VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

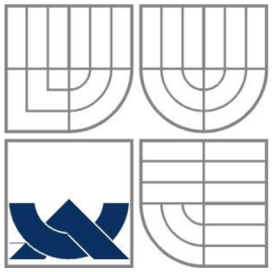
MULTI-TASK NEURAL NETWORKS FOR SPEECH RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

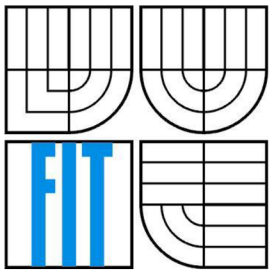
AUTOR PRÁCE
AUTHOR

BSc. Ekaterina Egorova

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MULTI-TASK NEURAL NETWORKS FOR SPEECH RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
BSc. Ekaterina Egorova

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Martin Karafiát, Ph.D.

BRNO 2014

zadani

Abstract

The first past of this Master's thesis covers theoretical investigation into the principles and usage of neural networks, including their usability for the speech recognition tasks. Then it proceeds to summarize the multi-task neural networks' operating principles and some recent experiments with them. The practical part of the semester project reports changes made to a tool for neural network training which support multi-task training. Then the preparation of the settings is described, including a number of scripts written especially for this purpose. The experiments presented in the thesis explore the idea of using articulatory characteristics of phonemes as secondary tasks for multi-task training. The experiments are conducted on two different datasets of different quality and size and representing different languages – English and Vietnamese. Articulatory characteristics are occasionally combined with different secondary tasks, such as context, to see how well they function together. A comparison is made between the networks of different sizes to see how their size affects the effectiveness of multi-task training. These experiments show that multi-task training with the use of articulatory characteristics as secondary tasks can enhance training and yield better phoneme accuracy as a result. Finally, multi-task training is embedded to a speech recognition system as a feature extractor.

Keywords

Speech recognition, neural networks, deep neural networks, multi-task neural networks.

Citation

Ekaterina Egorova: Multi-Task Neural Networks for Speech Recognition, master's thesis, Brno, FIT VUT, 2014

Multi-task Neural Networks for Speech Recognition

Statement

By this I declare that I have written this master's thesis by myself under the supervision of Ing. Martin Karafiát, Ph.D. All the sources that I have used for this project are listed in the reference section. All further information I owe to Ing. Martin Karafiát, Ph.D. and Ing. Karel Veselý.

.....
BSc. Ekaterina Egorova
27.05.2014

Thanks

By this I'd like to thank my supervisor, Ing. Martin Karafiát, Ph.D., for ideas, restless proofreading and inspiration at times when nothing seemed to work. My thanks also go to Ing. Karel Veselý for introduction to the TNet tool and his invaluable help with troubleshooting. I'd also like to thank Ing. Lukaš Burget, Ph.D., for his constructive remarks on the results. Finally, I thank my parents for listening every evening to things they don't understand a bit and for their support.

Contents

1 Introduction

Speech processing is an area of computer science which deals with extracting information from human speech by means of computation. For different applications different information is extracted. Some of the most essential tasks are language identification, speaker identification and speech recognition. Speech recognition is the most complex of them, as the ultimate goal of automatic speech recognition (ASR) is to transcribe text from acoustic input. The usability of this technique is very wide – from supporting people with different disabilities and military applications to every-day programs for dictating.

For an ASR system to be effective, it needs to incorporate knowledge about phonetics, phonotactics, grammar, vocabulary and syntax of the language in question. Moreover, the speaker factor adds further complications as each individual speaks differently. Not to mention the additional confusion from recording channel, as the microphone characteristics also influence the resulting speech recording. As ideal formalization of all this information is practically impossible, self-learning systems are used for this task.

The key notion for speech recognition is modeling. Acoustic models are statistical representations of acoustic units which are then used for recognition and language models assign probabilities to sequence of words in speech. In this master's thesis we will investigate one of the methods for training acoustic models, namely neural network approach.

For a couple of years till now, neural networks have been growing stronger in the field of speech recognition. They are typically used to train acoustic models, each of the output usually being a phoneme. Recent developments in computer power have enabled training of bigger networks which successfully rivaled and even exceeded traditional Gaussian Mixture Models. They also give uncountable possibilities as to experimenting with their structure, learning methods and many other parameters that can yield better and better results.

The purpose of this thesis is to experiment with multi-task neural networks in speech recognition with the aim of finding out if their use is effective. If they do prove to be helpful, then the next objective is to investigate for which tasks they can be effectively used.

Multi-task neural networks differ from common neural networks in that its output consists of several blocks of nodes, each block representing a different task. The network is trained for all the tasks and during the learning error backpropagates from all of them in turn. It has been known that for two interconnected tasks which are trained jointly, a significant accuracy increase is possible [7]. If the tasks do not add to each other's performance, they can still be trained with the same success as alone and used separately as before. Having one network do several connected classifications is not only good in terms of universalizing the training, it can also open new possibilities in multilingual training.

The original idea of the master's thesis is to try and use articulatory characteristics of phonemes as secondary tasks to enhance phoneme recognition accuracy. Articulatory features have been used for speech recognition before but never in a multi-task setting. Articulatory characteristics are fairly easily extracted, they are mostly universal across languages and the nature contained information suggests their suitability for multi-task training.

1.1 Thesis Structure

The theoretical part of master's thesis includes in-depth studying of neural networks problematic (Section 2) and recent papers on the topic of multi-task training (Section 3). It also includes an

introduction into articulatory phonetics, this field containing some useful knowledge which is used in the experiments (Section 4).

The practical part includes getting acquainted with the tool for neural network training – TNet, changing TNet source code allowing multi-task training (Section 6), writing scripts for extracting the necessary information from the transcription (Section 5) and testing the new setting on a small database (Subsection 7.2). The database chosen for these first experiments is TIMIT English.

The experimental part then continues with trying multi-task training with articulatory characteristics as a secondary tasks on TIMIT English (Section 7) and on BABEL Vietnamese (Section 8) databases. In the end, the new multi-task training is embedded to a BABEL Vietnamese speech recognition system as a feature extractor (Section 9). The conclusion is then made about the most effective way of using articulatory information to improve neural network training (Section 10).

In the end, the appendix section (Section 12) shows the samples of the scripts written for the needs of this work.

2 Neural Networks

This section presents an overview of neural networks, principles of their operation, basic equations and functions, strategies of the error estimation and learning. In the end, usefulness of neural networks for speech recognition is investigated.

2.1 Biological Inspiration

The idea of neural nets has appeared as an attempt to make a mathematical model of how brain works. Relatively complex tasks of vision, recognition and learning are very hard to formalize and yet animals perform them with the high degree of success. The natural consequence of these findings was an attempt to imitate the structure and the process of learning of an animal brain. A brain is a part of nervous system and is comprised of nerve cells (neurons) that are connected to each other in a form of a network. Neurobiological studies have achieved a fairly good understanding of how network of neurons work.

A biological neuron consists of a body, an axon and multiple dendrites (see Fig.1). Axon is a long protrusion which reaches dendrites of another cells. The connection between an axon of one cell and a dendrite of another forms a synapse. Electrostatic impulses are passed between cells through synapses. The magnitude of the signal received by a neuron from another neuron depends on the efficiency of the synaptic transmission, and can be thought of as the strength of the connection between the neurons. The neuron sends an output signal if the amount of received signal exceeds the threshold of the neuron [5].

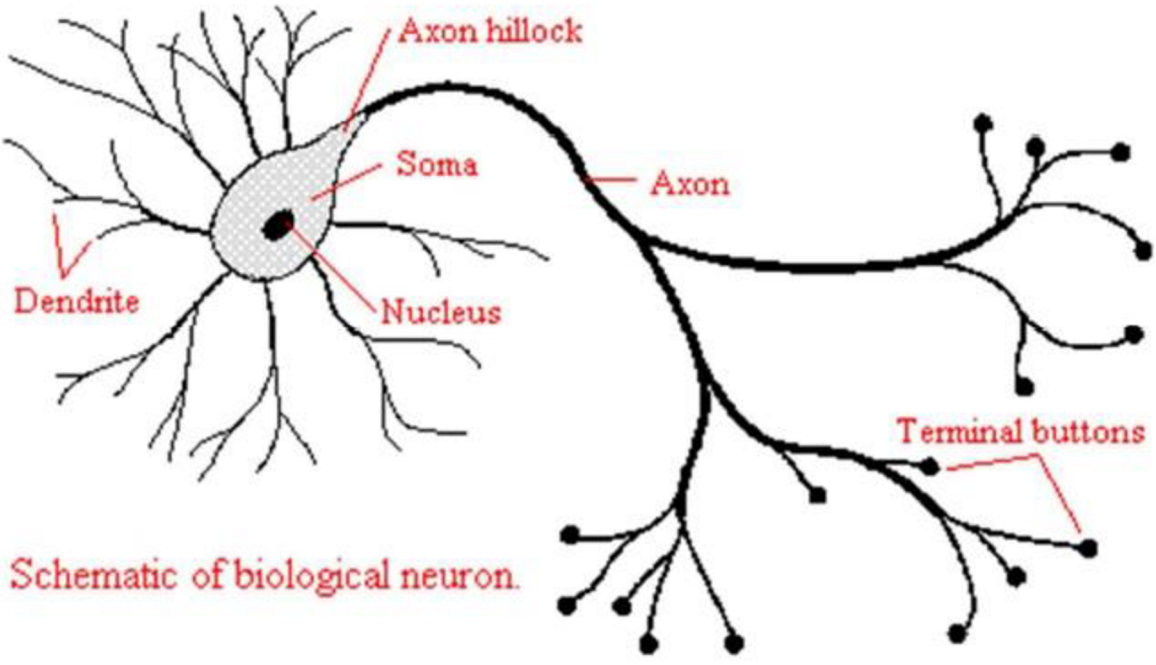


Figure 1: Biological neuron [http://vv.carleton.ca/~neil/neural/neuron-a.html].

Different parts of the nervous system have different organizations of neurons. The cerebral cortex, for example, is believed to consist of layers, though the boundaries are not strict. In this aspect, artificial neural networks are also hereditary to the natural systems.

2.2 Mathematics behind Neural Networks

2.2.1 Neuron

Alike its natural counterpart, an artificial neuron takes information from several other neurons and generates an output value which is then transmitted to several other neurons. All the connections between neurons are characterized with certain weigh, and a neuron is characterized with a function, which yields an output based on the inputs (see Fig.2).

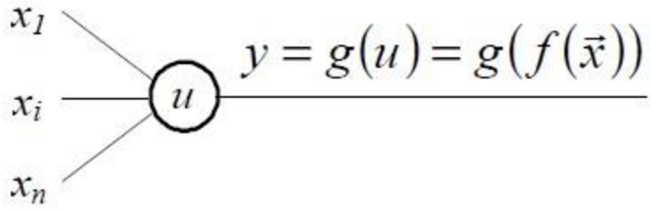


Figure 2: Artificial neuron [5].

Neural networks can differ in structure and in functions that are used in neurons. The type of network which is most commonly used in speech processing is a feedforward network with continuous activation function.

The activation function is calculated like this:

$$y = f\left(\sum_{i=1}^{N} w_i x_i - \vartheta\right), \qquad (1)$$

where n is the number of inputs, x_i is a value of i-th input, w_i is the weight of the connection between x_i input and the current node and ϑ is the neuron's threshold. The argument of the activation function is called basic function.

Usually, by adding zero input $x_0 = 1$ with the weight $w_0 = -\vartheta$, we can simplify this equation to the following:

$$y = f\left(\sum_{i=0}^{N} w_i x_i\right) \qquad (2)$$

Some of the activation functions don't take weighted summation as an argument, but rather the product of weights and inputs, or the distance between the vector of inputs and the vector of weights. However, these cases will not be discussed here in particular as they are not so widely used in speech recognition as a weighted summation.

A continuous activation function takes the result of the basic function as an argument and yields a result which lies in continuous space between certain boundaries. It's dependent on the type of the function. The most widely used continuous activation functions are sigmoid and hyperbole tangent. The functions and their graphical representations are the following:

a) Sigmoid function:

$$y = \frac{1}{1 + e^{(-u)}} \qquad (3)$$

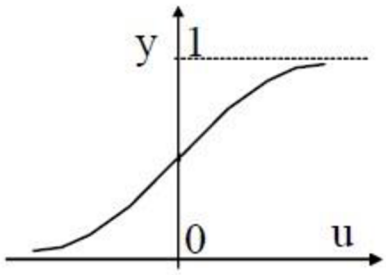


Figure 3: Sigmoid function.

b) Hyperbole tangent function:

$$y = \tan(u) \quad (4)$$

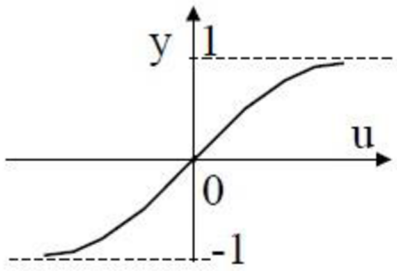


Figure 4: Hyperbole tangent function.

As you can see, a sigmoid function can be convenient to use because its outputs can be interpreted as probabilities for classification. If a layered network is used, all the nodes in the same layer usually have the same activation function.

If we want to treat the outputs of the network as probabilities of a unit belonging to a particular class, we need the outputs to conform to the following conditions: $0 \leq y_i \leq 1$ and $\sum_i y_i = 1$. So for the output layer a softmax function is usually used to define the value of *j*-th output:

$$f_j(u_1, u_2, \dots, u_M) = \frac{e^{u_j}}{\sum_{k=1}^{M} e^{u_k}}, \quad (5)$$

where *M* is the number of the nodes in the output layer, and u_j is the output of the basic function of an output neuron [5].

For a multi-task neural network softmax needs to be calculated separately for each block, as we treat each block as a separate task in which the probabilities of a unit belonging to a certain class in this task must sum to 1.

2.2.2 Network Structure

As for the structures of neural networks, only feedforward network will be considered here in detail. In this type of network, nodes are divided into subsets, called layers, so, that no connection leads from any node to a node in the preceding layer. Moreover, a feedforward network is acyclic, which means there is also no connection between nodes in one layer. The final condition is that connections are only allowed between a node in layer *x* to a node in layer *x+1* (see Fig.5).

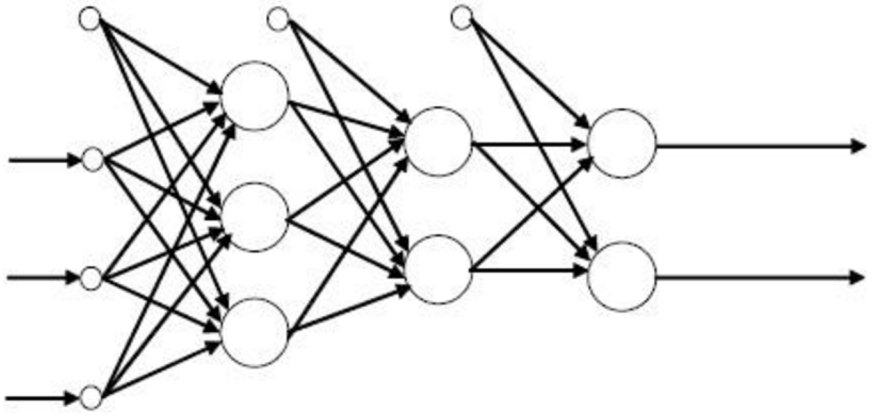


Figure 5: Feedforward network [5].

The first layer in such a network is called an input layer, the last one an output layer, and all the rest are called hidden layers. Nodes in successively higher layers abstract successively higher level features from preceding layers.

This kind of architecture with many layers of non-linear hidden units has become known as Deep Neural Network (DNN) [8].

2.2.3 Error Function

It is said that we learn by our mistakes, which is also true for neural networks. To assess the performance, some criterion is needed. In case of training with a teacher, this criterion is an error between the result yielded by the network and the etalon result. Then we can train the network to minimize this error.

Error function for an output layer looks like this:

$$E(Y, D) = \sum_{p=0}^{P} \varepsilon_p(y_p, d_p), \qquad (6)$$

where Y is the vector of outputs, D is the vector of targets, y_p and d_p are elements of Y and D correspondingly and ε in an instant error.

There are two types of instant errors which are used in neural networks: mean square error and cross-entropy.

a) Mean square error:

$$\varepsilon_p = \frac{1}{2}\sum_{j=1}^{M}(jd_p - jy_p)^2 \qquad (7)$$

Mean square error is typically used for regression tasks.

b) Cross-entropy:

$$\varepsilon_p = -\sum_{j=1}^{M} jd_p \ln(jy_p) \qquad (8)$$

Cross-entropy is typically used for classification tasks.

2.2.4 Gradient Descent

Gradient descent is a method of iterative searching for the minimum of the error function. If error function is represented in a multi-dimensional space, where one of the dimensions is the error value and all the other are parameters of the model, then a specific combination of parameters correspond with only one point of the function. For this point, a direction in which error decreases fastest is found (see Fig. 6). To this end, we calculate a gradient, which shows the direction in which error increases fastest, which is precisely the opposite direction to the one we need.

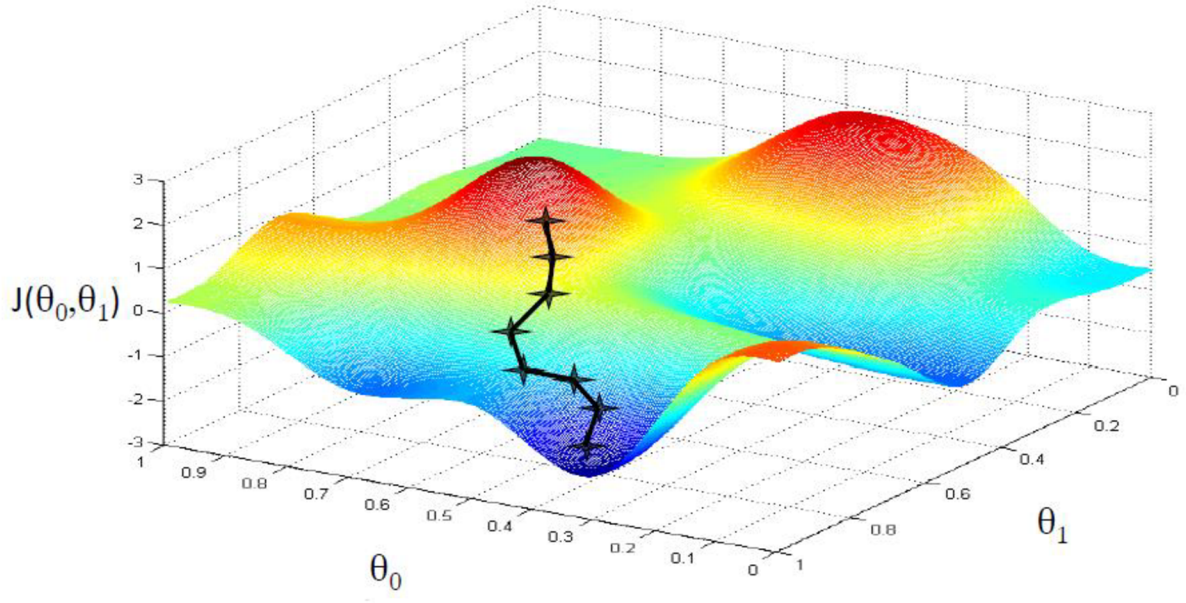


Figure 6: Gradient descent in 3D [10].

The gradient is a vector of first derivations of error E along the space dimensions:

$$\nabla E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_M}\right]^T \qquad (9)$$

After gradient is calculated, one iteration of gradient descent is defined like this:

$$w(t+1) = w(t) - \mu \nabla E, \qquad (10)$$

where μ is a learning coefficient, which defines how fast the function converges to the minimum.

If μ is too small, the learning takes a lot of time and may not converge at all. If it is too big, training can miss the optimal minimum. μ can be constant through the whole learning, but the more effective approach is to gradually make it smaller. To this end, a halving procedure is commonly used. The idea of halving is that if the accuracy on the cross validation set is getting better in one iteration by a value smaller than some threshold, the learning coefficient is divided by two. This helps make the training more precise in the end. If on some step the accuracy happens to become worse than after the previous step, the weights are reversed.

If the error function is not convex, gradient descend can find different minimums, depending on different initializations.

2.2.5 Backpropagation

After getting an error for a certain hypothesis generated by the current state of a neural network, we want neural network to learn and improve its performance. Learning for neural network means updating the weights of the connections between elements in different layers. To calculate how weights

need to be updated, the algorithm of backpropagation is used. It goes back to each node in each layer and calculates its part in the general error of the last layer.

For each node in a layer, the weights are updated as following:

$$\Delta w_i = -\mu \frac{\partial E}{\partial w_i} \qquad (11)$$

To calculate E, an instant error of each node needs to be calculated. The following formula shows the computation of error for node i in layer n-1:

$$\varepsilon_i^{n-1} = \sum_{j=1}^{M} \varepsilon_j^n f'(u_j^n) w_i^j, \qquad (12)$$

where ε_j^n is an error of node j in layer n, f' is a derivation of the activation function of node j in layer n and w_i^j is the value of weight between node i and node j.

So, the error is first calculated for the output layer, then for the last hidden layer, and all the way to the first hidden layer. After all the errors are known, each weight of the network is updated, and the next iteration of training starts with feeding new training data to the network for forward propagation.

The following illustration (see Fig.7) shows how backpropagation is going. Here, δ is the error, g is an activation function, z is the basic function and θ is the weights. In the output layer, a is the output value and y is a demanded value. All the calculation is written in the vector form for all the values in one layer.

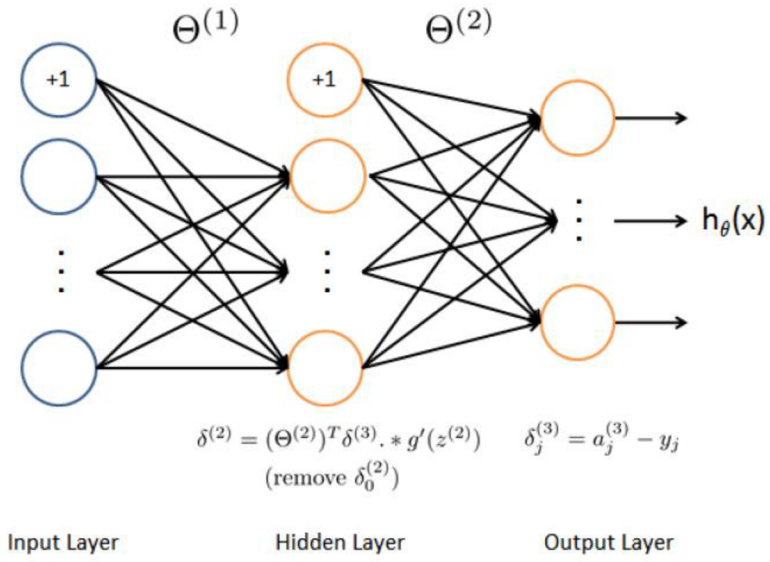


Figure 7: Backpropagation [10].

2.3 Neural Networks for Speech Recognition

Speech recognition is based on the concept of modelling: acoustic and language models are basic parts of most speech recognition systems. Acoustic model is a statistical representation of a phoneme or a triphone which is a phoneme with left and right context. Artificial neural networks, especially deep neural networks (DNNs), which are described in 2.2.2, are widely used for acoustic modelling and they are usually followed by a Hidden Markov models (HMM) system.

Thus, usually the structure of a DNN used for speech recognition is the following: the input layer of the network gets features extracted from every frame of an utterance, and the output layer usually represents phonemes or triphones or states, depending on the setting. In the context of DNN-HMM

acoustic modeling, the network's task is to compute likelihoods that can be used for the emission probabilities of the HMM.

Sometimes neural networks are also used to provide the feature vectors for Gaussian mixture models in a GMM-HMM system. The most common approach is to take the activations of the bottleneck hidden units as features, which is known as the tandem approach [8]. A more complex and more effective approach uses stacked bottleneck structure for generating features [14].

DNNs have also been proved effective for detecting sub-phonetic speech attributes (articulatory features). The total of 22 attributes, characterizing manner and place of articulation and additional phoneme features were detected with over 90% success with the help of DNN containing 5 hidden layers each with 2048 [9].

3 Multi-task Neural Networks

The idea behind multi-task learning is that it is sometimes more profitable to learn several things simultaneously rather than use separate neural networks for them. In multi-task learning, the network is trained to perform both the primary classification task and one or more secondary tasks using a shared representation (see Fig. 8 and 9). The backpropagation can be used to train artificial neural networks to learn these tasks.

The question of how tasks influence each other can have three answers. First, uncorrelated tasks might act as a source of noise, which can sometimes improve generalization when added to backpropagation. Second, adding tasks changes weight updating dynamics, so that the learning is more effective if the tasks are related. A third possibility is net capacity; multi-task networks share the hidden layer between all tasks, and reduced capacity improves generalization on these problems [7].

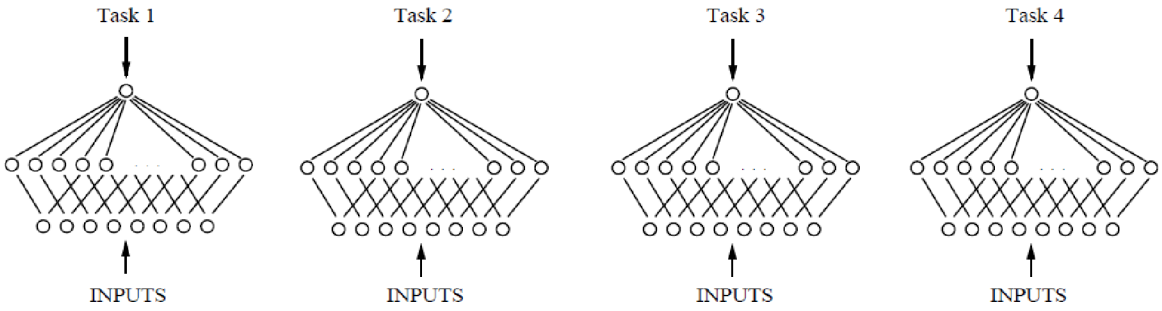


Figure 8: Separate networks for each task.

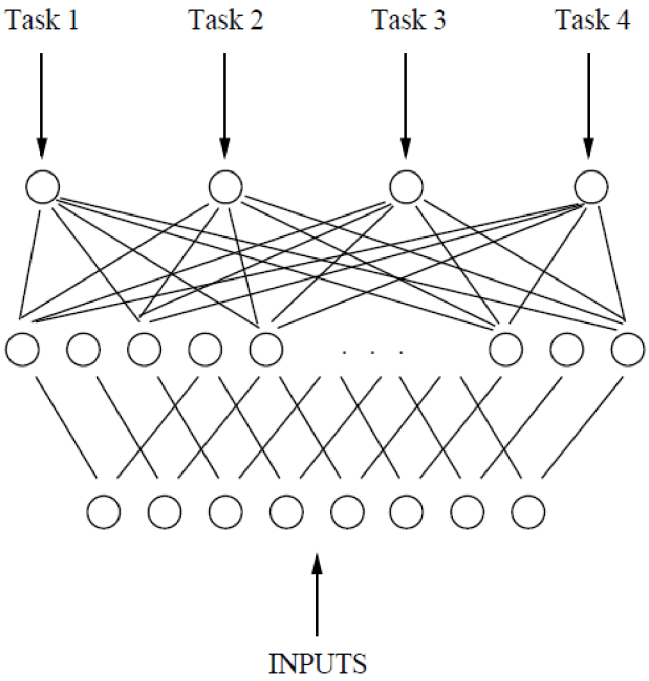


Figure 9: Multi-task neural network.

With the addition of the secondary tasks, neural network need not change its structure except for the size of the output layer. The additional parameters in the network associated with the secondary tasks are used only to aid in the training of the network. After training is complete, the portion of the network associated with the secondary tasks is discarded and the classification is performed identically to a conventional single task classifier.

Experiments have shown that if the tasks are related, performance is increased with addition of more tasks [7]. If tasks are related, it is likely that the computed subfeatures needed by some tasks will sometimes be useful to other tasks. These subfeatures may not be learnable by the tasks that "eavesdrop" on them if they are trained separately, but might prove valuable sources of information when they are learned for other tasks.

The main drawback of multi-task neural networks is the difficulty of choosing secondary tasks. The term "related" is far from being a formal definition, and although sensible guesses can be made about the effectiveness of one of another secondary task, it is impossible to be sure that this will appear to be true. So the only way to choose secondary tasks is experimenting.

3.1 Multi-task Neural Networks in Speech Recognition

For speech recognition multi-task structure opens uncountable possibilities of usage, as a lot of speech characteristics are interdependent. Multi-task neural networks are not new and have been experimented with since 1989, when classic NETtalk application used one net to learn both phonemes and their stresses [7]. But this is only one of the many possible combinations of tasks that could yield some improvement in speech recognition. Some of the possible settings can include, for example, joined learning of segmental and suprasegmental characteristics (e.g. tones in tonal languages), phoneme labels and phoneme characteristics, or phoneme inventories of different languages in a multilingual task. In a recent paper, [3], the following secondary tasks were explored: the phone label, the phone context, and the state context. The best results were achieved with phone context as a secondary task, with 1.4% decrease in error rate.

4 Articulatory Characteristics of a Phoneme

When working with products of human activity, which speech is, it is always helpful to understand how this activity looks like. The production of speech lies in the competence of articulatory phonetics which deals with the physiological nature of speech.

Speech is produced by humans with a help of a complex system called a vocal tract. This system has a source of sound (air, which is expired by lungs, goes through the vibrating vocal cords and produces sound) and a number of resonators which can be configured and reconfigured to produce different sound effects. The position of different parts of vocal tract thus defines the sound which is produced, and the dependence between the vocal tract configuration and acoustic characteristics of a sound is more or less direct. From this knowledge springs up the classification of the sounds of speech according to the configuration and activity of different parts of vocal tract when producing it. Let's have a look at some of these articulatory characteristics, as they will be used further in the experiments.

4.1 Vowel / Consonant Opposition

The most obvious articulatory characteristic of a sound, which even naive speakers recognize, is whether it is a vowel or a consonant. In case of a consonant, a sound is articulated by creating a complete or partial closure of a vocal tract, whereas when a vowel is pronounced, the vocal tract is open and there is no identifiable place of the biggest pressure.

Being so very different, vowels and consonants are also characterized differently. Consonants are usually defined by a place and manner of articulation and phonation. Vowels are characterized by height, backness and roundedness. We'll start with discussing articulatory characteristics of consonants, as they lie in the basis for the experiments. Then we proceed to tackle articulatory characteristics of vowels.

4.2 Characteristics of Consonants

I have chosen consonant characteristics to form secondary tasks in the experiments, so they are discussed here with more detail than articulatory characteristics of vowels.

4.2.1 Place of Articulation

A place of articulation is a place where an obstruction is created when pronouncing a consonant. This obstruction is between a passive location, which is some part of the upper-back palate of the mouth, and an active articulator, which is the lower surface of the mouth. Figure 10 lists the possible places of articulation, both active and passive. Figure 11 shows the possible interactions between these parts and what sounds they result in.

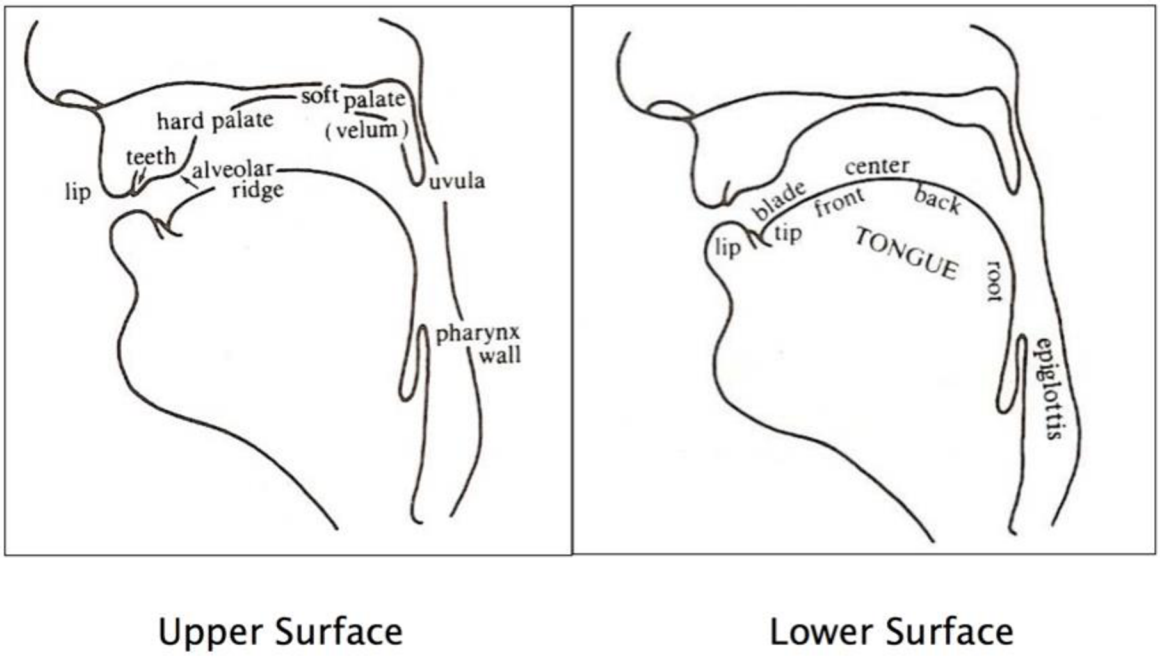


Figure 10: Places of articulation [http://www.ling.upenn.edu/courses/ling520/LectureNotes2.html].

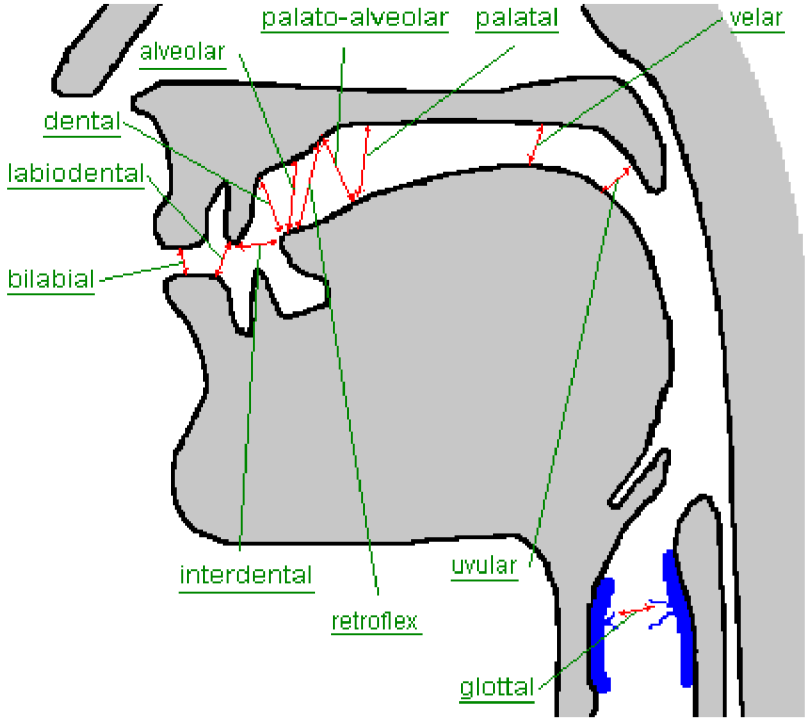


Figure 11: Types of consonants according to the place of articulation [http://www-01.sil.org/mexico/ling/glosario/E005ci-PlacesArt.htm].

4.2.2 Manner of Articulation

The manner of articulation characterizes the nature of the interaction between the active and passive parts of the vocal tract in the process of producing a consonant. We will discuss 6 main modes of interaction, although they can be subdivided more particularly.

- **Stop.** Includes a full occlusion of air flow in a vocal tract with subsequent release. *Examples: /p/, /t/, /k/.*

- **Nasal.** Implies an occlusion of a vocal tract with air released through nasal cavity instead. *Examples: /m/, /n/.*
- **Fricative.** Characterizes a consonant produced by a turbulent airflow which results from drawing an articulator close to the surface of the mouth without a complete closure. *Examples: /f/, /s/, /x/.*
- **Affricate.** A sound which begins as a stop, but whish release phase is an affricate. *Examples: /ts/, /tʃ/, /dʒ/.*
- **Tap/Flap.** A consonant produced by a very short closure of a vocal tract. *Examples: /ɾ/.*
- **Trill.** A sound produced by a vibration articulator. *Examples: /r/, /R/.*
- **Approximant.** A consonant produced with little obtrusion in the vocal tract. *Examples: /j/, /w/.*

4.2.3 Phonation

Phonation is another name for voicing which is characterizing if the vocal cords are under enough pressure to oscillate and thus produce sound of not. Thus a lot of the consonants have its voiced/unvoiced counterparts, sharing their place and manner and differing only in voicing. *Examples: /s/ and /z/.*

4.2.4 Classification of Consonants According to Their Articulatory Characteristics

Articulatory characteristics provide a great basis for classification of phonemes, so a lot of attempts have been made to standardize this classification. IPA (International Phonetic Association) has succeeded in it. Their classification is now an international standard for phoneticians all around the world. IPA table of consonants which was used for the experiments of this work is shown on Figure 12. This includes only pulmonic consonants, but as we will not meet any non-pulmonic ones in this work, this will suffice.

THE INTERNATIONAL PHONETIC ALPHABET (revised to 2005)

CONSONANTS (PULMONIC) © 2005 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

Figure 12: Classification of consonants according to their articulatory characteristics [11].

4.3 Characteristics of Vowels

Vowels are not produced by a closure of vocal tract and they are all inherently voiced. So, it's impossible to classify them by the same rules as consonants. Instead, they are categorized according to their backness, height and roundedness.

Backness characterizes the position of the tongue in the horizontal dimension – from back to front. Height characterizes its position in vertical dimension – from open to close. Roundness indicates participation of lips in the pronunciation – they can be either rounded or not. Figure 13 shows possible positions of vowels according to these characteristics [13].

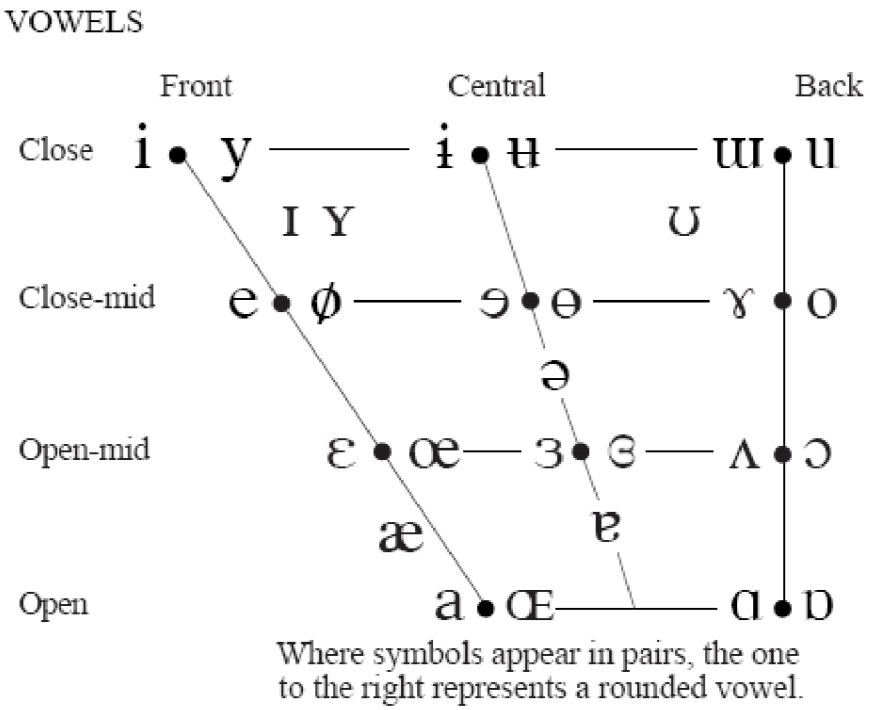


Figure 13: Articulatory characteristics of vowels [11].

4.4 Suprasegmental Characteristics

Suprasegmental characteristics are elements of speech that can coexist with multiple segments (phonemes) and cannot be discretely ordered by them. Some examples of suprasegmental characteristics are stress, tone and intonation.

Tone is characterized by the use of pitch to distinguish words and word forms. In tonal languages, the same syllable with a different tone has different meaning. Some of the experiments in this work are conducted on a Vietnamese dataset, and Vietnamese is a tonal language with 6 tones: mid-level, low falling, low rising, high broken, high rising and low broken (see Fig.14) Change of pitch affects phoneme's acoustic characteristics a lot, so tone secondary task was chosen as one of the secondary tasks on Vietnamese dataset.

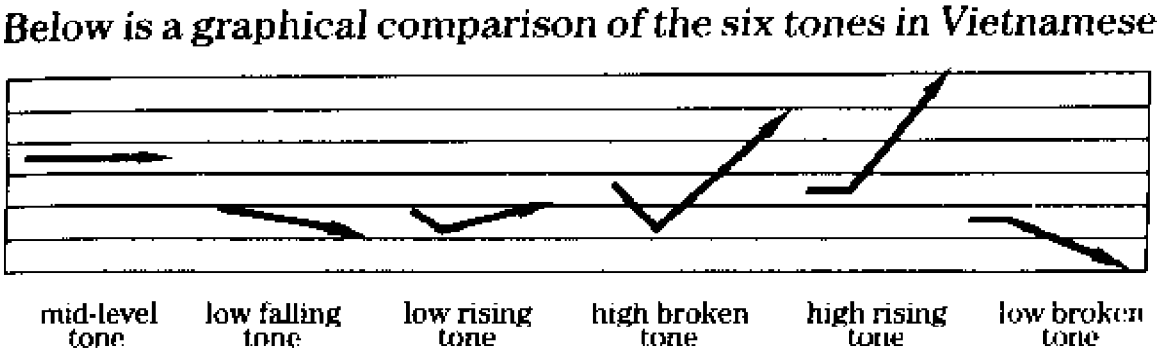


Figure 14: Vietnamese tones [http://dylansung.tripod.com/flux/vietnam.htm].

4.5 Coarticulation

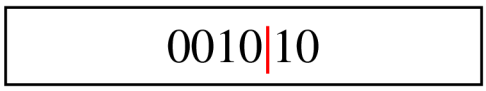
Coarticulation is a process occurring in continuous speech, in which the articulation of a phoneme is influenced by the preceding or a following phoneme and becomes more like it. Coarticulation usually affects place of articulation, as the movement of vocal tract is continuous and in the process of switching between two phonemes vocal tract inevitably finds itself in some middle position between the two articulations. The result of the coarticulation effect is that phonemes may differ depending on the context, that is, on what are the preceding and the following phonemes [13].

As for our experiments context-independent phoneme labels have been chosen, which means coarticulation effects were not represented in this model, context information was one of the secondary tasks chosen for the multi-task experiments. It has already been found useful in such function [3].

5 Target Generation

In a multi-task setting, the network is trained to deal with more than one task: the primary task of phoneme classification and one or more secondary tasks. The resulting network must create a hypothesis for every frame in the utterance assuming either which phoneme or which unit of a secondary classification is represented in it. To train such a network, target files need to be created.

Target file is a file which defines the correct output in each block for each frame in a training sequence. It looks like a matrix with the number of rows equaling the number of frames in an utterance. The number of columns equals the number of units in the output layer of the network. The target matrix consists only of zeroes and ones, ones representing the correct output and zeroes representing wrong outputs. As there are several tasks in the output, there must be only one correct output in each task. To better imagine it, let's take a hypothetic language with four "phonemes" and two secondary characteristics: let them be voiced and unvoiced, for example. For such a language, one row of the output target file can look like this:



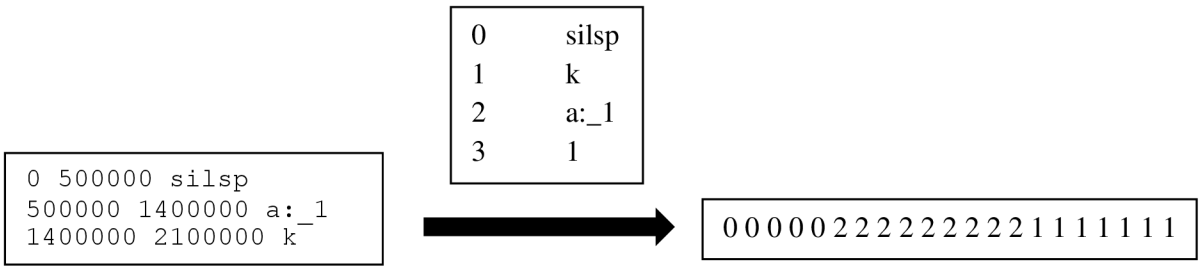
Here, the red line shows the division between the two tasks. This row means that the frame represents "phoneme" number three, and it is voiced.

The information for generation of target files can be easily extracted from the transcription files by a sequence of scripts, especially written for this work. Let's proceed to investigate the process step by step.

5.1 Step 1. Transcription to Label Arrays

The transcription file, also called master label file or simply mlf, contains all the necessary information, and is usually structured as following: the first column is the beginning time of a speech segment in 100 nanoseconds, the second column is the end of the speech segment in 100 nanoseconds and the third column is phoneme label corresponding to this segment. In Vietnamese there is also information about the beginning and the end of syllables (See Appendix 12.2.1 for an example of a transcription file).

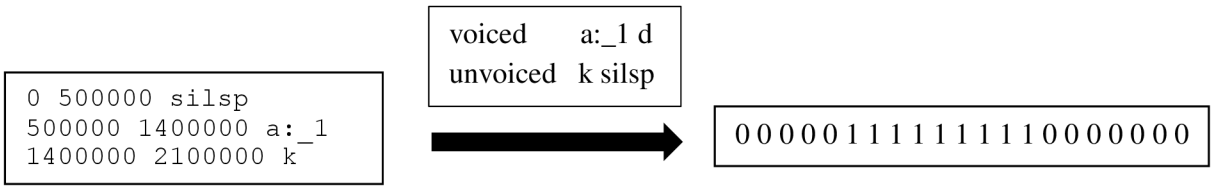
Each phoneme label is encoded as a number. In our small example we will have only four "phonemes" which will be represented by numbers from 0 to 3. At this step, we want to get an array of the length equaling the number of frames in an utterance with its elements being numbers representing phonemes.



Note that while in the transcription file the time is in 100 nanoseconds, we need to extract the length of each segment in frames, and frame rate is 10 milliseconds, so 500000 gives us 5 frames.

Extracting of gender was done in a similar way, only the information was taken from the first letter of the file label: “m” of “f”, and the array consisted of zeroes and ones only.

Extraction of articulatory features was a bit trickier, as each phoneme had to be checked against the list of phonetic questions (see Appendix 12.2.2 for an example). If a phoneme was found to possess a certain characteristic, the numerical label for this characteristic was put in a corresponding file. On our small example with voiced and unvoiced phonemes, we’ll get the phonation array which looks like this:

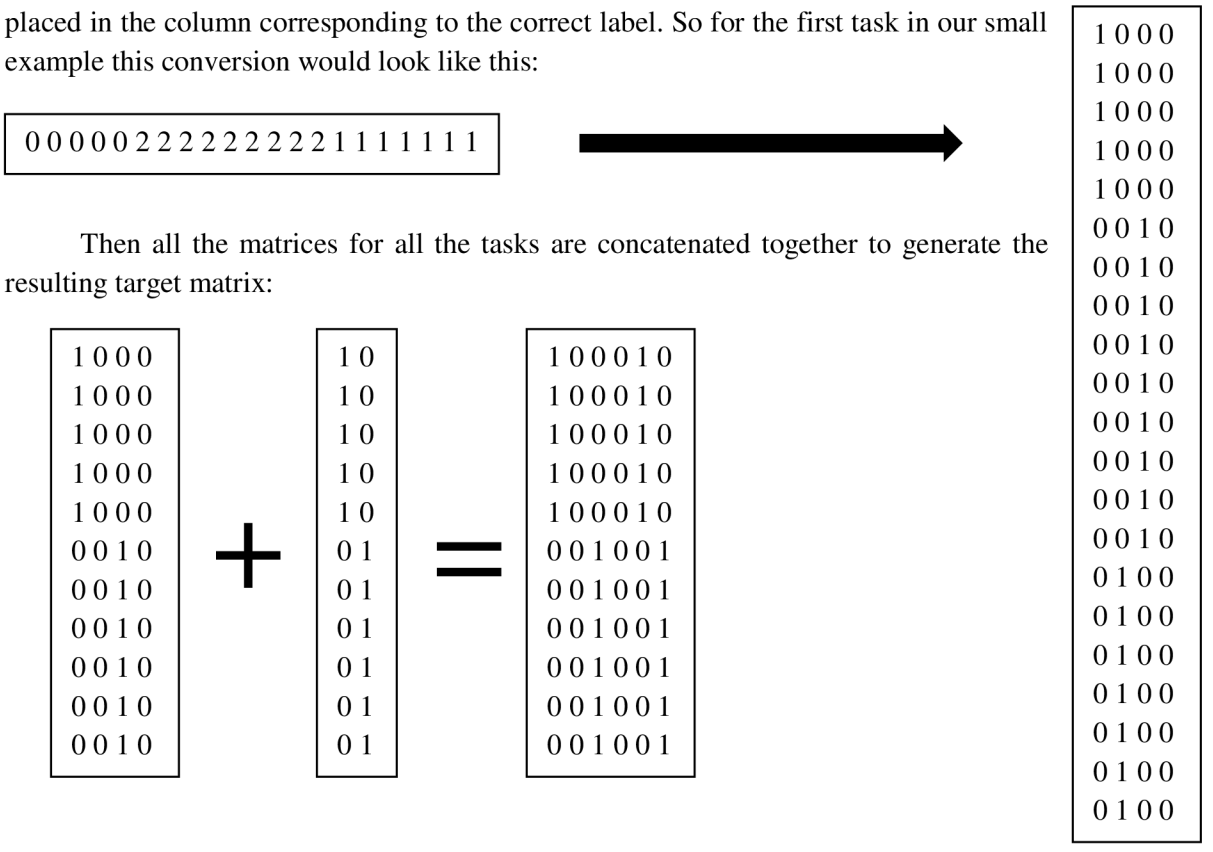


Extraction of tones was also done in a similar way, but tone labels were extrapolated for the whole length of the syllable.

The scripts for doing all this were written in Python, as it is a very convenient programming language for processing text files. The examples of a couple of such scripts can be found in Appendices 12.2.3 and 12.2.4.

5.2 Step 2. Label Arrays to Target Matrices

At this step, the files with the label arrays are read in MatLab and converted to the target array. First, a matrix is created for each label array with the number of rows equaling the length of this array, which is the number of frames, and with the number of columns equaling the number of labels. So for example if we use labels from 0 to 3, there will be 4 columns in the corresponding matrix. The arrays are generated to be full of zeroes, and then in each row corresponding to a frame a one is placed in the column corresponding to the correct label. So for the first task in our small example this conversion would look like this:



Then all the matrices for all the tasks are concatenated together to generate the resulting target matrix:

After the target matrix is ready, it is written to the file of the same format as the feature files, with the help of writehtk.m script (free software, Copyright Mike Brookes 2005). An example of a MatLab script for step 2 can be seen in Appendix 12.3. MatLab was chosen for this part of the target preparation because it's a convenient tool to work with matrices.

5.3 Step 3. Generating Training Lists

After the target files are ready, training lists have to be generated. The lists for multitask training contain the path to the feature file and then on the next line the path to the target file, created according to the procedure described above. So the resulting lists look like this:

```
Feature file 1
Target file 1
Feature file 2
Target file 2
…
```

6 TNet Extension

The second half of the practical part of the semester project included extending Neural Network Trainer TNet framework[1] for multi-task training.

TNet tool, which is used as a basis for this project, is a free tool developed at FIT Speech for parallel training of neural networks. It supports both classification and regression tasks. As an activation function either sigmoid, hyperbole tangent or softmax can be chosen. For objective function, the user can choose between mean square error and cross entropy. The tool allows initialization and training of networks of any sizes of user's choice and supports training parallelization.

There are two possibilities for training. The first one is to use master label file (mlf) which contains transcription to provide target information. The second possibility is to create custom target feature files and submit pairs of features and targets for training instead of features and mlf. This second possibility is particularly useful for multi-task training, as targets for the output layer must be customized according to the chosen secondary tasks as described in the previous section.

In multi-task training, the output layer is virtually divided into several parts, each part containing outputs for one task. Each of these blocks has one correct output according to the target file, so when it comes to calculating the error, they need to be treated separately.

For the multi-task experiments, ObjFun (src/TNetLib/ObjFun.cc) class was extended with the new functionality: MultiCrossEntropy function (see appendix 12.1), which goes block by block over the output neurons and calculates cross entropy error function in each block separately. Errors are accumulated in an array with one value for each block, averaged across the number of blocks and backpropagated.

The number and size of blocks is set up during the network initialization. It can be seen in the end of the generated init file which contains the initial weights of the network. Then the information about blocks is passed to the new flavor of objective function where it is used for cutting outputs and targets into corresponding blocks for separate error calculation.

The new flavor of objective function is incorporated in the overall structure of the framework, which means that its usage can be called through parameter setting, as all the other objective functions. The check is added to assure that the new objective function is used only with block softmax, as it doesn't make sense with the others.

[1] http://speech.fit.vutbr.cz/cs/software/neural-network-trainer-tnet

7 TIMIT English Experiments

This section describes the set of experiments on the first of the two databases used in this work – TIMIT. For the purpose of testing the new functionality of TNet, an experiment was made with a simple setting of two target tasks: phoneme labels and gender of the speaker. After that, the key idea of the diploma was tested: if articulatory secondary tasks prove helpful in the context of multi-task training. The previously made experiments with context secondary tasks have been replicated and combined with the newly introduced articulatory tasks. In the end, a comparison of the effectiveness of different network structures for multitask training has been done.

7.1 Database Description

TIMIT Acoustic-Phonetic Continuous Speech Corpus is a corpus of read speech, which is designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems. TIMIT contains broadband recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences. The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16kHz speech waveform file for each utterance. Corpus design was a joint effort among the Massachusetts Institute of Technology (MIT), SRI International (SRI) and Texas Instruments, Inc. (TI). The speech was recorded at TI, transcribed at MIT and verified and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST). The TIMIT corpus transcriptions have been hand verified. Test and training subsets, balanced for phonetic and dialectal coverage, are specified. Tabular computer-searchable information is included as well as written documentation.

TIMIT corpus was chosen for the baseline experiments because of its relatively small size, which allows for fast testing and for the availability of additional information to be chosen as a second task. However, it is necessary to remember that the performance on TIMIT do not necessarily translate into performance improvements on large vocabulary tasks with less controlled recording conditions and much more training data [8].

7.2 Gender Secondary Task

This first small experiment was designed as a test of the functionality of the new setting, so the secondary task was chosen to be rather simple – gender of the speaker, which consists of only two classes. The network size was also small, with just one hidden layer of 500 units. The output layer was of size 41: 39 phoneme labels + 2 gender labels. Features were extracted on 25ms frames with 10ms shift with the use of 23 Mel frequency filterbank. On TIMIT, it usually takes the network around 12 iterations to converge, with occasional halving of the learning rate.

To test that the system learn not only the primary task – phonemes, but also the secondary task – gender, the part of the output corresponding to gender task was not simply discarded, but used to produce gender hypothesis. The setting forbade to change between male and female hypothesis for different frames in one utterance.

On the phoneme task, the system yielded about the same result as the baseline. (The baseline was trained on a neural network with the same dimensions with the output layer of the size of 39). The accuracy rate for both tasks can be seen in the table (see Tab. 1). Note, that it can differ depending on the initialization, but this difference usually doesn't exceed 0.2 percent.

baseline	multi-task
69.56	69.41

Table 1: Accuracy on phoneme label task.

On the gender task, the multi-task system yielded accuracy 98.07, which means that it successfully learned both tasks.

There was no increase in accuracy for the phoneme label task due to the introduction of the secondary task, and this can be explained by the fact that these two tasks are not very correlated. In comparison with [3], where context information was chosen to serve as the secondary task and which yielded reasonable improvement, it looks discouraging. However, it can help to remember that phoneme models are not very dependent on the gender of the speaker and certainly not vice versa, so to expect some improvement would be over-optimistic. Anyway, learning two tasks in one network can still be helpful and resources-saving.

Overall, the small experiment on the TIMIT database with two tasks - gender and phoneme label - has shown that the new code works correctly and that further more complex experiments can be made with this setting.

7.3 Articulatory Secondary Tasks

In this section, the original idea of the usability of articulatory characteristics of a phoneme is tested on TIMIT database and the results of these experiments are analyzed.

7.3.1 Preparation of Articulatory Secondary Tasks

The object of this first set of experiments was to show if classification according to articulatory characteristics added as a secondary task increases the ability of neural network to learn the main task – phoneme recognition. For this setting, a bigger network had to be chosen for reasons explored in detail in subsection 7.4. Thus, the following set of experiments was conducted on a network with four hidden layers, consisting of 2048 units each.

Articulatory characteristics for secondary tasks can be easily extracted from phonetic questions (see Section 5) used for clustering in ASR systems.

The following groups of articulatory characteristics were chosen:

- place ('none', 'front', 'semi-front', 'central', 'semi-back', 'back', 'bilabial', 'labiodental', 'dental', 'alveolar', 'post-alveolar', 'palatal', 'velar', 'glottal')
- manner ('none', 'open', 'open-mid', 'close-mid', 'close', 'plosive', 'nasal', 'tap-flap', 'affricate', 'fricative', 'approximant', 'lateral approximant')
- vowel or consonant ('none', 'vowel', 'consonant')
- voice ('voiced', 'unvoiced')
- additional characteristics ('none', 'rounded', 'unrounded', 'diphthong')

Each of these groups serves as one secondary task. The most important requirement for these groups is that no phoneme can belong to more than one class of each group, as there may be only one correct answer in each task. Note that as the classification needed to accommodate both vowels and consonants, backness characteristics were moved with “place”, height – with “manner” and roundness – with “additional characteristics”. This last group was formed from all the characteristics left in the questions which didn't fit into other groups. Item "none" in each group was created to accommodate silences, which don't belong to any other group. In the "voice" group, silences belong to "unvoiced" class.

Targets for calculating of the objective function were generated according to these groupings. For example, group 'place' consists of 14 outputs and the frame corresponding to phoneme /p/ will have one in the seventh position in this group (bilabial) and the rest will be zeros. In total, the target matrix will have 6 ones in the row of the number of the frame corresponding to phoneme /p/, on the positions of the following characteristics: 'p', 'bilabial', 'plosive', 'consonant', 'unvoiced', 'none'.

7.3.2 Experimental Results

The results for this set of experiments can be found in the table below:

	%PhnAcc	%diff
baseline	72.3	0
all articulatory	71.5	-0,8
place	72.5	+0,2
manner	72.5	+0,2
vowel/consonant	72.4	+0,1
voice	72.3	0
additional	72.1	-0,2
place+manner+vowel/cons	72.6	+0,3

Table 2: TIMIT articulatory secondary tasks.

First, all the secondary tasks were added at once, which resulted in 6 tasks overall, but, as shown in the table, it didn't work and accuracy only decreased. So a set of experiments was made with each secondary task separately to see which ones help and which ones don't.

It turned out that the information about the place and manner of articulation of a phoneme and if a phoneme is a vowel or a consonant can yield some improvement in accuracy, the information about voicing doesn't help, and additional information is even making matters worse, which is predictable, because this information is the least universal.

7.3.3 Combining Context and Articulatory Secondary Tasks

As was mentioned before (see Section 3), the experiments with context secondary tasks have already shown good results on TIMIT database [3]. So I've decided to replicate them and combine them with articulatory secondary tasks.

The context tasks represent information about the phoneme label for the previous and next frame. There are two context tasks: left context and right context, each consisting of the same number of outputs as the man task, 39 in case of TIMIT. This gives us a total 117 units in the output layer. The left context of the first frame in an utterance is set to silence, and the right context of the last frame in an utterance is likely set to silence.

	%PhnAcc	%diff
baseline	72,3	0
context	72,6	+0,3
context+place+manner+vowel/cons	72,8	+0,5

Table 3: Adding context task to articulatory tasks.

As can be seen in the Table 3, context secondary tasks has proved to work well on TIMIT, as expected [3], so context was combined with the best combination of articulatory tasks from previous

experiments. The total gain on TIMIT with the help of multi-task training reaches 0.5%, which is quite substantial, if not stunning.

7.3.4 Sanity Check Experiment

It has been suggested that the reason multitask training helps is the effect of regularization. That is, we provide more information, which is not necessarily new (that is, we can infer any articulatory characteristic from the phoneme label), but as the amount of information is bigger, the solution found is less likely to be overfit. This supposition would suggest that adding any relevant information as a secondary task will help.

To check that the tasks really overhear on each other, and that the choice of secondary tasks is important, a sanity check experiment has been conducted. For this, the secondary task was chosen to be exactly the same as the primary task – phoneme labels. If this addition turns out to be helpful, it means multitask training is just a question of regularization, and if it doesn't improve the accuracy, it proves that multitask training is really something more.

baseline	2x phonemes
72,3	71,9

Table 4: Accuracy for baseline and multi-task with two equal tasks.

The results in Table 4 show that this suggestion is not true and just adding more information without thinking of its helpfulness doesn't result in the same accuracy rate as a carefully chosen multitask setting.

7.4 Dependency on the Network Size

It has been noticed during the network tuning that multitask training doesn't work with networks of smaller sizes. Multitask actually worked worse than the baseline on the initial setting of one hidden layer with 500 units and started helping the training only when the neural network was increased to the size of four hidden layers, each consisting of 2048 units.

Further work aimed to explore how changing the number of layers affected the performance in presence of secondary tasks. For the sake of speed, this set of experiments was conducted on TIMIT database, on the networks with the number of hidden layers ranging from 1 to 4, each consisting of 2048 units. After the addition of the fifth layer, the learning efficiency stops increasing as the neural network gets over-trained, so adding more hidden layers doesn't seem reasonable for such a small dataset.

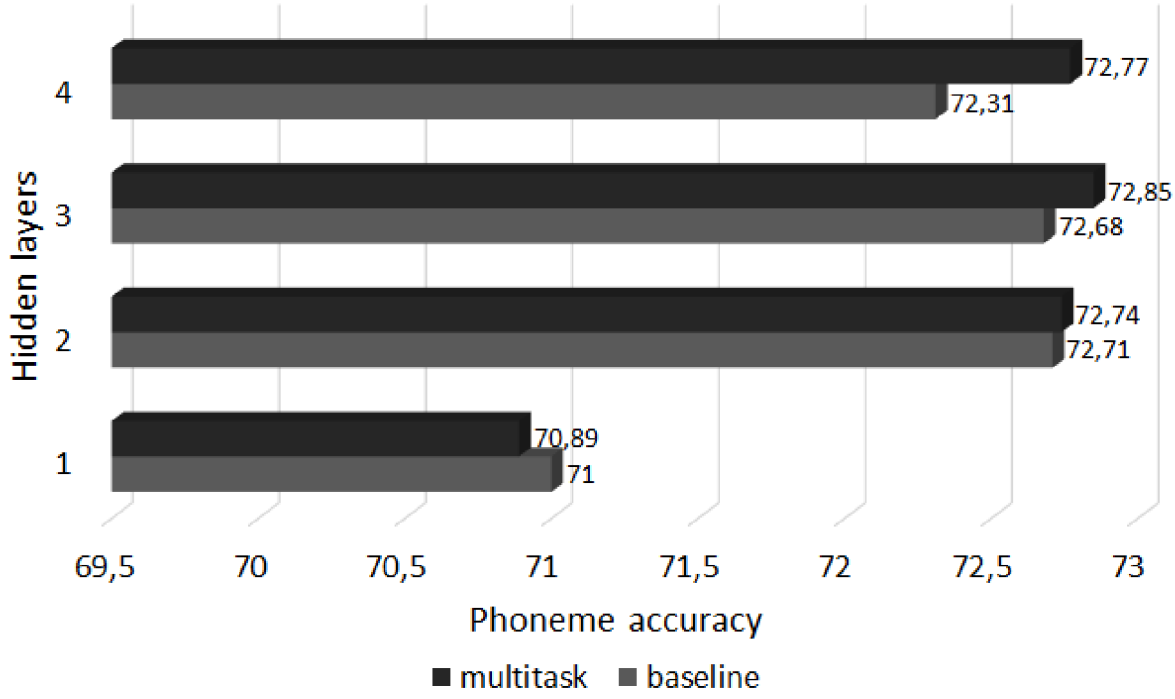


Figure 15: Dependency between the size of the network and the efficiency of multitask training.

As can be seen on Figure 15, multitask setting helps most on bigger (more than two hidden layers) neural networks, which probably means that smaller networks are not able to accommodate additional information from secondary tasks.

7.5 TIMIT English Conclusion

It has been confirmed that for TIMIT English addition of some articulatory characteristics as secondary tasks may be helpful. The examples of these successful choices are place and manner of articulation and the information about whether a phoneme is a vowel or a consonant. However, the information about voicing and other characteristics haven't proved helpful. Further improvement can be achieved with the addition of context information as an additional secondary task. The total improvement of accuracy rate reaches 0.5%, which is not a breakthrough result, but can be helpful anyway. At least it allows to state that these numbers are not results of different initialization and that multi-task learning does improve the learning.

It has been shown that the network should be big enough to be able to accommodate the increase in the amount of information it needs to learn resulting from the addition of secondary tasks. On a small network, though, multi-task training can even yield worse results than the regular training.

8 BABEL Vietnamese Experiments

Although very well annotated and of a good quality, TIMIT database is fairly small and too devoid of noise to represent most real-life speech recognition situations. Therefore, to test TIMIT conclusions, another database and another language, very different from English, have been chosen.

8.1 Database and System Description

The following set of experiments is conducted on Limited Language Pack (about 10h of clean speech) of Vietnamese spontaneous telephone speech collected for BABEL project (Vietnamese IARPA-babel107b-v0.7 (VI)).[2] The main task in the Vietnamese experiments is also phoneme label classification. Because of the tones, the number of phoneme labels is much bigger than in TIMIT experiments and equals to 91 (25 consonants and 45 vowels). The experiments in this section were also performed on a network with 4 hidden layers, consisting of 2048 units each.

The NN input features for Vietnamese are 15 critical band energies (squared FFT magnitudes binned by Mel scaled filter-bank and logarithmized) concatenated with estimates of F0 and probability of voicing. It makes 17 dimensional feature stream. The estimation of F0 (implemented according to [15]) is based on normalized cross correlation function.

The output targets were generated in the similar way as for TIMIT, see section 5 for details and appendix 12.2-3 for scripts.

8.2 Articulatory Secondary Tasks

Vietnamese is a syllabic tonal language with a phoneme set which differs a lot from that of English, so the choice of articulatory characteristics must also be different. The following articulatory characteristics were chosen for the experiments:

- place (’none’, ’front’, ’central’, ’back’, ’bilabial’, ’labiodental’, ’dental’, ’alveolar’, ’palatal’, ’velar’, ’glottal’)
- manner (’none’, ’open’, ’mid’, ’close’, ’plosive’, ’nasal’, ’affricate’, ’fricative’, ’approximant’)
- vowel or consonant (’none’, ’vowel’, ’consonant’)
- voice (’voiced’, ’unvoiced’)
- additional (’none’, ’rounded’, ’unrounded’, ’aspirated’, ’glottalized’, ’unaspirated’)

As with TIMIT, separate experiments were conducted first using each articulatory characteristic as the sole secondary task to see which ones help the learning.

[2] http://www.iarpa.gov/Programs/ia/Babel/babel.html

	%PhnAcc	%diff
baseline	35,5	0
place	36,2	+0,7
manner	37,4	+1,9
vowel/cons	36,3	+0,8
voice	37,0	+1,5
additional	36,1	+0,6
all articulatory	35,6	+0,1

Table 5: Articulatory secondary tasks on Vietnamese.

As can be seen from the table, the situation with Vietnamese is a bit different than with TIMIT. For once, all the secondary tasks do help improve the phoneme accuracy. This improvement is also much bigger, probably due to the worse quality and greater size of the database. However, when takes all together, secondary tasks stop being so effective as when used separately, probably because the pattern becomes too complex to be learned.

8.3 Tones Secondary Task

Another characteristic feature of Vietnamese language are tones which are suprasegmental characteristics affecting pitch, length, contour melody, intensity and phonation of a phoneme. Although formally only vowels are "tone-carriers", because of co-articulation, all the phonemes in a syllable are affected by the tone (especially voiced ones). The tonal information in BABEL data is attached only to vowels (e.g. for phoneme /O/ there are 6 variants: /O_1/, /O_2/, /O_3/, /O_4/, /O_5/ and /O_6/) but consonants in the same syllable, even voiced ones, are not marked. Therefore, as tones modify the pronunciation of phonemes which otherwise are labeled equally, it makes sense to add tonal information as a secondary task.

Thereby, the tonal secondary task was tried for Vietnamese. This secondary task consisted of 7 outputs: six tones and one 'no tone' for silences. All the phonemes in a syllable are targeted with the tone of the vowel in the syllable (see subsection 5.1). Again, as with TIMIT gender secondary task, instead of just discarding the second half of the output layer (secondary task), it was used to generate tones hypothesis mlf in order to check not only how phoneme recognition is influenced be adding tonal information as a secondary task, but also how tonal recognition is influenced by training the tone models together with phoneme models.

	baseline	multitask
phonemes	35,5	35,4
tones	43,9	51,0

Table 6: Phoneme accuracy for phonemes and tones classification with baseline and multi-task setting.

The results of these experiments are ambiguous. On the one hand, adding tonal information as a secondary task doesn't enhance phoneme recognition, and even results in a slight accuracy drop. On the other hand, the recognition of tones works much better if the network for this is not trained separately, but with the addition of phonemes as a secondary task. This second result may be encouraging, as the increase in accuracy is very big – 6%, but unfortunately the usability of tone recognition is much narrower than of phoneme recognition.

8.4 Context Secondary Tasks

As with TIMIT English, it has been decided to test if context information is as useful as a secondary task for Vietnamese as for English. Context targets were generated in the same way as they were for TIMIT, and the resulting output layer consists of 273 units: three times the number of phoneme labels – 91.

	%PhnAcc	%diff
baseline	35,5	0
context	32,2	-3,3

Table 7: BABEL Vietnamese context secondary task

Unlike on TIMIT, on BABEL addition of context information doesn't prove successful and results in a drastic decrease in accuracy rate. A possible explanation for this can be that BABEL database is bigger than TIMIT database and instead of just a several phrases, although context rich, contains spontaneous speech. Or it can be because of Vietnamese being structurally a very different language – syllabic, in which prediction of context on the syllable boundaries can be a difficult task.

8.5 BABEL Vietnamese Conclusion

The main resume of this second set of experiments is that the effectiveness of multi-task training depends a lot on the database and the language it's applied to. In case of Vietnamese, adding all articulatory characteristics can help increase phoneme accuracy and by a much bigger percent than on TIMIT. However, context information proves useless on Vietnamese. Experiments with tones have shown that adding tones as a secondary task doesn't help phoneme recognition, but the recognition of tones improves a lot if the network is trained together with phonemes.

9 Bottleneck Training

Using neural network for phoneme recognition is not too effective in itself. Much more useful is to incorporate neural networks in a Large Vocabulary Continuous Speech Recognition system (LVCSR). In the further experiments, multi-task neural networks are used for feature extraction in Hidden Markov Model (HMM) based system.

In our BABEL Vietnamese system, bottleneck features are used, as they have been shown to outperform probabilistic features [12]. BN features use five-layers MLP with a narrow layer in the middle (bottle-neck). The fundamental difference between probabilistic and BN features is that the latter are not derived from the class posteriors. Instead, they are obtained as linear outputs of the neurons in the bottle-neck layer. This structure makes the size of the features independent of the number of the MLP training targets. It is important for multi-task experiments, as it is easy to replace the phoneme targets by more complex multi-task targets, while retaining a small feature vector without a need of a dimensionality reduction.

The bottle-neck MLP training process is the same as for probabilistic features and employs all five layers. During feature extraction only the first three layers are involved. It is illustrated in Figure 16.

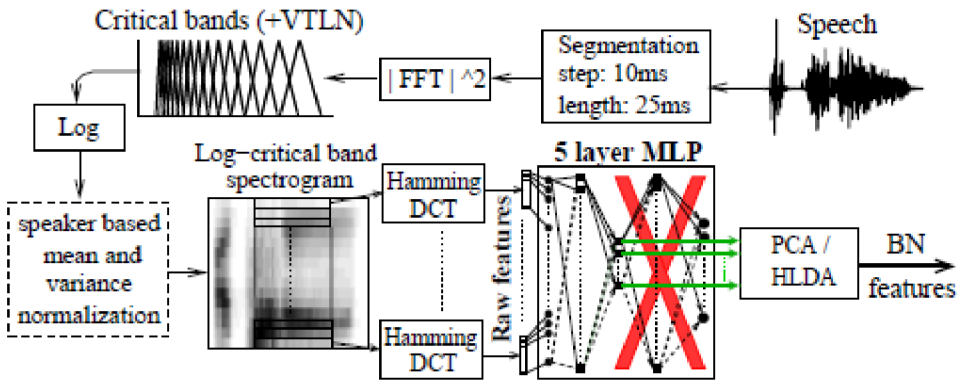


Figure 16: Block diagram of the Bottle-Neck feature extraction [12].

An even more complex neural network structure which has been found to overcome standard bottleneck is a stacked bottle neck neural network [16]. It contains two neural networks: the bottleneck outputs from the first one are stacked, down-sampled, and taken as an input vector for the second neural network. This second neural network has again a bottleneck layer, of which the outputs are taken as input features for GMM/HMM recognition system.

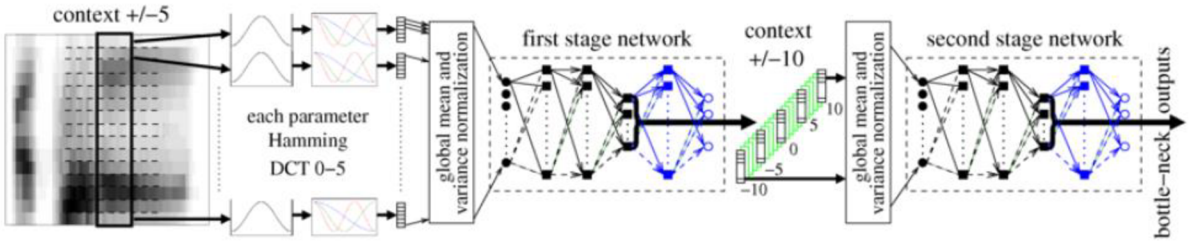


Figure 17: Stacked bottleneck neural network feature extraction [14].

9.1 Articulatory Secondary Tasks for Bottleneck Networks

In the following experiments two neural network architectures were used for feature extraction: one with 30 linear units in the bottleneck layer and another with 80 units in the bottleneck layer. In both cases the bottleneck layer is the third out of four hidden layers, the rest consisting of 2048 units each, as usual.

On top of the features obtained from the bottleneck layer a GMM-HMM recognition system was trained. In all the following experiments the results are given in word error rate (WER) instead of phoneme accuracy.

	%WER	%diff
baseline	72,2	0
place	73,9	1,7
manner	73,4	1,2
vowel/cons	*	*
voice	73,6	1,4
additional	73,2	1
all articulatory	*	*

Table 8: WER on Vietnamese with BN 30 units.

Several problems were encountered during this training. First, some settings were not able to train at all (marked with an asterisk). Different initializations and different training parameters have been chosen, but the learning always got stuck at about 1% CV accuracy. The most probable reason for it is that there is too much of too diverse information to be compressed into just 30 units of the bottleneck. Anyway, even in the settings in which training went normally, the resulting WER is still worse for the multitask setting than for the baseline, the difference being about 1%.

In order to amend for possible insufficiency of 30 units of the bottleneck for accommodating all the information provided by the multi-task training, another setting was tried, now with 80 units in the bottleneck layer.

	%WER	%diff
Baseline	72,7	0
Manner	74,0	+1,3
all articulatory	74,3	+1,6

Table 9: WER on Vietnamese with BN 80 units.

Not all the possible combinations of the secondary tasks have been tried here, as the same dynamics as with 30 units in a bottleneck can be observed from just a couple of experiments.

Unfortunately, having more units in the bottleneck layer doesn't seem to help. One possible conclusion is that for word error rate articulatory characteristics don't add any useful information. Another explanation can be that some other bottleneck structure should be found to accommodate articulatory multi-task settings.

9.2 Context Secondary Tasks for Bottleneck Networks

After the memorable failure with articulatory secondary tasks in a bottleneck setting, context secondary task has been tried on the bottleneck network with both 30 and 80 units in the bottleneck layer.

	30 units	80 units
baseline	72,2	72,7
context	72,9	73,0

Table 10: Context secondary task for bottleneck training

On 30 units WER increase, although smaller than in case of articulatory characteristics, is still present. On 80 units the situation is exactly the same. Remembering that context secondary tasks didn't help even with phoneme recognition setting on BABEL Vietnamese, this is not surprising.

9.3 Bottleneck Training Conclusion

To our disappointment, we have to conclude that multitask training doesn't work well with bottleneck networks, and provides a lamentable error rate increase on the task of word recognition. There can be several possible explanations to that.

First, it is possible that several different tasks just don't compress well into the bottleneck units, and even taking a fairly big (80) amount of units in the bottleneck layer doesn't help.

Second, it is possible that the effectiveness of a dictionary just overrides the effectiveness of everything else for word recognition. Even in case of one or several phonemes wrongly recognized the word can be recognized correctly, so additional information about context or articulatory characteristics of phonemes is useless.

Thirdly, it just may be that the training parameters which have been found to be optimal for normal training are not suitable for multi-task training. Tuning these parameters may be material for future research. Another possibility of improving the results can be to use stacked bottleneck method for feature extraction instead of a single bottleneck.

As it is, the conclusion seem to be that multi-task learning is better left for more simple tasks, as phoneme recognition. In a more complex language recognition system they prove useless.

10 Conclusion and Future Plans

The aim of this master's thesis was to experiment with multi-task neural networks in the context of speech recognition and particularly test the usability of articulatory characteristics of phonemes as secondary tasks for the multi-task setting.

The theoretical part of the thesis started with obtaining a more in-depth understanding of neural networks, the principles of their operating and their usage in speech recognition. Then the more specific problematic of multi-task neural networks was researched, including the more recent paper experimenting with their usability for speech recognition.

As the original idea of the thesis was to work with articulatory characteristics of phonemes, an introduction into articulatory phonetics has been made in the end of the theoretical part.

The practical part included modifying the existing TNet framework for training neural networks to allow new functionality of multi-task neural networks. To this end, a new method has been added and its functionality was integrated into existing the framework. Moreover, a number of scripts have been written for extracting articulatory, context and other information from the transcriptions and for creating target files with this information.

For the experimental part of the thesis, two databases have been chosen: TIMIT English and BABEL Vietnamese. The first is broadband recordings of read speech, and second is spontaneous telephone conversation.

The first short experiment on TIMIT database was with just one secondary task – speaker's gender. It has shown full functionality of the new multi-task setting of TNet. It was proved that a multi-task neural network can be successfully trained to perform more than one task without any accuracy decrease on any of the tasks.

Further experiments were conducted with different articulatory characteristics as secondary tasks. It has been proven that for TIMIT the information about the place and manner of articulation and whether a phoneme is a vowel or a consonant is helpful, and the best phoneme accuracy increase (0,5%) has been reached by the combination of context and some articulatory secondary tasks.

The last group of experiments on TIMIT has shown that multi-task training works best on bigger neural networks which are able to accommodate the increased amount of information.

Similar experiments on Vietnamese have shown that, unlike in English, all articulatory secondary tasks enhance phoneme accuracy rate. However, context information on Vietnamese has proven useless, as well at the information about tones.

In the end, multi-task neural networks have been integrated in an LVCSR system in the role of feature extractors. For this, bottleneck neural network structures wit 30 and 80 units in the bottleneck layer have been used. However, either because bottleneck structure is not able to learn several tasks simultaneously or because on the higher level of recognition – on word error rate measurements – articulatory characteristics don't play such a big role, multi-task training has proved useless in this setting. Context information also didn't add anything to the recognizer.

The resume of this work can be that using multi-task learning can help a bit in certain cases, but the benefits are far from tremendous. It is very difficult to guess what settings will work and what will not, and this also differs across languages and datasets, so multi-task training is not predictable either. However, if used in favorable circumstances, adding secondary tasks to the training may help to gain a couple percents of phoneme accuracy, and in some cases it may be very helpful.

10.1 Publications

The experimental results have been presented at EEICT student conference[3] (2nd place in the group) and included in a paper for INTERSPEECH 2014[4], with review results still unknown.

10.2 Future Research Plans

Experimenting with multi-task neural networks opens more suggestions and possibilities than is possible to test. As shown above, on different languages and different databases different things prove to be effective, so each new setting needs to be experimented with before it can be said if multi-task training will be helpful for it. Instead of listing all the possible further developments of the topic, which will be a tedious task, I have chosen just a couple of the more promising areas where research can be done.

First, future work with multi-task neural networks can include tuning bottleneck training to achieve successful training in a multi-task setting. It is also possible to train a stacked bottleneck with multitask target, as two-step compression of the information can prove more effective. Another possibility is to use outputs themselves as emission probabilities.

Second, it is possible to train a multi-lingual multi-task neural network with articulatory characteristics. As most of them are shared across languages, especially if the languages are chosen to be close enough, this approach can be proved advantageous.

Of course, it is also always possible to choose new interesting secondary tasks that can enhance the learning. Although tones have proved less than helpful on Vietnamese, in other languages suprasegmental characteristics may be more helpful. The information about the speaker and about the recording devices can also be tested. There are literally endless possibilities.

[3] http://www.feec.vutbr.cz/EEICT/
[4] http://www.interspeech2014.org/

11 References

[1] Ch. Bishop. *Pattern Recognition and Machine Learning*. Springer Science + Business Media, LLC, 2006. ISBN 0-387-31073-8.

[2] K. Veselý. *Paralelní trénování neuronových sítí pro rozpoznávání řeči*, diplomová práce. Brno, FIT VUT v Brně, 2010

[3] M. L. Seltzer and J. Droppo. *Multi-task learning in Deep Neural Networks for improved phoneme recognition*. In Proceedings of ICASSP 2013 Vancouver, Canada

[4] K. Mehrotra, C.K. Mohan, S. Ranka. *Elements of artificial neural networks*. MIT Press, 1997. ISBN 0-262-13328-8.

[5] F. Zbořil. Soft Computing lectures. https://www.fit.vutbr.cz/study/courses/SFC/private/.cs

[6] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. *The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM*. 1986.

[7] R. Caruana. *Multitask learning: A knowledge-based source of inductive bias*. Machine Learning, vol. 28, pp. 41–75, 1997.

[8] G. Hinton, L. Deng, Y. Dong, G. Dahl & Co. *Deep Neural Networks for Acoustic Modeling in Speech Recognition*. 2012

[9] Y. Sabato, M. Siniscalchi, L. Deng, C. Lee. *Boosting Attribute and Phone Estimation Accuracies With Deep Neural Networks for Detection Based Speech*. In Proceedings of ICASSP 2012

[10] A. Ng. Coursera Machine Learning Course. https://www.coursera.org/course/ml

[11] *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet.* Cambridge University Press, 1999

[12] F. Grézl, P. Fousek. *Optimizing Bottle-neck Features for LVCSR*. In: 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing. Las Vegas, Nevada: IEEE Signal Processing Society, 2008, pp. 4729-4732. ISBN 1-4244-1484-9.

[13] P. Ladefoged, and K. Johnson. *A course in phonetics*. 6th ed. Boston: Wadsworth. 2011.

[14] M. Karafiát, F. Grézl, M. Hannemann, J. Černocký. *BUT Neural Network Features for Spontaneous Vietnamese in BABEL.* In: 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing. Florence, Italy.

[15] D. Talkin. *A Robust Algorithm for Pitch Tracking (RAPT).* In Speech Coding and Synthesis, New York, 1995.

[16] F. Grézl, M. Karafiát, L. Burget. *Investigation into bottle-neck features for meeting speech recognition*. In Proc. Interspeech 2009, 2009, number 9, pp. 2947–2950.

12 Appendices

12.1 ObjFun Appendix

```
class MultiCrossEntropy : public ObjectiveFunction
  {
   public:
    MultiCrossEntropy()
     : ObjectiveFunction(), frames_(0), error_(0), corr_(0)
    { }

    ~MultiCrossEntropy()
    { }

    ObjFunType GetType()
    { return MULTI_CROSS_ENTROPY; }

    const char* GetName()
    { return "<multi_cross_entropy>"; }

    ObjectiveFunction* Clone()
    { return new MultiCrossEntropy(*this); }

    void Evaluate(const Matrix<BaseFloat>& net_out, const
Matrix<BaseFloat>& target, Matrix<BaseFloat>* err);

    size_t GetFrames()
    { return frames_; }

    double GetError()
    { return error_; }

    void SetDimensions(Vector<int> mDim)
    {
      mDim_.Init(mDim.Dim());
      for (int i = 0; i < mDim.Dim(); i++)
        mDim_[i] = mDim[i];
    }

    std::string Report();

    void MergeStats(const ObjectiveFunction& inst);

   private:
    size_t frames_;
    double error_;
    size_t corr_;
    Vector<int> mDim_;

  };

void MultiCrossEntropy::Evaluate(const Matrix<BaseFloat>& net_out, const
Matrix<BaseFloat>& target, Matrix<BaseFloat>* err)
{
  if(!mDim_.IsInitialized()){
        KALDI_ERR << "Dimensions vector not initialized!";
  }
  if(net_out.Cols() != target.Cols()) {
```

```
    KALDI_ERR << "Nonmatching dim of data : net_out " << net_out.Cols()
              << " target " << target.Cols();
  }
  int sum = 0;
  for(int i = 0; i < mDim_.Dim(); i++){
    sum = sum + mDim_[i];
  }
  if(sum != net_out.Cols()) {
    KALDI_ERR << "sum of vectors values does not match data dimensions!" ;
  }
  if(err->Rows() != net_out.Rows() || err->Cols() != net_out.Cols()) {
    err->Init(net_out.Rows(),net_out.Cols());
  }

  err->Copy(net_out);
  err->AddScaled(-1,target);

  int curBlockCol = 0;
  Vector<double> sumerr(mDim_.Dim());
  int corr = 0;

  sumerr.Zero();

  for(int blockNum = 0; blockNum < mDim_.Dim(); blockNum++) {

    //allocate and copy block submatrix
    Matrix<BaseFloat> net_out_sub_block;
    Matrix<BaseFloat> target_sub_block;
    net_out_sub_block.Init(net_out.Rows(), mDim_[blockNum]);
    net_out_sub_block.Copy(net_out.Range(0, net_out.Rows(), curBlockCol,
mDim_[blockNum]));
    target_sub_block.Init(target.Rows(), mDim_[blockNum]);
    target_sub_block.Copy(target.Range(0, target.Rows(), curBlockCol,
mDim_[blockNum]));

    //assertions for blocks copying
    assert(mDim_[blockNum] == net_out_sub_block.Cols());
    assert(mDim_[blockNum] == net_out_sub_block.Cols());

    //collect max values
    std::vector<size_t> max_target_id(target.Rows());
    std::vector<size_t> max_netout_id(target.Rows());

    //check correct classification
    for(size_t r=0; r<net_out_sub_block.Rows(); r++) { //goes through rows
      int id_netout =
FindMaxId(net_out_sub_block[r].pData(),net_out_sub_block.Cols());
      int id_target =
FindMaxId(target_sub_block[r].pData(),target_sub_block.Cols());
      if(blockNum == 0 && id_netout == id_target) corr++;
      max_target_id[r] = id_target; //store the max value
      max_netout_id[r] = id_netout;
    }

    //compute loss function
    for(size_t r=0; r<net_out.Rows(); r++) {
      if(target(r,max_target_id[r]) == 1.0) {
      //pick the max value..., rest is zero
        BaseFloat val = log(net_out_sub_block(r,max_target_id[r]));
        if(val < -1e10f) val = -1e10f;
        sumerr[blockNum] += val;
```

```
      } else {
      //process whole posterior vect.
        for(size_t c=0; c<net_out_sub_block.Cols(); c++) {
          if(target_sub_block(r,c) != 0.0) {
            BaseFloat val =
target_sub_block(r,c)*log(net_out_sub_block(r,c));
            if(val < -1e10f) val = -1e10f;
            sumerr[blockNum] += val;
          }
        }
      }
    }

    //go to the beginning of the next block
    curBlockCol = curBlockCol + mDim_[blockNum];
  }

  double sumerror = sumerr.Sum()/mDim_.Dim();

  error_ -= sumerror;
  frames_ += net_out.Rows();
  corr_ += corr;
}

std::string MultiCrossEntropy::Report() {
  std::stringstream ss;
  ss << "Xent:" << error_ << " frames:" << frames_
     << " err/frm:" << error_/frames_
     << " correct[" << 100.0*corr_/frames_ << "%]"
     << "\n";
  return ss.str();
}

void MultiCrossEntropy::MergeStats(const ObjectiveFunction& inst) {
  const MultiCrossEntropy& xent = dynamic_cast<const
MultiCrossEntropy&>(inst);
  frames_ += xent.frames_; error_ += xent.error_; corr_ += xent.corr_;
}
```

12.2 Python Appendix (for Vietnamese)

12.2.1 Source Transcription File

```
#!MLF!#
"*/90559i1_20120608_184439_01236_01762-01270-01323.lab"
0 500000 silsp -2341.275146
500000 1500000 silsp -1056.613159
1500000 1600000 a:_1 -112.157066 sil-a:_1+l -616.848877 A
1600000 2000000 a:_1 -394.328949
2000000 2100000 a:_1 -110.362877
2100000 2200000 l -104.193024 E_1-l+o_1 -471.323975 L\303\224
2200000 2300000 l -96.849724
2300000 2600000 l -270.281219
2600000 3200000 o_1 -473.293396 l-o_1+sil -1120.518555
3200000 3300000 o_1 -91.544006
3300000 3900000 o_1 -555.681091
3900000 4700000 silsp -503.755432 sil -27926.580078
4700000 5100000 silsp -299.730530
5100000 5400000 silsp -237.993744
.
"*/90559i1_20120608_184439_01236_01762-01675-01762.lab"
0 1000000 silsp -1810.889160
1000000 1100000 silsp -108.765495 sp -571.199524
1100000 1400000 silsp -337.070892
1400000 1500000 silsp -121.915863
1500000 2200000 d< -831.994507 sil-d<+a:_1 -1180.271729 \304\220ANG
2200000 2300000 d< -128.749054
2300000 2500000 d< -219.528229
```

...

12.2.2 Source Question File

```
front a:  a:_1 a:_2 a:_3 a:_4 a:_5 a:_6 a  a_1 a_2 a_3 a_4 a_5 a_6 E  E_1
E_2 E_3 E_4 E_5 E_6 e  e_1 e_2 e_3 e_4 e_5 e_6 i  i_1 i_2 i_3 i_4 i_5 i_6
mid @:  @:_1 @:_2 @:_3 @:_4 @:_5 @:_6 @  @_1 @_2 @_3 @_4 @_5 @_6 E  E_1
E_2 E_3 E_4 E_5 E_6 e  e_1 e_2 e_3 e_4 e_5 e_6 O  O_1 O_2 O_3 O_4 O_5 O_6
o  o_6 o_1 o_2 o_3 o_4 o_5
nasal m  m_1 m_2 m_3 m_4 m_5 m_6 n  n_1 n_2 n_3 n_4 n_5 n_6 N  N_1 N_2 N_3
N_4 N_5 N_6 J  J_1 J_2 J_3 J_4 J_5 J_6
velar k x G  G_1 G_2 G_3 G_4 G_5 G_6 N  N_1 N_2 N_3 N_4 N_5 N_6 w  w_1 w_2
w_3 w_4 w_5 w_6
close e  e_1 e_2 e_3 e_4 e_5 e_6 i  i_1 i_2 i_3 i_4 i_5 i_6 o  o_6 o_1 o_2
o_3 o_4 o_5 u  u_1 u_2 u_3 u_4 u_5 u_6 _1  _1_1 _1_2 _1_3 _1_4 _1_5 _1_6
palatal c J<back>  J<back>_1 J<back>_2 J<back>_3 J<back>_4 J<back>_5
J<back>_6 ts<back> J  J_1 J_2 J_3 J_4 J_5 J_6 j  j_1 j_2 j_3 j_4 j_5 j_6
aspirated    th
glottalized b<  b<_1 b<_2 b<_3 b<_4 b<_5 b<_6 d<  d<_1 d<_2 d<_3 d<_4 d<_5
d<_6
bilabial p b<  b<_1 b<_2 b<_3 b<_4 b<_5 b<_6 m  m_1 m_2 m_3 m_4 m_5 m_6
open a:  a:_1 a:_2 a:_3 a:_4 a:_5 a:_6 a  a_1 a_2 a_3 a_4 a_5 a_6 E  E_1
E_2 E_3 E_4 E_5 E_6 O  O_1 O_2 O_3 O_4 O_5 O_6
voiced b<  b<_1 b<_2 b<_3 b<_4 b<_5 b<_6 d<  d<_1 d<_2 d<_3 d<_4 d<_5 d<_6
J<back>  J<back>_1 J<back>_2 J<back>_3 J<back>_4 J<back>_5 J<back>_6 v
v_1 v_2 v_3 v_4 v_5 v_6 z  z_1 z_2 z_3 z_4 z_5 z_6 G  G_1 G_2 G_3 G_4 G_5
G_6 m  m_1 m_2 m_3 m_4 m_5 m_6 n  n_1 n_2 n_3 n_4 n_5 n_6 N  N_1 N_2 N_3
N_4 N_5 N_6 J  J_1 J_2 J_3 J_4 J_5 J_6 l  l_1 l_2 l_3 l_4 l_5 l_6 r<back>
r<back>_1 r<back>_2 r<back>_3 r<back>_4 r<back>_5 r<back>_6 j  j_1 j_2 j_3
j_4 j_5 j_6 w  w_1 w_2 w_3 w_4 w_5 w_6 a: a:_1 a:_2 a:_3 a:_4 a:_5 a:_6 a
```

```
a_1 a_2 a_3 a_4 a_5 a_6 @:  @:_1 @:_2 @:_3 @:_4 @:_5 @:_6 @  @_1 @_2 @_3
@_4 @_5 @_6 E  E_1 E_2 E_3 E_4 E_5 E_6 e  e_1 e_2 e_3 e_4 e_5 e_6 i  i_1
i_2 i_3 i_4 i_5 i_6 O  O_1 O_2 O_3 O_4 O_5 O_6 o  o_6 o_1 o_2 o_3 o_4 o_5
u  u_1 u_2 u_3 u_4 u_5 u_6 _1  _1_1 _1_2 _1_3 _1_4 _1_5 _1_6 a:I a:U aU @U
aI @I EU eU i@ iU Oa: Oa OE OI oI @:I u@ _1@ ue uI _1I u@: _1U ui: i@U oaI
oaI: u@I uI@ _1@I _1@U
plosive p b<  b<_1 b<_2 b<_3 b<_4 b<_5 b<_6 t th d<  d<_1 d<_2 d<_3 d<_4
d<_5 d<_6 c J<back>  J<back>_1 J<back>_2 J<back>_3 J<back>_4 J<back>_5
J<back>_6 k
central @:  @:_1 @:_2 @:_3 @:_4 @:_5 @:_6 @  @_1 @_2 @_3 @_4 @_5 @_6 _1
_1_1 _1_2 _1_3 _1_4 _1_5 _1_6
approximant l  l_1 l_2 l_3 l_4 l_5 l_6 r<back>  r<back>_1 r<back>_2
r<back>_3 r<back>_4 r<back>_5 r<back>_6 j  j_1 j_2 j_3 j_4 j_5 j_6 w  w_1
w_2 w_3 w_4 w_5 w_6
rounded O  O_1 O_2 O_3 O_4 O_5 O_6 o  o_6 o_1 o_2 o_3 o_4 o_5 u  u_1 u_2
u_3 u_4 u_5 u_6
unrounded a:  a:_1 a:_2 a:_3 a:_4 a:_5 a:_6 a  a_1 a_2 a_3 a_4 a_5 a_6 E
E_1 E_2 E_3 E_4 E_5 E_6 e  e_1 e_2 e_3 e_4 e_5 e_6 i  i_1 i_2 i_3 i_4 i_5
i_6 _1  _1_1 _1_2 _1_3 _1_4 _1_5 _1_6
vowel a:  a:_1 a:_2 a:_3 a:_4 a:_5 a:_6 a  a_1 a_2 a_3 a_4 a_5 a_6 @:
@:_1 @:_2 @:_3 @:_4 @:_5 @:_6 @  @_1 @_2 @_3 @_4 @_5 @_6 E  E_1 E_2 E_3
E_4 E_5 E_6 e  e_1 e_2 e_3 e_4 e_5 e_6 i  i_1 i_2 i_3 i_4 i_5 i_6 O  O_1
O_2 O_3 O_4 O_5 O_6 o  o_6 o_1 o_2 o_3 o_4 o_5 u  u_1 u_2 u_3 u_4 u_5 u_6
_1  _1_1 _1_2 _1_3 _1_4 _1_5 _1_6 a:I a:U aU @U aI @I EU eU i@ iU Oa: Oa
OE OI oI @:I u@ _1@ ue uI _1I u@: _1U ui: i@U oaI oaI: u@I uI@ _1@I _1@U
glottal      h
labiodental f v  v_1 v_2 v_3 v_4 v_5 v_6
affricate    ts<back> ts`
alveolar t d<  d<_1 d<_2 d<_3 d<_4 d<_5 d<_6 c ts<back> s z  z_1 z_2 z_3
z_4 z_5 z_6 n  n_1 n_2 n_3 n_4 n_5 n_6 J  J_1 J_2 J_3 J_4 J_5 J_6 r<back>
r<back>_1 r<back>_2 r<back>_3 r<back>_4 r<back>_5 r<back>_6
dental th s z  z_1 z_2 z_3 z_4 z_5 z_6 l  l_1 l_2 l_3 l_4 l_5 l_6
consonant p b<  b<_1 b<_2 b<_3 b<_4 b<_5 b<_6 t th d<  d<_1 d<_2 d<_3 d<_4
d<_5 d<_6 c J<back>  J<back>_1 J<back>_2 J<back>_3 J<back>_4 J<back>_5
J<back>_6 ts<back> ts` k f v  v_1 v_2 v_3 v_4 v_5 v_6 s z  z_1 z_2 z_3 z_4
z_5 z_6 s` x G  G_1 G_2 G_3 G_4 G_5 G_6 h m  m_1 m_2 m_3 m_4 m_5 m_6 n
n_1 n_2 n_3 n_4 n_5 n_6 N  N_1 N_2 N_3 N_4 N_5 N_6 J  J_1 J_2 J_3 J_4 J_5
J_6 l  l_1 l_2 l_3 l_4 l_5 l_6 r<back>  r<back>_1 r<back>_2 r<back>_3
r<back>_4 r<back>_5 r<back>_6 j  j_1 j_2 j_3 j_4 j_5 j_6 w  w_1 w_2 w_3
w_4 w_5 w_6
back O  O_1 O_2 O_3 O_4 O_5 O_6 o  o_6 o_1 o_2 o_3 o_4 o_5 u  u_1 u_2 u_3
u_4 u_5 u_6
fricative f v  v_1 v_2 v_3 v_4 v_5 v_6 s z  z_1 z_2 z_3 z_4 z_5 z_6 s` x G
G_1 G_2 G_3 G_4 G_5 G_6 h
unvoiced     p t th c ts<back> ts` k f s s` x h silsp
unaspirated  p t c ts<back> ts` k
```

12.2.3 Extracting Articulatory Characteristics

```
mlf = open('dicts/mlf_sub_wo_unk.mlf', 'r')
fout_place = open('Prepare_data_scripts/place_sub.txt', 'w')
fout_manner = open('Prepare_data_scripts/manner_sub.txt', 'w')
fout_vowel_cons = open('Prepare_data_scripts/vowel_cons_sub.txt', 'w')
fout_voice = open('Prepare_data_scripts/voice_sub.txt', 'w')
fout_additional = open('Prepare_data_scripts/additional_sub.txt', 'w')

place = ['none', 'front', 'central', 'back', 'bilabial', 'labiodental',
'dental', 'alveolar', 'palatal', 'velar', 'glottal']
manner = ['none', 'open', 'mid', 'close', 'plosive', 'nasal', 'affricate',
'fricative', 'approximant']
vowel_cons = ['none', 'vowel', 'consonant']
```

```
voice = ['voiced', 'unvoiced']
additional = ['none', 'rounded', 'unrounded', 'aspirated', 'glottalized',
'unaspirated']

for line in mlf: # Processing mlf file line by line

      line_list = list(line)

      if line_list[0] == '.': # End of file
            fout_place.write('\n')
            fout_manner.write('\n')
            fout_vowel_cons.write('\n')
            fout_voice.write('\n')
            fout_additional.write('\n')

      elif line_list[0] != '"' and line_list[0] != '#': # Processing
transcription itself

            split_string = list(line.split())

            phoneme = str(split_string[2])
            duration_fr = (int(split_string[1]) -
int(split_string[0]))//100000

            place_found = 0
            manner_found = 0
            vowel_cons_found = 0
            voice_found = 0
            additional_found = 0

            quest = open('dicts/Questions_cleaned.quest','r')

            for quest_line in quest:

                  cur_split = list(quest_line.split())

                  if cur_split[0] in place:
                        if phoneme in cur_split:
                              place_found = 1
                              for x in range(0, duration_fr):

      fout_place.write(str(place.index(cur_split[0])) + ' ')

                  elif cur_split[0] in manner:
                        if phoneme in cur_split:
                              manner_found = 1
                              for x in range(0, duration_fr):

      fout_manner.write(str(manner.index(cur_split[0])) + ' ')

                  elif cur_split[0] in vowel_cons:
                        if phoneme in cur_split:
                              vowel_cons_found = 1
                              for x in range(0, duration_fr):

      fout_vowel_cons.write(str(vowel_cons.index(cur_split[0])) + ' ')

                  elif cur_split[0] in voice:
                        if phoneme in cur_split:
                              voice_found = 1
                              for x in range(0, duration_fr):
```

```
            fout_voice.write(str(voice.index(cur_split[0])) + ' ')

                  elif cur_split[0] in additional:
                        if phoneme in cur_split:
                              additional_found = 1
                              for x in range(0, duration_fr):

            fout_additional.write(str(additional.index(cur_split[0])) + ' ')

                  else:
                        print('Unknown name of characteristic!')

            if place_found == 0:
                  for t in range(0, duration_fr):
                        fout_place.write(str(place.index('none')) + ' ')

            if manner_found == 0:
                  for t in range(0, duration_fr):
                        fout_manner.write(str(manner.index('none')) + ' ')

            if vowel_cons_found == 0:
                  for t in range(0, duration_fr):
                        fout_vowel_cons.write(str(vowel_cons.index('none'))
+ ' ')

            if voice_found == 0:
                  print('No voice information for phoneme ' + phoneme +
'!')

            if additional_found == 0:
                  for t in range(0, duration_fr):
                        fout_additional.write(str(additional.index('none'))
+ ' ')

            quest.close()
```

12.2.4 Extracting Phonemes and Tones

```
def read_dict(dict_file):
    """
    Reads list of phonemes from dict file
    """
    f = open(dict_file, 'r')
    phonemes = []
    for line in f:
        split_string = list(line.split())
        phonemes.append(split_string[0])

    return phonemes

f = open('dicts/mlf_sub_wo_unk.mlf', 'r')
fout_p = open('phn_matrix_sil_fixed.txt', 'w')
fout_t = open('Prepare_data_scripts/tone_matrix_sub.txt', 'w')
fout_l = open('Prepare_data_scripts/file_labels_sub.txt', 'w')

flag = 0
is_first_syllable = 1
sum_frames_syl = 0
tone = '0'
```

```
phonemes = read_dict('monophones_without_tones_cleaned.txt')

for line in f: # Processing mlf file line by line

    line_list = list(line)

    if flag == 0: # Beginning of new file

        if line_list[0] == '"': # Processing file header

            flag = 1

            fout_l.write(''.join (line_list[3:line_list.index('.')]) +
'\n') # Write current file name to the output

    else: # Continue processing file

        if line_list[0] == '.': # End of file

            # Write info about the last syllable of the previous file
            for x in range(0, sum_frames_syl):
                fout_t.write(str(tone) + ' ')

            tone = '0'
            sum_frames_syl = 0
            flag = 0
            is_first_syllable = 1

            fout_p.write('\n')
            fout_t.write('\n')

        else: # Processing transcription itself
            split_string = list(line.split())

            if len(split_string) > 6: # Syllable beginning
                # Write to files info about the previous syllable
                for x in range(0, sum_frames_syl):
                    fout_t.write(str(tone) + ' ')
                tone = '0'
                sum_frames_syl = 0

            if len(str(split_string[2])) > 2 and str(split_string[2])[-2]
== '_': # Has tone
                tone = str(split_string[2])[-1]
                phoneme = str(split_string[2])[0:-2]
            else: # No tone
                phoneme = str(split_string[2])

            if phoneme == 'silsp':#silence! Finish the syllable
                for x in range(0, sum_frames_syl):
                    fout_t.write(str(tone) + ' ')
                tone = '0'
                sum_frames_syl = 0

            duration_fr = (int(split_string[1]) -
int(split_string[0]))//100000 # Phoneme duration
            sum_frames_syl = sum_frames_syl + duration_fr # Add to current
syllable duration

            # Write phoneme indices
```

```
index = phonemes.index(phoneme)
for x in range(0, duration_fr):
    fout_p.write(str(index) + ' ')
```

12.3 Matlab Appendix

```
fid1 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/phonemes_with_tones_no_unk_matrix.txt');
fid2 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/tone_matrix_sub.txt');
fid3 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/place_sub.txt');
fid4 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/manner_sub.txt');
fid5 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/vowel_cons_sub.txt');
fid6 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/voice_sub.txt');
fid7 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/additional_sub.txt');
fid8 =
fopen('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basic
/Prepare_data_scripts/file_labels_sub.txt');

line_phon = fgets(fid1);
line_tone = fgets(fid2);
line_place = fgets(fid3);
line_manner = fgets(fid4);
line_vowel_cons = fgets(fid5);
line_voice = fgets(fid6);
line_additional = fgets(fid7);

while ischar(line_phon)
    phon_array = str2num(line_phon);
    [m,n] = size(phon_array); % m = 1, n = number of frames
    phon_matrix = zeros(n, 91); % <- Set number of phonemes!
    for i = 1:n
        phon_matrix(i, phon_array(i)+1) = 1;
    end

    tone_array = str2num(line_tone);
    tone_matrix = zeros(n, 7); % <- Set number of secondary
characteristics (tone)
    for i = 1:n
        tone_matrix(i, tone_array(i)+1) = 1;
    end

    place_array = str2num(line_place);
    place_matrix = zeros(n, 11); % <- Set number of secondary
characteristics (place)
    for i = 1:n
        place_matrix(i, place_array(i)+1) = 1;
    end

    manner_array = str2num(line_manner);
    manner_matrix = zeros(n, 9); % <- Set number of secondary
characteristics (manner)
    for i = 1:n
```

```
        manner_matrix(i, manner_array(i)+1) = 1;
    end

    vowel_cons_array = str2num(line_vowel_cons);
    vowel_cons_matrix = zeros(n, 3); % <- Set number of secondary
characteristics (vowel/cons)
    for i = 1:n
        vowel_cons_matrix(i, vowel_cons_array(i)+1) = 1;
    end

    voice_array = str2num(line_voice);
    voice_matrix = zeros(n, 2); % <- Set number of secondary
characteristics (voice)
    for i = 1:n
        voice_matrix(i, voice_array(i)+1) = 1;
    end

    additional_array = str2num(line_additional);
    additional_matrix = zeros(n, 6); % <- Set number of secondary
characteristics (additional)
    for i = 1:n
        additional_matrix(i, additional_array(i)+1) = 1;
    end

    res_matrix = [phon_matrix tone_matrix place_matrix manner_matrix
vowel_cons_matrix voice_matrix additional_matrix];

    file_label  = fgets(fid8);
    file_name =
strcat('/mnt/matylda6/xegoro00/nn_exp/trunk/examples/BABEL_Vietnamese_basi
c/generated_targets_articulatory_sub/', file_label, '.tgt');

    writehtk(file_name,res_matrix,0.01,12);

    line_phon = fgets(fid1);
    line_tone = fgets(fid2);
    line_place = fgets(fid3);
    line_manner = fgets(fid4);
    line_vowel_cons = fgets(fid5);
    line_voice = fgets(fid6);
    line_additional = fgets(fid7);
end

fclose(fid1);
fclose(fid2);
fclose(fid3);
fclose(fid4);
fclose(fid5);
fclose(fid6);
fclose(fid7);
fclose(fid8);
```

12.4 Contents of the DVD

1. PDF of the thesis
2. DOC of the thesis
3. Modified TNet tool
4. Python scripts folder
5. MatLab scripts folder