Sound event detection with neural networks

Moritz Augustin

NI Group Meeting, 13th July 2018

Introduction

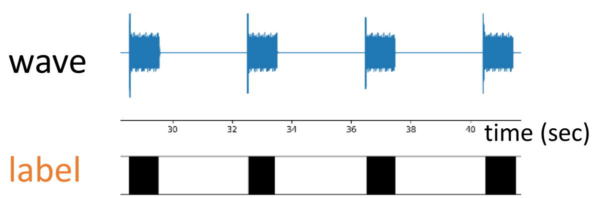
- Two!Ears EU project
  - Sound event database: *NIGENS* anechoic earsignals
    Neural Information Processing Group General Sounds Database Earsignals
  - Binaural simulator software: scene mixtures varying noise & sources
  - System expertise: Ivo
- Methods relied on
  - Linear feedforward models: logistic regression
    Trowitzsch, Mohr, Kashef, Obermayer 2017, IEEE Audio Speech Language Process.
  - Engineered features: temporal information via derivative statistics (500ms blocks)
- Extension in progress: nonlinear & temporal models
  - Deep neural networks
    Heiner Spieß, NI project
    - Recurrent networks: long short-term memory
      Changbin Lu, Master thesis
    - Feedforward (convolutional) neural networks
      Alessandro Schneider, Bachelor thesis
  - Directly applicable to features with fine temporal resolution
  - Representations learnt via supervised training
  - Hypothesis: improved generalization performance over baseline model

Data: Isolated Sounds

Sounds from combined training+devel set only:

NIGENS:
human-labeled sounds (on/offsets) from 13 classes

Example: a phone sound (12sec)



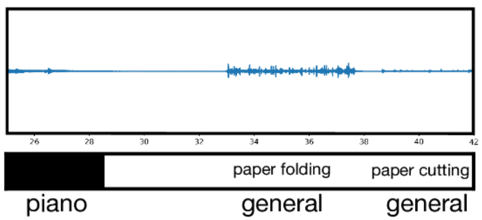
Class	Waves (count)	min-max (sec)	Total time (min)
Alarm	49	0.9 – 64.6	16
Baby	40	1.7 - 123.4	18
Crash	50	1.7 – 48.3	8
Dog	45	0.2 – 49.4	9
Engine	39	4.0 - 132.7	35
Scream	76	0.5 – 94.5	6
FemaleSpeech	100	1.2 – 5.1	5
Fire	51	2.4 – 162.1	45
Footsteps	42	3.0 – 33.0	19
Knock	40	0.6 – 14.4	2
MaleSpeech	100	1.5 – 5.0	4
Phone	40	1.0 - 65.0	12
Piano	42	2.3 – 196.0	15
Total	**714**	0.2 – 196.0	194
General class	303	0.2 – 180.6	92

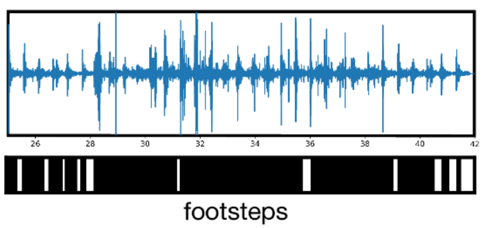
Trowitzsch, Taghia, Kashef, Obermayer 2016,
Zenodo: https://zenodo.org/record/168042

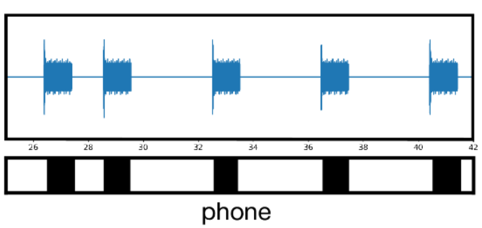
Data: Scenes

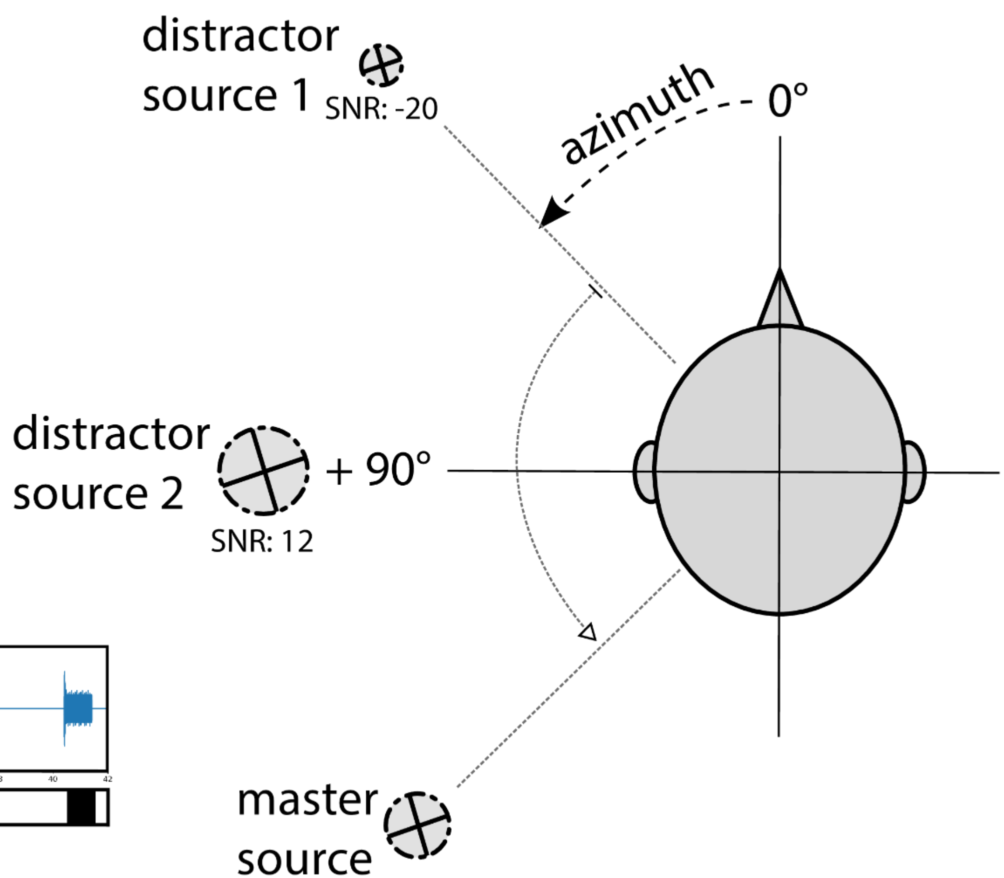
- Mixtures: binaurally recorded simulated scene instance (min: 30sec)
- Master: **one** sound (of non-general class), e.g. Phone (<30sec: repeated)
- 0-3 distractors: random sounds (incl. general & master class)
- **Scene**: Fixed values of scene parameter (#src, azimuth, SNR)
- **Scene instance**: Scene with a specific sound mixture

Correction (visualization wrong):
ds1 amp > master > ds2 amp

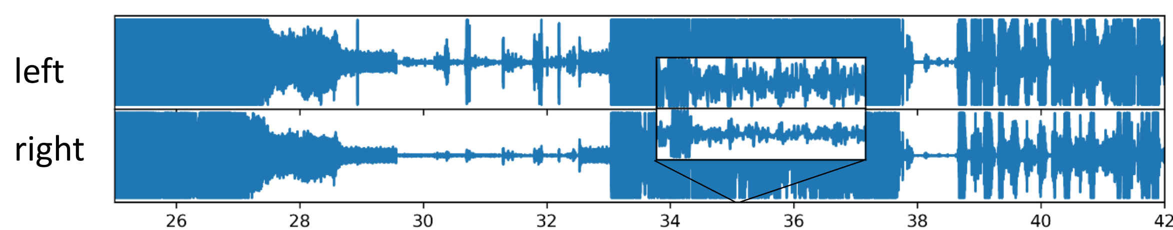






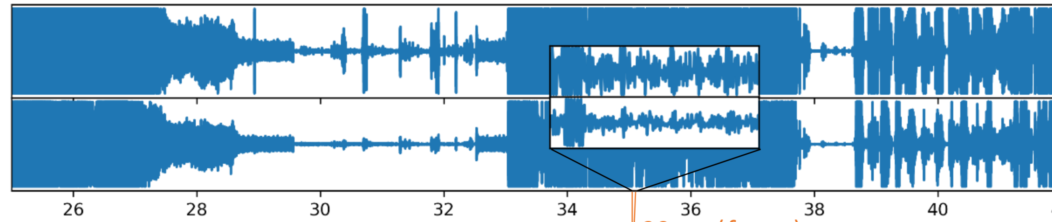


Resulting scene instance (stereo mixture)



Data: Features

Mixture
(stereo)



20 ms (*frame*)

Left/right ratemap avgeraged

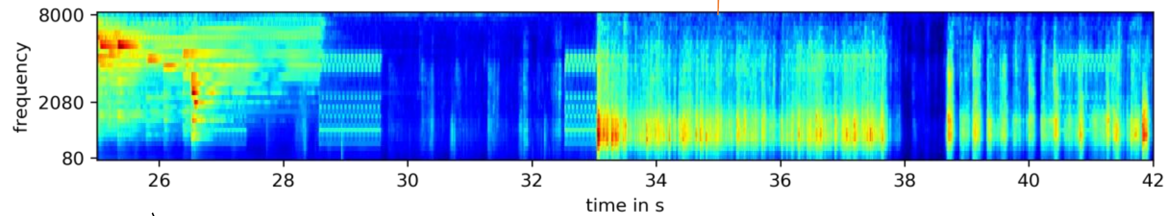
Left/right AMS averaged

=> concatenated & flattened:

$\mathbf{x}_t \in \mathbb{R}^{160}$

Ratemap: 32 dim.

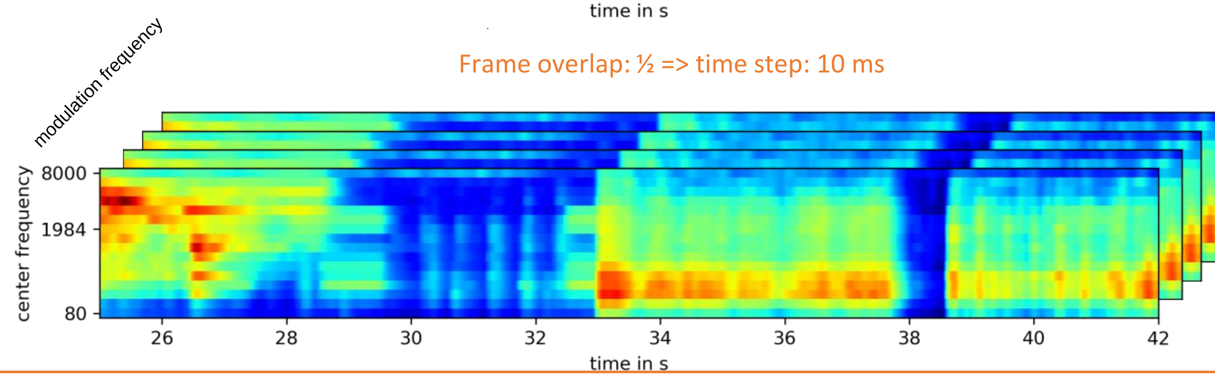
~biologically inspired Fourier transform



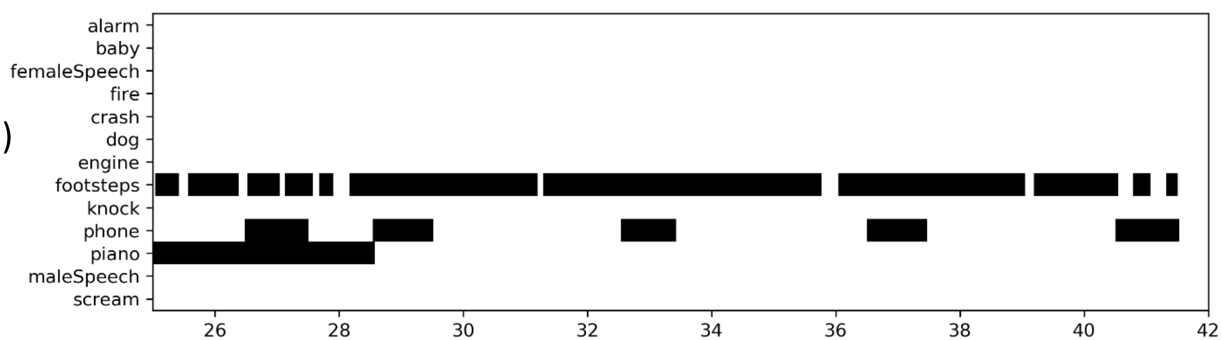
Amplitude modulation spectra (AMS): 16x8 dim.

~Fourier transform of signal envelope

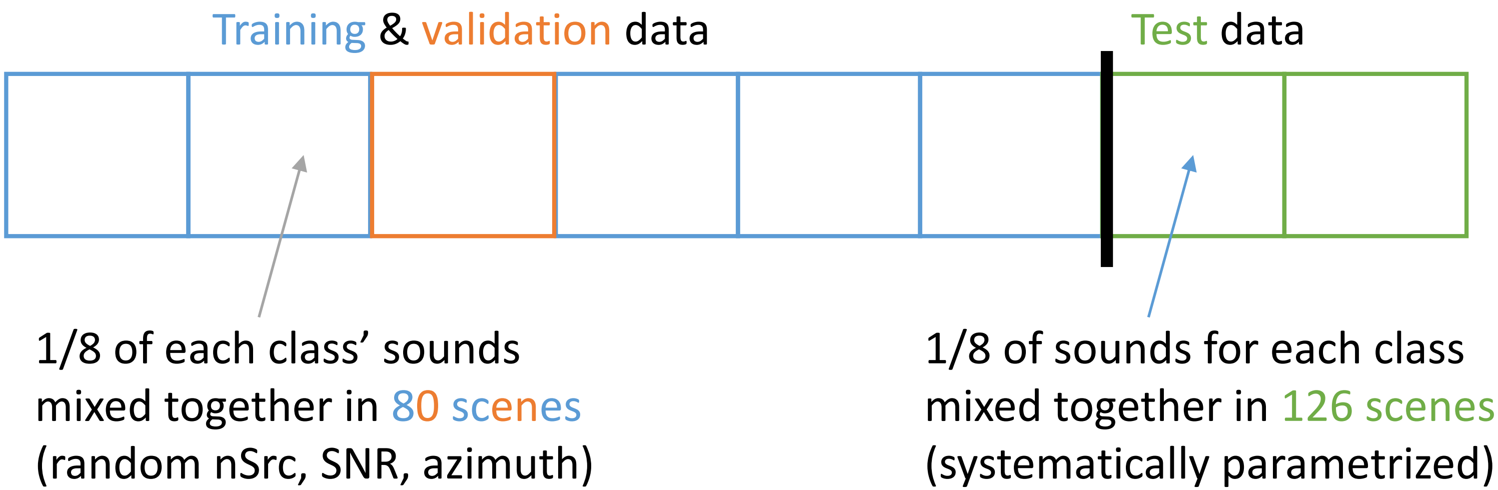
Frame overlap: ½ => time step: 10 ms



Labels: 13 dim.
(binary, multilabel)



Data: Validation

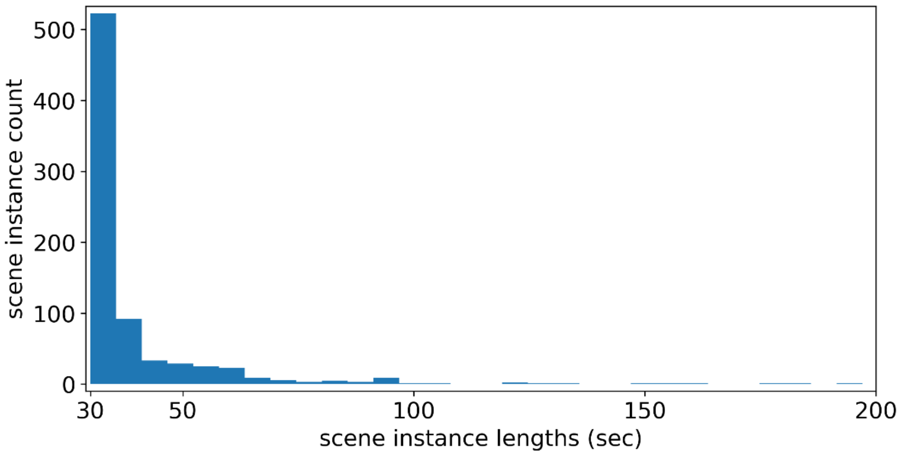


Multiconditional training as demonstrated: Trowitzsch, Mohr, Kashef, Obermayer 2017, IEEE Audio Speech Language Process.

Data: Heterogenities

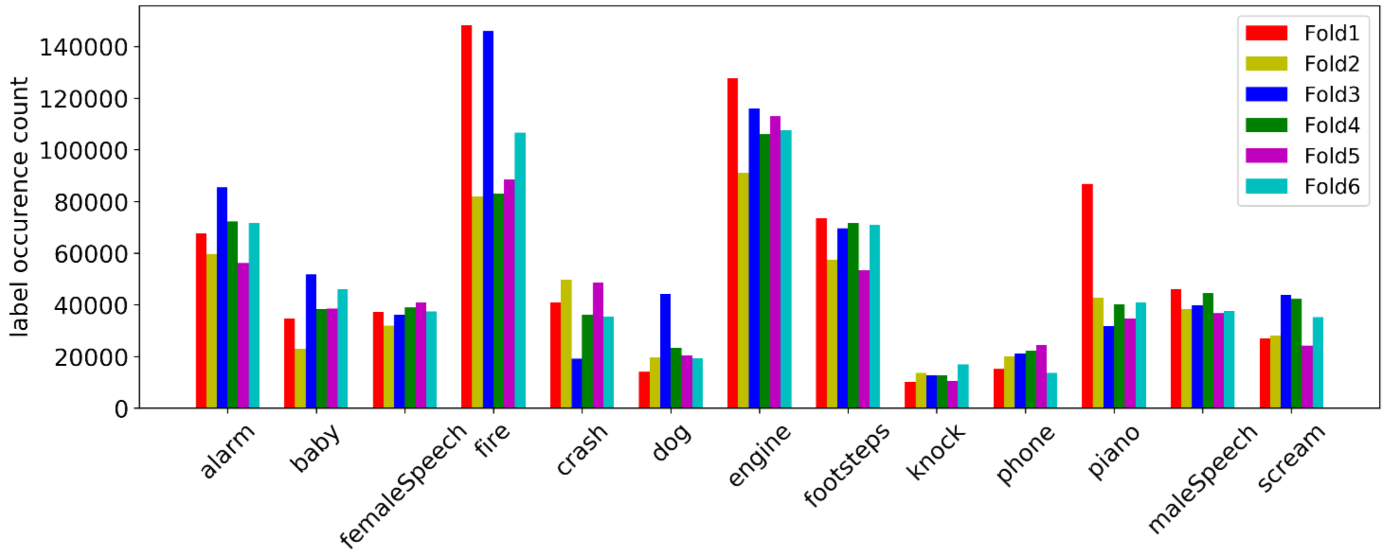
Scene instance distribution highly skewed
- long sounds, e.g., fire/piano
- short sounds, e.g.,
  knock, (fe)maleSpeech
- min. mixture length 30 sec

=> choice of input history length

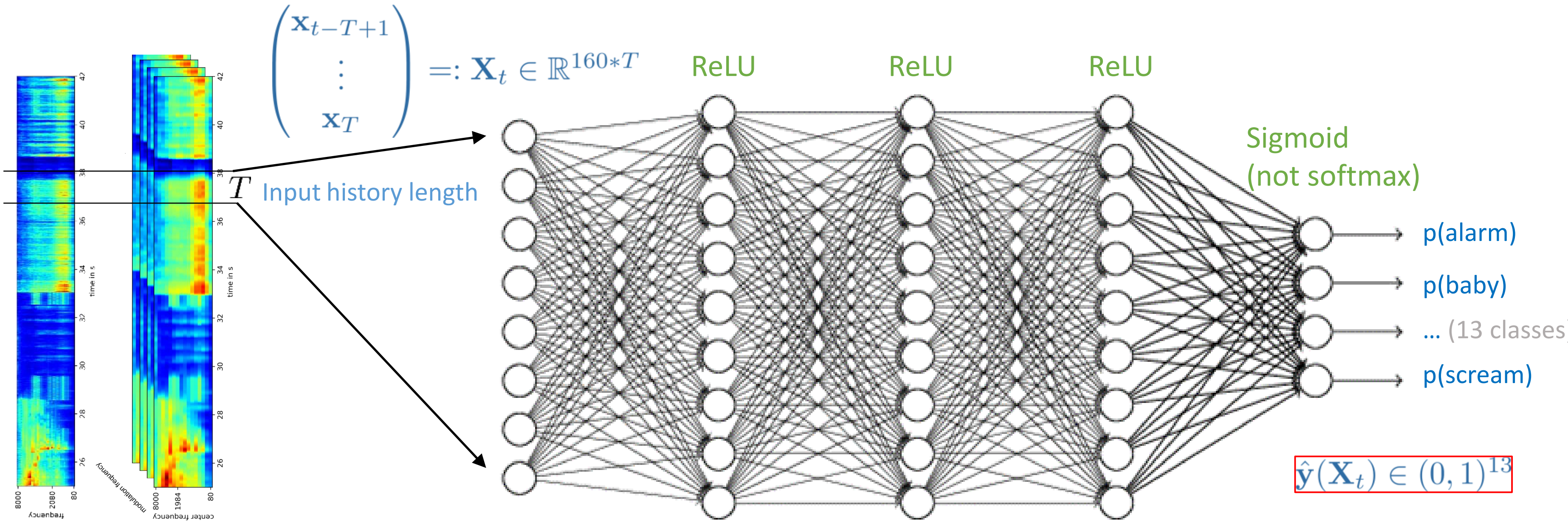


Label distribution variations
- across folds
- across classes (1-vs-all)

=> balanced training



Model 1: Multilayer Perceptron



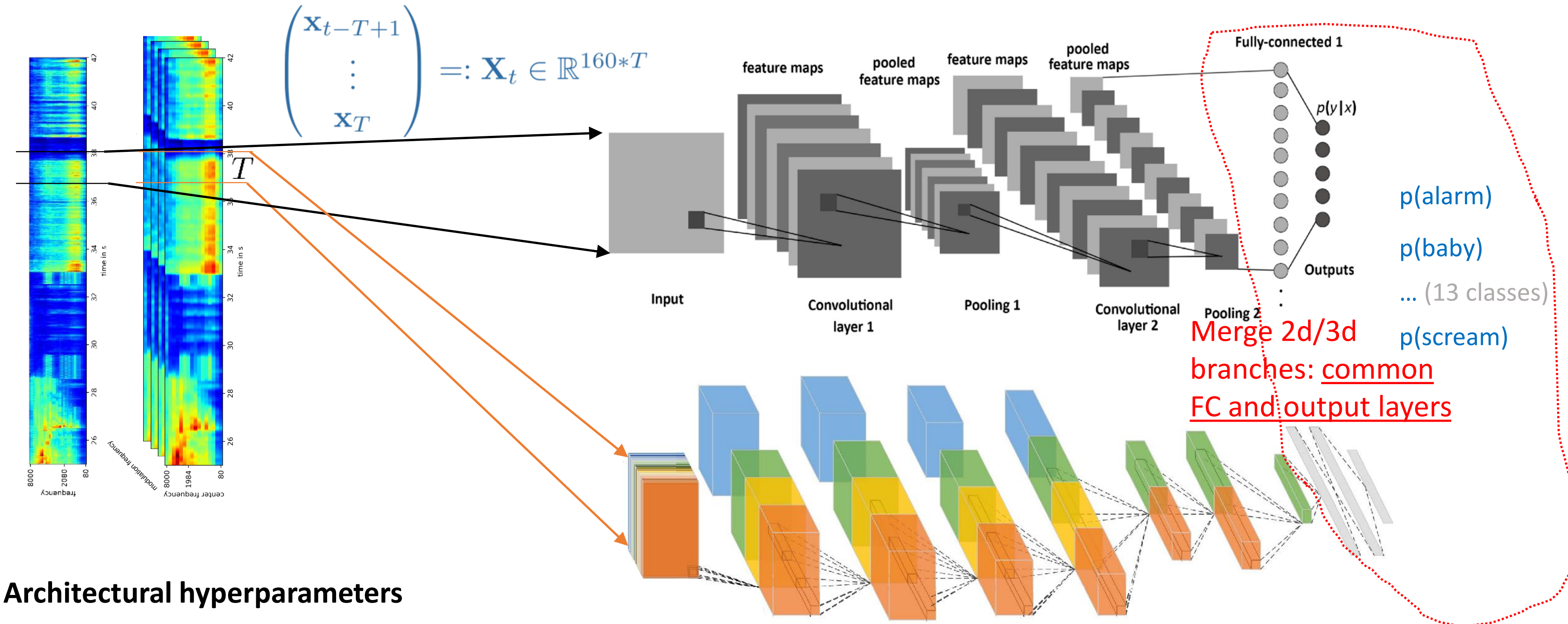
Uniform weights Initialization, biases 0
Glorot & Bengio 2010

Dropout regularization
(see later slide)
Hinton et al 2012
Srivastava et al 2015

Architectural hyperparameters

- Input history length T (500ms, 5000ms, longer?) #layers (3, 4, 5, 6)
- #neurons_per_layer (50, 100, 200, 400)
- #dropout_rate (25%, 50%, 75%, 90%; same or increasing with depth)
- data_stride (fix: stride 167ms = 500ms/3, subsampling resolution independent of T)

Model 2: Convolutional neural net



Architectural hyperparameters

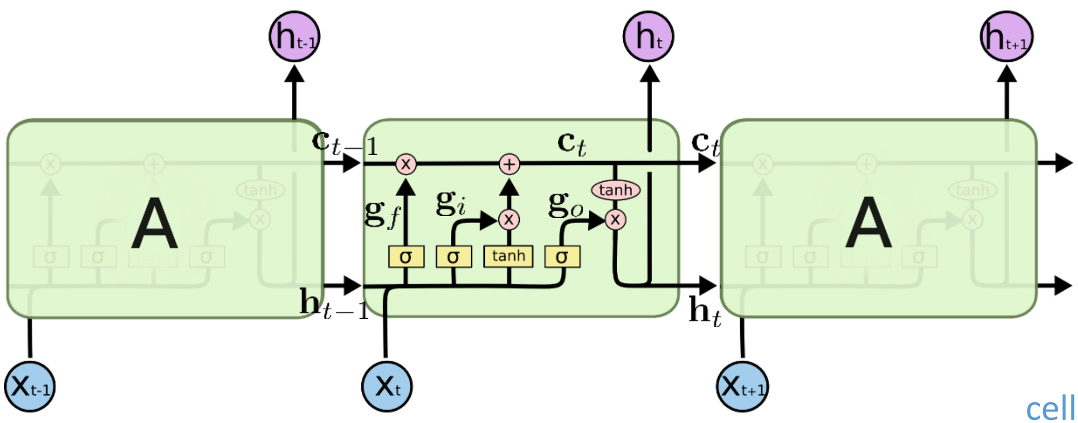
- Input history length T (500ms, 5000ms, longer?)
- #convolutional_layers (fix: 3)
- #featuremaps_per_convlayer (increasing with depth)
- #convolution_winsizes (fix: 3x3 / 3x3x3 with stride 1)
- #pooling_winsizes (fix to 2 with stride 2)
- #fullyconnected_layers (2, 3, 4, 5)
- #neurons_per_fc_layer (fix: 100)
- #dropout_rate (increasing with depth)
- data_stride (fix: stride 167ms = 500ms/3, subsampling resolution independent of T)

Methodological choices

- Max-pooling
- ReLU transfer
- Sigmoid output
- Batch normalization
- Dropout regularization

Model 3: Long short-term memory

- Simple RNN cannot learn long-term relationships (vanishing gradient)
- Solution: LSTM (here: no peepholes) or Gated Recurrent Units (GRU)



Transformation per time step

$$(\mathbf{h}_t,\, \mathbf{c}_t) = \phi(\mathbf{x}_t,\, \mathbf{h}_{t-1},\, \mathbf{c}_{t-1})$$

$$\mathbf{h}_t \in \mathbb{R}^{N_{\text{cells}}} \qquad \mathbf{x}_t \in \mathbb{R}^{N_{\text{input}}}$$

$$\mathbf{c}_t \in \mathbb{R}^{N_{\text{cells}}}$$

Vanilla LSTM:

cell state $\mathbf{c}_t = \mathbf{g}_f \odot \mathbf{c}_{t-1} + \mathbf{g}_i \odot \tilde{\mathbf{c}}_t$

cell state candidate $\tilde{\mathbf{c}}_t = \mathbf{tanh}\left(\mathbf{W}_c\, \mathbf{x}_t + \mathbf{R}_c\, \mathbf{h}_{t-1} + \mathbf{b}_c\right)$

forget gate $\mathbf{g}_f = \boldsymbol{\sigma}\left(\mathbf{W}_f\, \mathbf{x}_t + \mathbf{R}_f\, \mathbf{h}_{t-1} + \mathbf{b}_f\right)$

input gate $\mathbf{g}_i = \boldsymbol{\sigma}\left(\mathbf{W}_i\, \mathbf{x}_t + \mathbf{R}_i\, \mathbf{h}_{t-1} + \mathbf{b}_i\right)$

output gate $\mathbf{g}_o = \boldsymbol{\sigma}\left(\mathbf{W}_o\, \mathbf{x}_t + \mathbf{R}_o\, \mathbf{h}_{t-1} + \mathbf{b}_o\right)$

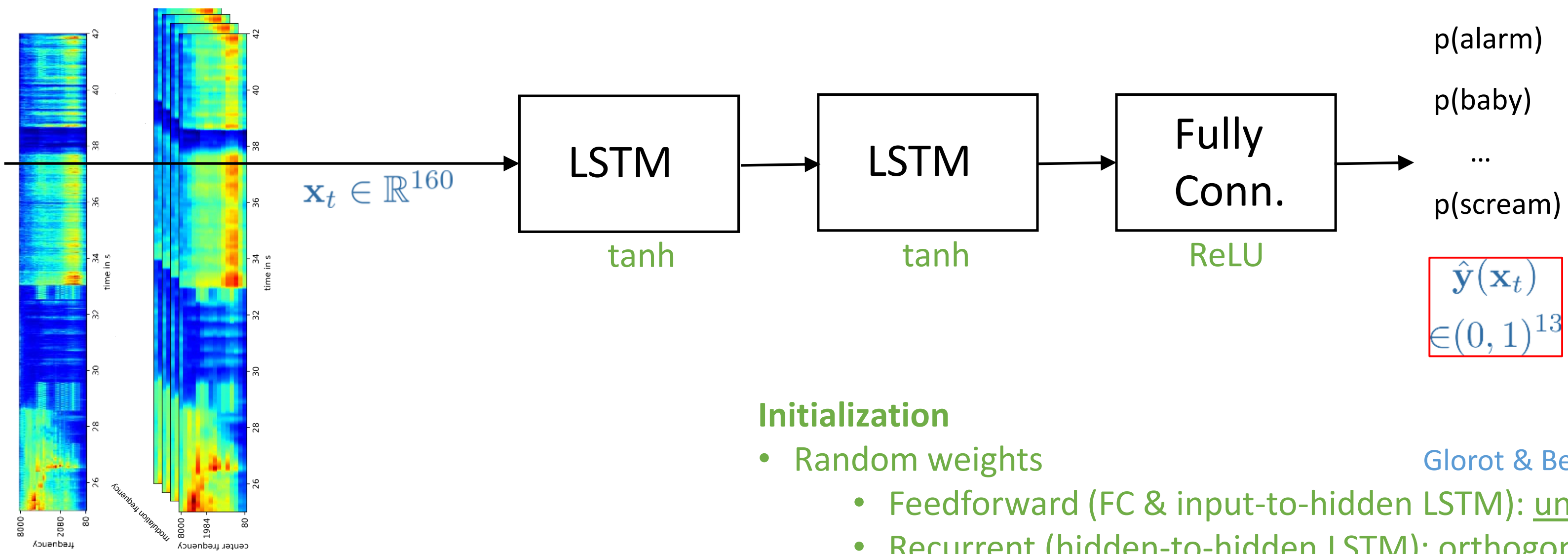
layer output $\mathbf{h}_t = \mathbf{g}_o \odot \mathbf{tanh}\left(\mathbf{c}_t\right)$
(=recurrent input for next time step)

Learning:
Backpropagation through time
Forward pass: O(T)
Backward pass: O(T)

=> effectively: very deep feedforward network (T layers)

T: backpropagation length

Model 3: Long short-term memory



Initialization

- Random weights — Glorot & Bengio 2010
  - Feedforward (FC & input-to-hidden LSTM): uniform
  - Recurrent (hidden-to-hidden LSTM): orthogonal — Saxe et al 2013
- Biases 0 (except forget gate: bias 1) — Gers et al 1999

Statefulness

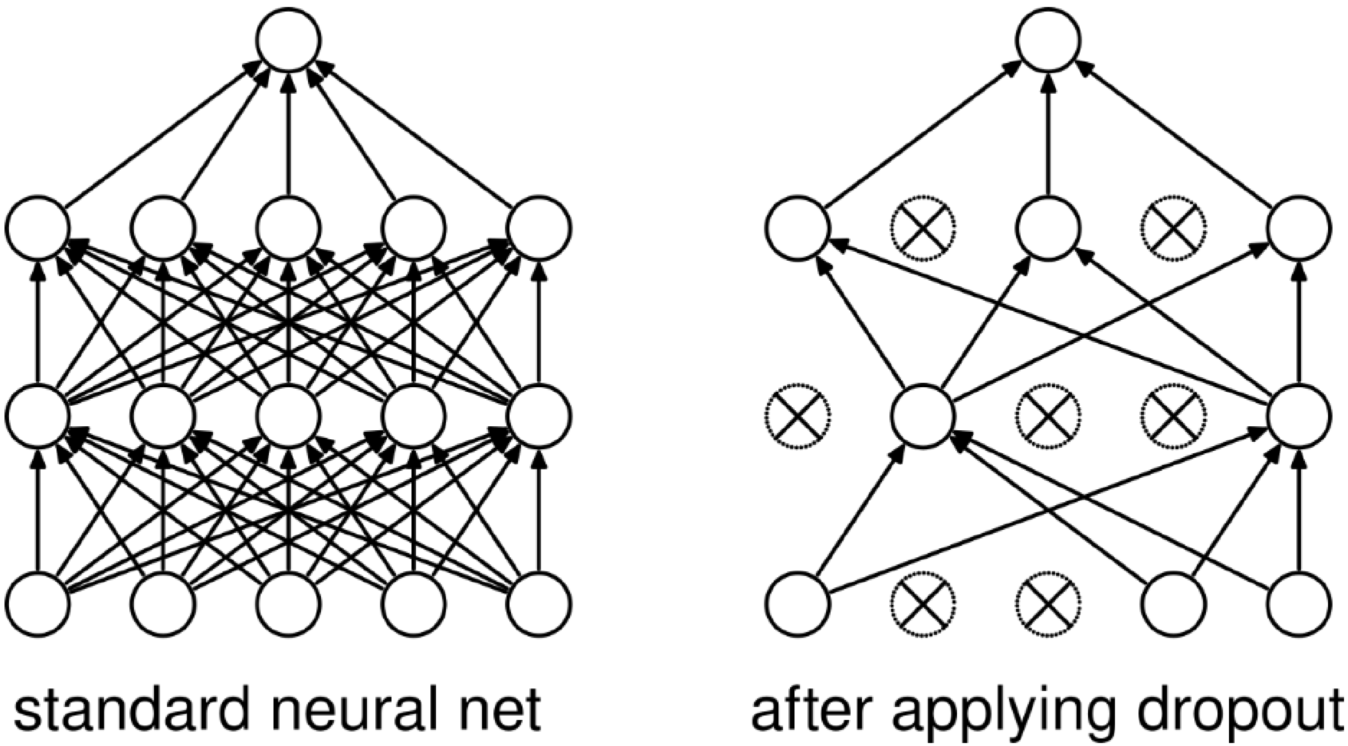
- cell state and output: saved and resumed between consecutive batches

Architectural hyperparameters

- #LSTM_layers (2, 3, 4)
- #FC_layers (fix: 2)
- #neurons_per_LSTM_layer (50, 100, 200, 400)
- #neurons_per_FC_layer (50, 100, 200, 400)
- #dropout_rate_LSTM (25%, 50%, 75%, 90%; same or increasing with depth)
- #dropout_rate_FC (25%, 50%, 75%, 90%; same or increasing with depth)
- Input time history T for truncated backpropagation

Dropout regularization

- Effective regularization by randomly dropping neurons
- Trained model corresponds to averaging the ensemble ($2^{N_{\text{neurons}}}$ large)



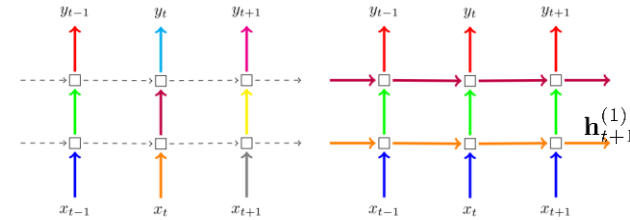
Hinton et al 2012

Srivastava et al 2015

MLP/Convnet: dropout directly applicable

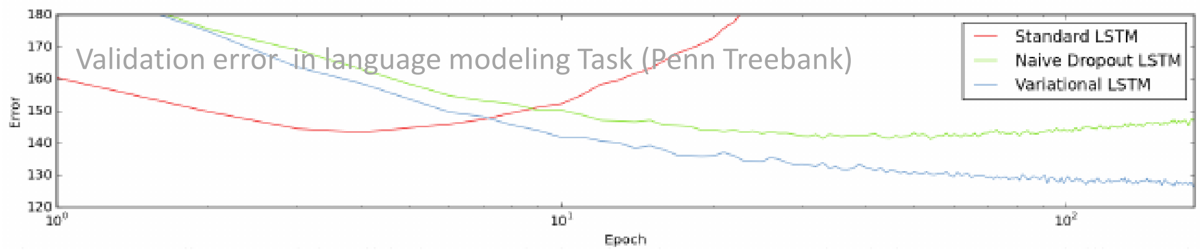
LSTM: more challenging because of temporally dependent states

Recurrent Dropout



Recently: recurrent dropout technique developed Gal & Ghahramani 2015

- Interpretation: variational inference for posterior weight distribution
- Implemented in deep learning frameworks but not in CuDNN



proof

$$[\mathbf{R}_* \{\mathbf{m} \odot \mathbf{h}_{t-1}\}]_i = \sum_{j=1}^{N_{\text{cells}}} (\mathbf{R}_*)_{ij} \{(\mathbf{m})_j (\mathbf{h}_{t-1})_j\}$$

$$= \sum_{j=1}^{N_{\text{cells}}} \{(\mathbf{R}_*)_{ij} \{(\mathbf{m})_j\} (\mathbf{h}_{t-1})_j = [\{\mathbf{R}_* \odot \mathbf{M}\} \mathbf{h}_{t-1}]_i$$

LSTM with *variational* Dropout

$$\mathbf{c}_t = \mathbf{g}_f \odot \mathbf{c}_{t-1} + \mathbf{g}_i \odot \widetilde{\mathbf{c}}_t$$ cell state

$$\widetilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{R}_c (\mathbf{m} \odot \mathbf{h}_{t-1}) + \mathbf{b}_c)$$ cell state candidate

$$\mathbf{g}_f = \boldsymbol{\sigma}(\mathbf{W}_f \mathbf{x}_t + \mathbf{R}_f (\mathbf{m} \odot \mathbf{h}_{t-1}) + \mathbf{b}_f)$$ forget gate

$$\mathbf{g}_i = \boldsymbol{\sigma}(\mathbf{W}_i \mathbf{x}_t + \mathbf{R}_i (\mathbf{m} \odot \mathbf{h}_{t-1}) + \mathbf{b}_i)$$ input gate

$$\mathbf{g}_o = \boldsymbol{\sigma}(\mathbf{W}_o \mathbf{x}_t + \mathbf{R}_o (\mathbf{m} \odot \mathbf{h}_{t-1}) + \mathbf{b}_o)$$ output gate

$$\mathbf{h}_t = \mathbf{g}_o \odot \tanh(\mathbf{c}_t)$$ output/recurrent input

$$\mathbf{m} \in \{0,1\}^{N_{\text{cells}}}$$

Binary dropout mask for recurrent input, sampled **once** per forward/backward pass [per mini-batch]

New: **LSTM with *variational* Dropout (cuDNN compatible)**

$$\mathbf{c}_t = \mathbf{g}_f \odot \mathbf{c}_{t-1} + \mathbf{g}_i \odot \widetilde{\mathbf{c}}_t$$

$$\widetilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + (\mathbf{M} \odot \mathbf{R}_c) \mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{g}_f = \boldsymbol{\sigma}(\mathbf{W}_f \mathbf{x}_t + (\mathbf{M} \odot \mathbf{R}_f) \mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{g}_i = \boldsymbol{\sigma}(\mathbf{W}_i \mathbf{x}_t + (\mathbf{M} \odot \mathbf{R}_i) \mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{g}_o = \boldsymbol{\sigma}(\mathbf{W}_o \mathbf{x}_t + (\mathbf{M} \odot \mathbf{R}_o) \mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{g}_o \odot \tanh(\mathbf{c}_t)$$

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}^T \\ \vdots \\ \mathbf{m}^T \end{pmatrix} \in \{0,1\}^{N_{\text{cells}}, N_{\text{cells}}}$$

Remark: input dropout as for feedforward networks (also sampled once per pass)

Neural network optimization

$$\mathrm{BAC} = \frac{\mathrm{Sensitivity} + \mathrm{Specificity}}{2}$$ averaged over the 13 classes

- Optimization objective: balanced accuracy (not differentiable)
- Cost function: cross entropy averaged over classes
  - class weights to balance the uneven training distribution
- Adam: modern mini-batch stochastic gradient descent Kingma & Ba 2015
  - Adaptive learning rate per parameter
  - Adaptive momentum per parameter
- Early stopping (free lunch) regularization
  - Stop when validation performance (avg. across the last few epochs) decreases
- Learning rate and batch size: *hyperparams that don't affect cost function*
  - Values first chosen s.t. training is most efficient
  - Fine tune them after cross validation based hyperparameter optimization

Hyperparameter Optimization

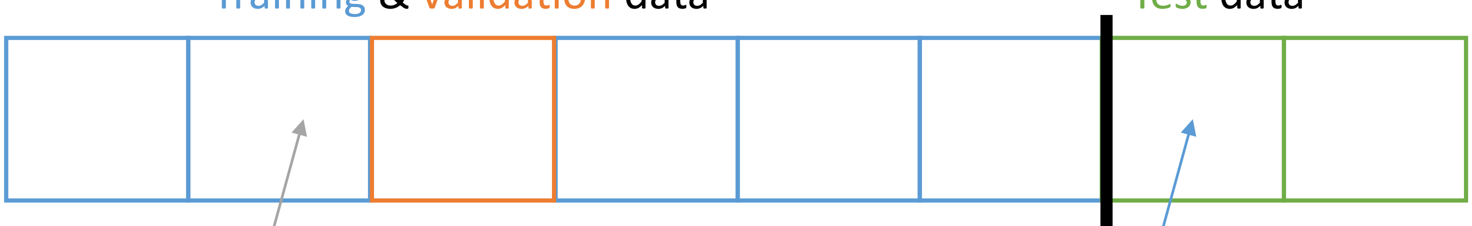
$$\mathrm{BAC} = \frac{\text{Sensitivity} + \text{Specificity}}{2}$$

Bergstra & Bengio 2012

- Approach: randomly sample hyperparameters
- Evaluation hyperparams. w.r.t. partial cross validation performance
  - Performance measure: balanced accuracy averaged over the 13 classes
  - First level: single 5-1 train-validation split => discard trash
  - Second level: two fixed (of 6) 5-1 splits => keep best 30% models
  - Third level: three (of 6) 5-1 splits => take best hyperparam. comb.
  - Fourth level: fine-tuning of learning rate, batch size, regularization strength
  - Final model: trained on all 6 folds, evaluated on test data

Training & validation data

Test data



1/8 of each class' sounds mixed together in 80 scenes (random nSrc, SNR, azimuth)

1/8 of sounds for each class mixed together in 126 scenes (systematically parametrized)

Multiconditional training as demonstrated: Trowitzsch, Mohr, Kashef, Obermayer 2017, IEEE Audio Speech Language Process.

Preliminary Results

Trowitzsch, Mohr, Kashef, Obermayer 2017, IEEE Audio Speech Language Process.

- For comparison: *block-interprete* labels
  - Class i active in last 500ms (with 75% coverage)? Then y_i=1 else 0
- Baseline model: logistic regression with L1 regularization
  - One feature vector for each block of size 500ms (2/3 overlap)
  - Features: statistical moments of ratemap/AMS and temporal derivatives thereof
  - Subsampling of data due to memory requirements of glmnet (here: harmless)
  - Cross validation Performance: **79.5 %** *nSrc-weighted* balanced accuracy v2
  - Validation performance of fold 1: 78.7% *nSrc-weighted* BAC_2 (>= BAC_2 [*probably*])
- Current state (beginning of hyperparameter search):
  - LSTM: 82.3 % balanced accuracy of full CV (fold 1: 83.4% BAC, **81.7 %** BAC_2) fold 6 (only slightly better, yet to be improved

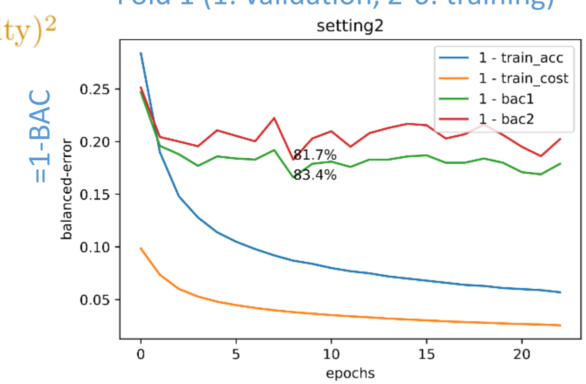
$$\mathrm{BAC} = \frac{\mathrm{Sensitivity} + \mathrm{Specificity}}{2} \qquad \mathrm{BAC}_2 = 1 - \sqrt{(1 - \mathrm{Sensitivity}^2) + (1 - \mathrm{Specificity})^2}$$

averaged over the 13 classes

Best model so-far (setting2)

- Backprop length T = 25sec (2500 frames)
- Batch size = 40
- LSTM layers: 3
- LSTM cells per layer: 581
- FC layers: 2
- Neurons per FC layer: 192
- Dropout rate: 10%
- Learning rate: default (0.001)

Fold 1 (1: validation, 2-6: training)



Challenges

- Limited resources
  - Despite: math cluster‘s GPUs & NVIDIA‘s donations mediated through Youssef
  - Consequence: hyperparameter space will be sampled rather coarsely
- Implementation complexity: specificities of data set and framework
  - One pipeline is already running, the second one almost
- Changes in design / methodology on the fly
  - Hopefully we have reached a stable configuration

Outlook

- Results for ConvNet and MLP model variants
  - Investigate effect of input history length T (and check input subsampling factor)
- More hyperparameter combinations for LSTM (level 1)
- (Partial) cross validation results (levels 2+3)
- Fine-tuning (level 4)
- Test set evaluation: effect of scene parameters (SNR, azimuth)
- *Instant* labels (=original labels; neural network models applicable)
- Change from Adam to problem-specific optimization:
  - Superconvergence (by large learning rates) Smith & Topin 2017
  - Learning to learn gradient descent by gradient descent Andrychowicz et al 2016

Contributions

- Heiner (NI project): data pipeline, LSTM, **visualizations**
- Changbin (Master Thesis): LSTM, **preliminary results**
- Alessandro (Bachelor Thesis): ConvNet
- Ivo: Dataset design, baseline (glmnet) model, supervision
- Moritz: MLP, ConvNet, supervision