5 měsíce),
- 2014, Letní workshop, JSALT, Praha (1.5 měsíce),
- 2013, Letní workshop, CLSP JHU Baltimore (1 měsíc),
- 2012, Letní stáž, CLSP JHU Baltimore (1 měsíc),
- 2010-2013, Letní 'Kaldi' workshopy, FIT VUT v Brně,
- 2010, Výzkumná stáž v IBM, Praha (5 měsíců),

Odborná orientace:

- Rozpoznávání řeči, detekce hlasu, hledání klíčových slov,
- Trénování neuronových sítí na GPU,

Přehled aktivit během studia,

- 2010+, Člen core-teamu vývoje open-source toolkitu Kaldi (login: vesis84), https://github.com/kaldi-asr/kaldi/graphs/contributors (celkový příspěvek 168tis. řádků z teď existujícího kódu)
- 2010-2013, Letní 'Kaldi' workshopy, Brno, podílení se na organizaci a průběhu,
- Vyhotovení posudků konferenčních článků pro konference: TSD 2012, ASRU 2013, TSD 2014, Excel 2015, ASRU 2015, Excel 2016, SLT 2016, Excel 2017, ASRU 2017,



Stáže, workshopy:

- 2015, Podzimní stáž, CLSP JHU Baltimore (2.5 měsíce),
- 2015, Letní workshop, JSALT, Seattle (1.5 měsíce),
- 2014, Letní workshop, JSALT, Praha (1.5 měsíce),
- 2013, Letní workshop, CLSP JHU Baltimore (1 měsíc),
- 2012, Letní stáž, CLSP JHU Baltimore (1 měsíc),
- 2010-2013, Letní 'Kaldi' workshopy, Brno,
- 2010, Výzkumná stáž v IBM, Praha (5 měsíců),

Výběr z publikací na mezinárodních konferencích:

(počet citací podle Google Scholar)

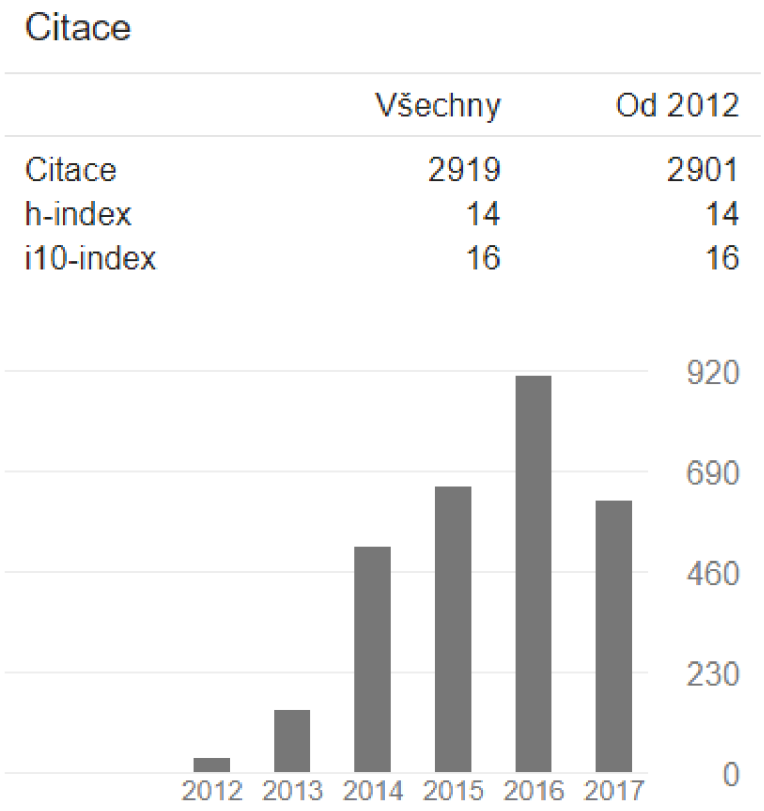
A K. Veselý, L. Burget, J. Černocký. **Semi-supervised DNN training with word selection for ASR.** Proceedings of INTERSPEECH 2017, Stockholm Sweden.
(90% autorství, zatím bez citací)

B K. Vesely, M. Hannemann, L. Burget. **Semi-supervised training of deep neural networks.** Proceedings of ASRU 2013, Olomouc Czech Republic.
(90% autorství, 54 citací)

C K. Veselý, A. Ghoshal, L. Burget, D. Povey. **Sequence-discriminative training of deep neural networks.** Proceedings of Interspeech 2013, Lyon France.
(70% autorství, 401 citací)

D K. Veselý, M. Karafiát, F. Grézl, M. Janda, E. Egorova. **The language-independent bottleneck features.** Proceedings of SLT 2012, Miami USA.
(90% autorství, 106 citací)

E D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. K. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer and K. Veselý. **The Kaldi Speech Recognition Toolkit.** Proceedings of ASRU 2011, Big Island Hawaii USA.
(10% autorství, 1700 citací)



Úplný seznam publikací

2017

- Karel Veselý, Baskar Karthick Murali, Mireia Diez, Karel Beneš. **MGB-3 BUT system: Low-resource ASR on Egyptian Youtube data.** Proceedings of ASRU 2017. **(70% autorství)**
- Karel Veselý, Lukáš Burget, Jan "Honza" Černocký. **Semi-supervised DNN training with word selection for ASR.** Proceedings of INTERSPEECH 2017, Stockholm Sweden. **(90% autorství)**
- BASKAR Murali K., KARAFIÁT Martin, BURGET Lukáš, VESELÝ Karel, GRÉZL František and ČERNOCKÝ Jan. **Residual Memory Networks: Feed-forward approach to learn long-term temporal dependencies.** Proceedings of ICASSP 2017, New Orleans USA. **(10% autorství)**

2016

- KARAFIÁT Martin, BASKAR Murali K., MATĚJKA Pavel, VESELÝ Karel, GRÉZL František and ČERNOCKÝ Jan. **Multilingual BLSTM and Speaker-Specific Vector Adaptation in 2016 BUT BABEL SYSTEM.** Proceedings of SLT 2016. San Diego USA. **(10% autorství)**
- KARAFIÁT Martin, BURGET Lukáš, GRÉZL František, VESELÝ Karel and ČERNOCKÝ Jan. **Multilingual Region-Dependent Transforms.** Proceedings of ICASSP 2016, Shanghai China. **(10% autorství)**
- PLCHOT Oldřich, MATĚJKA Pavel, FÉR Radek, GLEMBEK Ondřej, NOVOTNÝ Ondřej, PEŠÁN Jan, VESELÝ Karel, ONDEL Lucas, KARAFIÁT Martin, GRÉZL František, KESIRAJU Santosh, BURGET Lukáš, BRUMMER Niko, SWART Albert du Preez, CUMANI Sandro, MALLIDI Sri Harish and LI Ruizhi. **BAT System Description for NIST LRE 2015.** Proceedings of Odyssey 2016, Bilbao Spain. **(10% autorství)**
- VESELÝ Karel, WATANABE Shinji, ŽMOLÍKOVÁ Kateřina, KARAFIÁT Martin, BURGET Lukáš and ČERNOCKÝ Jan. **Sequence Summarizing Neural Network for Speaker Adaptation.** Proceedings of ICASSP 2016, Shanghai China. **(80% autorství)**
- ŽMOLÍKOVÁ Kateřina, KARAFIÁT Martin, VESELÝ Karel, DELCROIX Marc, WATANABE Shinji, BURGET Lukáš and ČERNOCKÝ Jan. **Data selection by sequence summarizing neural network in mismatch condition training.** Proceedings of Interspeech 2016. San Francisco USA. **(20% autorství)**

2015

- HEŘMANSKÝ Hynek, BURGET Lukáš, COHEN Jordan, DUPOUX Emmanuel, FELDMAN Naomi, GODFREY John, KHUDANPUR Sanjeev, MACIEJEWSKI Matthew, MALLIDI Sri Harish, MENON Anjali, OGAWA Tetsuji, PEDDINTI Vijayaditya, ROSE Richard, STERN Richard, WIESNER Matthew and VESELÝ Karel. **Towards Machines That Know When They Do Not Know: Summary of Work Done at 2014 FREDERICK JELINEK MEMORIAL WORKSHOP.** Proceedings of ICASSP 2015, Brisbane Australia. **(10% autorství)**
- MALLIDI Sri Harish, OGAWA Tetsuji, VESELÝ Karel, NIDADAVOLU Phani S. and HEŘMANSKÝ Hynek. **Autoencoder based multi-stream combination for noise robust speech recognition.** Proceeding of Interspeech 2015, Dresden Germany. **(20% autorství)**
- PEŠÁN Jan, BURGET Lukáš, HEŘMANSKÝ Hynek and VESELÝ Karel. **DNN derived filters for processing of modulation spectrum of speech.** Proceedings of Interspeech 2015, Dresden Germany. **(20% autorství)**

2014

- GRÉZL František, KARAFIÁT Martin and VESELÝ Karel. **Adaptation of Multilingual Stacked Bottle-neck Neural Network Structure for New Language.** Proceedings of ICASSP 2014, Florence Italy. **(10% autorství)**
- KARAFIÁT Martin, GRÉZL František, VESELÝ Karel, HANNEMANN Mirko, SZŐKE Igor and ČERNOCKÝ Jan. **BUT 2014 Babel System: Analysis of adaptation in NN based systems.** Proceedings of Interspeech 2014, Singapore. **(10% autorství)**
- KARAFIÁT Martin, VESELÝ Karel, SZŐKE Igor, BURGET Lukáš, GRÉZL František, HANNEMANN Mirko and ČERNOCKÝ Jan. **BUT ASR System for BABEL Surprise Evaluation 2014.** Proceedings of SLT 2014, Lake Tahoe USA. **(20% autorství)**
- NG Tim, HSIAO Roger, ZHANG Le, KARAKOS Damianos, MALLIDI Sri Harish, KARAFIÁT Martin, VESELÝ Karel, SZŐKE Igor, ZHANG Bing, NGUYEN Long and SCHWARTZ Richard. **Progress in the BBN Keyword Search System for the DARPA RATS Program.** Proceedings of Interspeech 2014, Singapore. **(10% autorství)**

2013

- EGOROVA Ekaterina, VESELÝ Karel, KARAFIÁT Martin, JANDA Miloš and ČERNOCKÝ Jan. **Manual and Semi-Automatic Approaches to Building a Multilingual Phoneme Set.** Proceedings of ICASSP 2013, Vancouver Canada. **(20% autorství)**
- GRÉZL František, CHALUPNÍČEK Kamil, KARAFIÁT Martin and VESELÝ Karel. **Souhrnná zpráva k projektu „Dodání anotací akustických dat, akustického modelu, jazykového modelu a výslovnostního slovníku pro arabský jazyk“ za rok 2013.** Brno: Phonexia s.r.o., 2013. **(10% autorství)**

- GRÉZL František, KARAFIÁT Martin, VESELÝ Karel and ŽIŽKA Josef. **Souhrnná zpráva k projektu „Zpracování audiovizuálních dat pro Superlectures.com“ za rok 2013.** Brno: ReplayWell, 2013. **(10% autorství)**

- KARAFIÁT Martin, GRÉZL František, HANNEMANN Mirko, VESELÝ Karel and ČERNOCKÝ Jan. **BUT BABEL System for Spontaneous Cantonese.** Proceedings of Interspeech 2013, Lyon France. **(20% autorství)**

- KARAKOS Damianos, SCHWARTZ Richard, TSAKALIDIS Stavros, ZHANG Le, RANJAN Shivesh, NG Tim, HSIAO Roger, NGUYEN Long, GRÉZL František, HANNEMANN Mirko, KARAFIÁT Martin, SZŐKE Igor and VESELÝ Karel et al. **Score Normalization and System Combination for Improved Keyword Spotting.** Proceedings of ASRU 2013, Olomouc Czech Republic. **(10% autorství)**

- RATH Shakti P., POVEY Daniel, VESELÝ Karel and ČERNOCKÝ Jan. **Improved Feature Processing for Deep Neural Networks.** Proceedings of Interspeech 2013, Lyon France. **(10% autorství)**

- VESELÝ Karel, GHOSHAL Arnab, BURGET Lukáš and POVEY Daniel. **Sequence-discriminative Training of Deep Neural Networks.** Proceedings of Interspeech 2013, Lyon France. **(70% autorství)**

- VESELÝ Karel, HANNEMANN Mirko and BURGET Lukáš. **Semi-supervised Training of Deep Neural Networks.** Proceedings of ASRU 2013, Olomouc Czech Republic. **(90% autorství)**

2012

- KARAFIÁT Martin, GRÉZL František, CHALUPNÍČEK Kamil and VESELÝ Karel. **Souhrnná zpráva za rok 2012 projektu „Anotace ruských akustických dat“.** Brno: Phonexia s.r.o., 2012. **(10% autorství)**

- MATĚJKA Pavel, PLCHOT Oldřich, SOUFIFAR Mehdi Mohammad, GLEMBEK Ondřej, D'HARO Luis Fernando, VESELÝ Karel, GRÉZL František, MA Jeff, MATSOUKAS Spyros and DEHAK Najim. **Patrol Team Language Identification System for DARPA RATS P1 Evaluation.** Proceedings of Interspeech 2012, Portland Oregon USA. **(20% autorství)**

- NG Tim, ZHANG Bing, NGUYEN Long, MATSOUKAS Spyros, ZHOU Xinhui, MESGARANI Nima, VESELÝ Karel and MATĚJKA Pavel. **Developing a Speech Activity Detection System for the DARPA RATS Program.** Proceedings of Interspeech 2012, Portland Oregon USA. **(30% autorství)**

- POVEY Daniel, HANNEMANN Mirko, BOULIANNE Gilles, BURGET Lukáš, GHOSHAL Arnab, JANDA Miloš, KARAFIÁT Martin, KOMBRINK Stefan, MOTLÍČEK Petr, QIAN Yanmin, RIEDHAMMER Korbinian, VESELÝ Karel and VU Ngoc Thang. **Generating Exact Lattices in The WFST Framework.** Proceedings of ICASSP 2012, Kyoto Japan. **(10% autorství)**

- SZŐKE Igor, FAPŠO Michal and VESELÝ Karel. **BUT2012 Approaches for Spoken Web Search - MediaEval 2012.** Working Notes Proceedings of the MediaEval 2012 Workshop, Pisa Italy. **(10% autorství)**

- VESELÝ Karel, KARAFIÁT Martin, GRÉZL František, JANDA Miloš and EGOROVA Ekaterina. **The Language-Independent Bottleneck Features.** Proceedings of SLT 2012, Miami USA. **(90% autorství)**

2011

- POVEY Daniel, GHOSHAL Arnab, BOULIANNE Gilles, BURGET Lukáš, GLEMBEK Ondřej, GOEL Nagendra K., HANNEMANN Mirko, MOTLÍČEK Petr, QIAN Yanmin, SCHWARZ Petr, SILOVSKÝ Jan, STEMMER Georg and VESELÝ Karel. **The Kaldi Speech Recognition Toolkit.** Proceedings of ASRU 2011, Big Island Hawaii USA. **(10% autorství)**

- VESELÝ Karel, KARAFIÁT Martin and GRÉZL František. **Convolutive Bottleneck Network Features for LVCSR.** Proceedings of ASRU 2011. Big Island Hawaii USA. **(90% autorství)**

2010

- VESELÝ Karel, BURGET Lukáš and GRÉZL František. **Parallel Training of Neural Networks for Speech Recognition.** Proceedings of INTERSPEECH 2010, Makuhari Japan. **(90% autorství)**

- VESELÝ Karel, BURGET Lukáš and GRÉZL František. **Parallel Training of Neural Networks for Speech Recognition.** Proceedings of TSD 2010, Brno Czech Republic. **(90% autorství)**

- VESELÝ Karel. **Parallel training of neural networks for speech recognition.** Proceedings of the 16th Conference STUDENT EEICT 2010, Brno Czech Republic. **(100% autorství)**

2007

- GRÉZL František, HRDLIČKA Pavel, VESELÝ Karel, CHALUPNÍČEK Kamil, ČERNOCKÝ Jan, KOSTKA Martin, PAVELEK Tomáš and VŠIANSKÝ Jan. **Vyhledávání slovníkových hesel hlasem.** Brno, Ministry of Industry and Trade of the Czech Republic, 2007. **(20% autorství)**

- VESELÝ Karel. **Hybrid recognizer of isolated words.** Proc. 13th Conference STUDENT EEICT 2007. Brno: Faculty of Electrical Engineering and Communication BUT, 2007 **(100% autorství)**