# The project Kaldi

## Open source speech recognition

Karel Vesely

Speech@FIT, BUT

ZRE, Brno, 17.4.2019

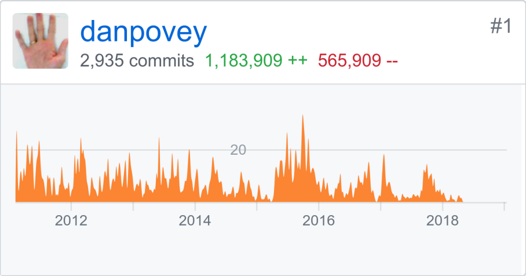
# What is Kaldi?

- Wiki: A legendary Ethiopian goatherd who found coffee seeds after seeing the ‘energetic jumping goats’ eating it.
- Github: Open-source toolkit for building **speech recognition** systems.

# A bit of history...

- 2009: Initiated in Summer workshop at Johns Hopkins University (Baltimore, USA)
  - a toolkit was needed for new acoustic model (SGMM)
  - colleagues from Brno were there
- 2010: Dan Povey started coding Kaldi at Microsoft
- **2010, 2011, 2012, 2013: Kaldi development workshops**
  - **organised in Brno at FIT**
  - **international team of self-funded volunteers**
    **(USA, Canada, China, India, Germany, Czech Republic, ...)**
- 2011: Kaldi toolkit presented at conferences
  ICASSP (Prague), ASRU (Hawaii)
- 2012: Dan Povey joins JHU in Baltimore (leaving Microsoft)
- 2015: Kaldi moves from SourceForge to **GitHub**

# Who is 'Daniel Povey'?




- main architect of Kaldi
  - also mathematician, programmer, help-support
- and supervisor of PhD students from JHU

# What is Kaldi? II.

Kaldi = GitHub project[1], it consists of:

- Set of command-line **programs for training and representing speech recognition models** (C++).
- **example recipes** = set of "**standard experiments**" on cluster computer (BASH, perl, awk, SGE cluster)
- **Documentation**[2]**:** Doxygen with tutorial, topic-based pages and C++ code reference
- **Support** (forum, issue tracking)

[1] https://github.com/kaldi-asr/kaldi

[2] http://kaldi-asr.org/doc/

# What is Kaldi? III.

Github traffic stats from last 14-days
(the blue curves are unique 'cloners' and 'visitors'),

# The example recipes = main strength of Kaldi

The recipes are main strength of Kaldi compared to other toolkits! (HTK, Sphinx, Julius, ...)

- Toy examples: yes/no, tidigits,
- Free-databases: AMI meetings (80h), TED-LIUM talks (120h), librispeech, voxforge, vystadial_cz
- The standard tasks (from easy to difficult):
  - **Read speech:** Wall Street Journal (80h, **WER[3]=2.5%**),
  - **Conversational telephone speech:** Switchboard (300h, **WER=9%**),
  - **Spontaneous 'distant microphone-array' speech:** AMI meetings (80h captured by 8 mic-array **WER=32%**, with 'close-talk mic' we get **WER=19%**)

[3] WER = word error rate

# Why is Kaldi good for research?

- **Experiments are reproducible:**
  (researchers work with same baseline systems)
- no need to implement everything from scratch
- some data formats can be loaded in Python
- It is a **community project**:
  - changes get-in by pull-requests
  - 2.7k forks
- **License:** Apache v2.0, a very liberal legal framework: allows modifications and commercial use.

# Speech recognition research ecosystem

**Researchers:**
- are using the toolkit
- some are contributors

KALDI

**Big companies:**
- some use Kaldi
- all have access to the code

**Start-ups:**
- getting free ASR technology
- creating new ASR applications

Big companies doing speech research: Nuance, IBM, Google, Microsoft, Apple, Amazon, Baidu, Telefonica, Samsung.
Many have open work positions...

# Implemented techniques

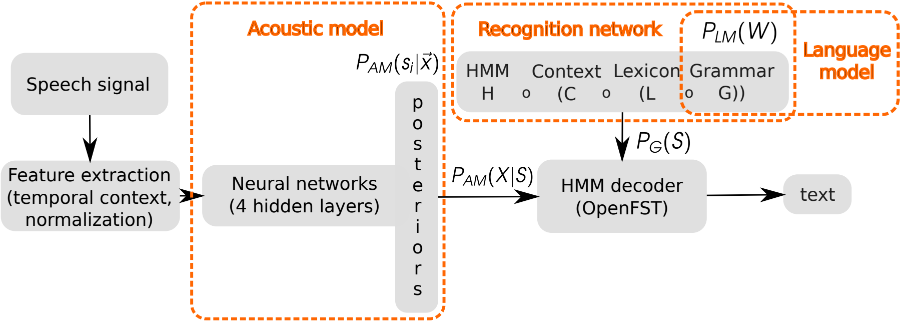
Speech recognition:

- HMM decoder using WFST transducers

- Acoustic models: GMM, DNNs,
- Language models: N-GRAM, RNNLM

- speaker adaptation techniques (iVector based)
- sequence-discriminative training sMBR, LF-MMI (global optimization instead of 'per-frame' training)

# Speech recognition: A hybrid approach

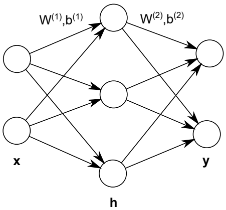



The decoding formula:

$$\hat{W} = \mathrm{wrds}\left(\underset{S}{\mathrm{argmax}}\, P_{AM}(\mathbf{X}|S)^{\kappa}\; P_G(S)^{\rho}\right)$$

# Acoustic model: Neural network

Example: feed-forward neural network with one hidden layer,




$\mathbf{x}$ input vector
$\mathbf{h}$ hidden-layer vector
$\mathbf{y}$ output vector

$\mathbf{W^{(1)}}, \mathbf{W^{(2)}}$ matrices of trainable weights
$\mathbf{b^{(1)}}, \mathbf{b^{(2)}}$ vectors of trainable biases

Sigmoid,

$$h_i^{(1)} = \sigma(a_i^{(1)}) = \frac{1}{1 + \exp(-a_i^{(1)})}$$

Softmax,

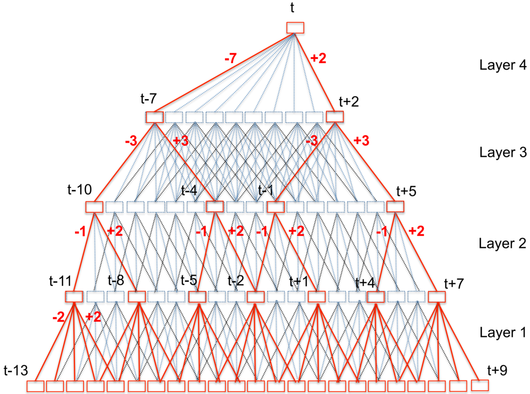
$$y_i = \frac{\exp(a_i^{(2)})}{\sum_j \exp(a_j^{(2)})}, \quad \sum_i y_i = 1$$

Forward pass,

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

# Acoustic model: Time-delay Neural network (TDNN)

- Inspired by our work on Stacked Bottleneck networks were created TDNNs.
- The weights in a 'Layer' are shared by each 'red' subtree (convolutional networks do the same)
- 'Big-picture' is gradually assembled from short-term inputs.
- Feed-forward network (fast, easy to train)

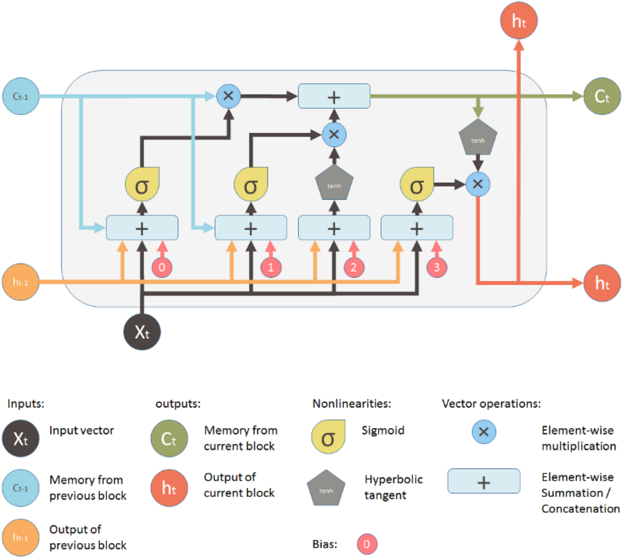


# Acoustic model: Long-short term memory cell (LSTM)

TDNN layers can have LSTM layers in between.

LSTM is a recurrent layer (=output depends on previous inputs via internal state)

LSTM has a memory cell $c_t$ and Sigmoid gates (forgetting, input, output)



# Acoustic model: Training the Neural Network

Supervised training of a classifier

- acoustic model is trained from transcribed speech,
- traditionally, classes are HMM states of phonemes in some context (biphone, triphone),
- but classes can be also whole-words, syllables or even graphemes (end-to-end systems),

Training algorithm: **mini-batch Stochastic Gradient Descent**, (noisy parameter updates according to local gradients with a 'good' global trend):

$$\vec{w}_{t+1} = \vec{w}_t - \eta \nabla E(\vec{w}_t)$$

When training, we need to somehow 'time-align' the transcriptions with the speech signal using an existing model.

# Acoustic model: Training the Neural Network

The current state-of-the art loss function in Kaldi is: **Lattice-free MMI** (sequence-discriminative training).

It is inspired in CTC (Connectionist Temporal Classification) and MMI (Maximum Mutal Information training).

As in CTC, it maximizes posterior of the correct transcript in an utterance and **updates the 'time-alignment' on-the-fly**.

As in MMI training there is a positive signals (**numerator** posteriors) and a negative signal (**denominator** posteriors).

The *numerator* is generated from transcription. The *denominator* are from 'alternative hypotheses' generated with the current acoustic model on-the-fly on a GPU.

# Where we use kaldi

- in research, for publishing results in conference articles,
- for building systems for funded research projects,

# What can you do with Kaldi

- Play with the toy examples:
  `yesno`, `voxforge`, `vystadial_cz`
- Think of a creative ‘speech-based’ application, the pre-built models are available:
  `http://kaldi-asr.org/downloads/all/`.

# Useful links

GitHub project:

- **https://github.com/kaldi-asr/kaldi**

Documentation:

- **http://kaldi-asr.org/doc/**

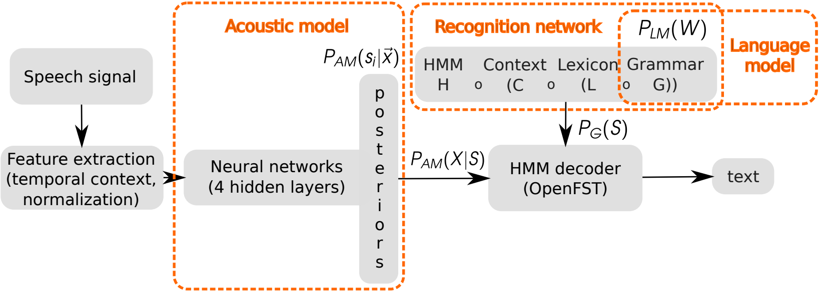
Support forum:

- https://groups.google.com/forum/#!forum/kaldi-help

Other resources:

- http://www.danielpovey.com/kaldi-lectures.html
- http://www.danielpovey.com/publications.html
- http://www.danielpovey.com/
- http://kaldi-asr.org

# The DEMO, I.




Feature extraction:

- `compute-fbank-feats`
- `compute-pitch-feats`
- `paste-feats`, `apply-cmvn`

Acoustic model evaluation: `nnet-forward`
HMM decoder: `decode-faster-mapped`
Showing the output: `utils\int2sym.pl`

# The DEMO, II.

Show the script...

- Lexicon with 579k ‘words’,
- HCLG network has 1.4GBs (after LM pruning),
- Acoustic model has 9.7 million trainable parameters
  (feed-forward neural network with 4 hidden layers and 5862 outputs),
- On-line cepstral mean normalization,
- Acoustic-model + HMM-decoder are background processes (communicating via ‘named pipes’),

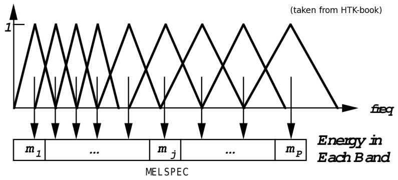
# The DEMO, III., FBANK features

FBANK features = a smooth spectrogram,

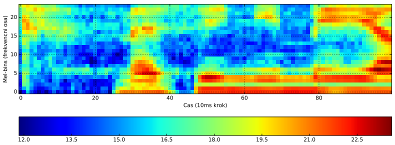
- 10ms time-steps, non-uniform steps in frequency

  (but uniform on Mel-scale, according to which we hear),
- $\log$ of the ‘power’ at particular frequency as integrated with the triangular Mel-filters,
- we splice 21 FBANK frames to form the DNN input

  (i.e. we take a window over 21 time-steps),




Bank of Mel-filters

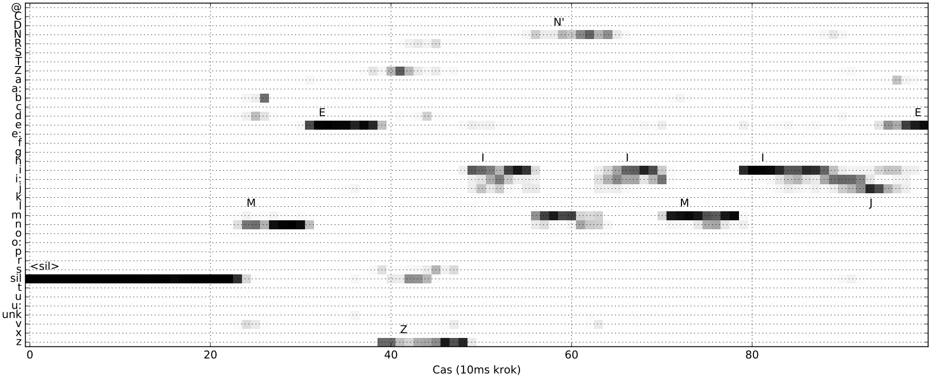



FBANK features

# The DEMO, IV., NN posteriors

How does the Neural Network output look like?
(posterior probabilities)

For illustration we summed the 5859 outputs into 36+2 phonemes:



# The end

Thank you!