Robust Classification using Hidden Markov models and Mixtures of Normalizing flows

Anubhab Ghosh, Antoine Honoré, Dong Liu, Gustav Eje Henter, Saikat Chatterjee

School of Electrical Engineering and Computer Science,
KTH Royal Institute of Technology, Stockholm, Sweden

30^{th} *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*

September 21-24, 2020

Outline

Introduction

Generative Modeling

- Focused on learning the joint distribution of the data $p(X, Y)$
- Advantages: Simplified mathematical structure, tractable training, can accommodate a growing number of classes
- Disadvantages: Conventional models like GMM-HMM not powerful enough for sequences

Discriminative Modeling

- Focused on learning the conditional distribution of the data $p(Y|X)$
- Advantages: Powerful modeling capability, larger context for modeling sequences, data-driven and model-free, etc.
- Disadvantages: Difficult to interpret, sensitive to adversarial perturbations, fixed number of classes, etc.

Questions and Challenges

Is there a way to ensure powerful modeling capability as well as mathematical tractability?

Normalizing flow models have been found to provide a powerful yet tractable method of estimating complicated probability distributions. They are capable of *exact* likelihood estimation and inference using cleverly designed architectures [4].

Previous work

An implementation of a mixture of Normalizing flows based HMM (using RealNVP flow) has been done by authors in [5] [6].

Research question

Can such models give some useful insights regarding robustness towards noisy data in the context of Speech Recognition?

NMM-HMM

Normalizing-flow mixture models (NMMs) are a class of generative models constituted as a weighted mixture of Normalizing flows [4]. The probabilistic model of the NMM-HMM for each hidden state is a weighted mixture of K density functions that is defined as:

$$p\left(\mathbf{x}|s;\mathbf{\Psi}_s\right) = \sum_{k=1}^{K} \pi_{s,k}\, p\left(\mathbf{x}|s;\phi_{s,k}\right) \tag{1}$$

In Eqn. (1), $\pi_{s,k} = p\left(k|s;\mathbf{H}\right)$, and they satisfy $\sum_{k=1}^{K} \pi_{s,k} = 1$, for each s. Assuming the function $\mathbf{g}_{s,k}$ is invertible, and the corresponding *normalizing* function is $\mathbf{f}_{s,k}$ (or equivalently $\mathbf{g}_{s,k}^{-1}$), s.t. $\mathbf{z} = \mathbf{f}_{s,k}\left(\mathbf{x}\right)$, we have:

$$p\left(\mathbf{x}|s;\mathbf{\Phi}_{s,k}\right) = p_{s,k}\left(\mathbf{f}_{s,k}\left(\mathbf{x}\right)\right)\left|\det\left(\frac{\partial \mathbf{f}_{s,k}\left(\mathbf{x}\right)}{\partial \mathbf{x}}\right)\right| \tag{2}$$

This equation shows that the density computation is exact.

Normalizing flows: Overview

Requirements of a Normalizing flow:

- Every Normalizing flow model is a function that defines a mapping from the data space $\mathrm{X} \in \mathbb{R}^{\mathbb{D}}$ to the latent space $\mathrm{Z} \in \mathbb{R}^{\mathbb{D}}$, and vice-versa.
- Both the latent and data space have to be of the *same dimensionality* in order for the mapping to considered *bijective*.
- The determinant (or *log* of determinant) of the Jacobian matrix $\left(det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right)$ needs to be *efficiently calculated*.

The signal flow for a *RealNVP* [2] based flow model having L layers:

$$\mathbf{z} = \mathbf{h_0} \underset{\mathbf{f}_{s,k}^{[1]}}{\overset{\mathbf{g}_{s,k}^{[1]}}{\rightleftharpoons}} \mathbf{h_1} \underset{\mathbf{f}_{s,k}^{[2]}}{\overset{\mathbf{g}_{s,k}^{[2]}}{\rightleftharpoons}} \mathbf{h_2} \underset{\mathbf{f}_{s,k}^{[3]}}{\overset{\mathbf{g}_{s,k}^{[3]}}{\rightleftharpoons}} \mathbf{h_3} \ldots \underset{\mathbf{f}_{s,k}^{[L]}}{\overset{\mathbf{g}_{s,k}^{[L]}}{\rightleftharpoons}} \mathbf{h_L} = \mathbf{x} \tag{3}$$

where $\mathbf{f}_{s,k}^{[l]}$ denotes the l^{th} layer network of $\mathbf{f}_{s,k}$, and each such $\mathbf{f}_{s,k}^{[l]}$ is invertible, i.e. $\mathbf{g}_{s,k}^{[l]} = \mathbf{f}^{-1}{}_{s,k}^{[l]}$ exists.

Normalizing flows: RealNVP flow

At every layer of the flow model the D-dimensional input feature is split into two parts. Let us assume the features are $[\mathbf{h}_{l,1:d}, \mathbf{h}_{l,d+1:D}]^T$ (where d denotes the number of components in the first sub part). The mapping defined by $\mathbf{f} : X \rightarrow Z$ is as follows:

$$\begin{aligned} \mathbf{h}_{l-1,1:d} &= \mathbf{h}_{l,1:d} \\ \mathbf{h}_{l-1,d+1:D} &= \left(\mathbf{h}_{l,d+1:D} - \mathbf{t}\left(\mathbf{h}_{l,1:d}\right)\right) \odot \exp\left(-\mathbf{s}\left(\mathbf{h}_{l,1:d}\right)\right) \end{aligned} \tag{4}$$

The inverse function ($\mathbf{g} : Z \rightarrow X$) is defined as:

$$\begin{aligned} \mathbf{h}_{l,1:d} &= \mathbf{h}_{l-1,1:d} \\ \mathbf{h}_{l,d+1:D} &= \mathbf{h}_{l-1,d+1:D} \odot \exp\left(\mathbf{s}\left(\mathbf{h}_{l-1,1:d}\right)\right) + \mathbf{t}\left(\mathbf{h}_{l-1,1:d}\right) \end{aligned} \tag{5}$$

where $\odot$ denotes element-wise multiplications, $\mathbf{s} : \mathbb{R}^d \rightarrow \mathbb{R}^{D-d}$, $\mathbf{t} : \mathbb{R}^d \rightarrow \mathbb{R}^{D-d}$, with $\mathbf{s}, \mathbf{t}$ being shallow feed-forward Neural Nets (`tanh`) activation function for ($\mathbf{s}$) (modeling logarithm of standard deviation) and an identity activation for ($\mathbf{t}$) [2].

Normalizing flows: RealNVP flow (Contd.)

The *alternate switching* is needed between the two splits of the data $\mathbf{x}_1, \mathbf{x}_2$ to make sure all elements of the data are transformed. An illustration of a RealNVP flow-network with 2 coupling layers is shown in Fig. 1.

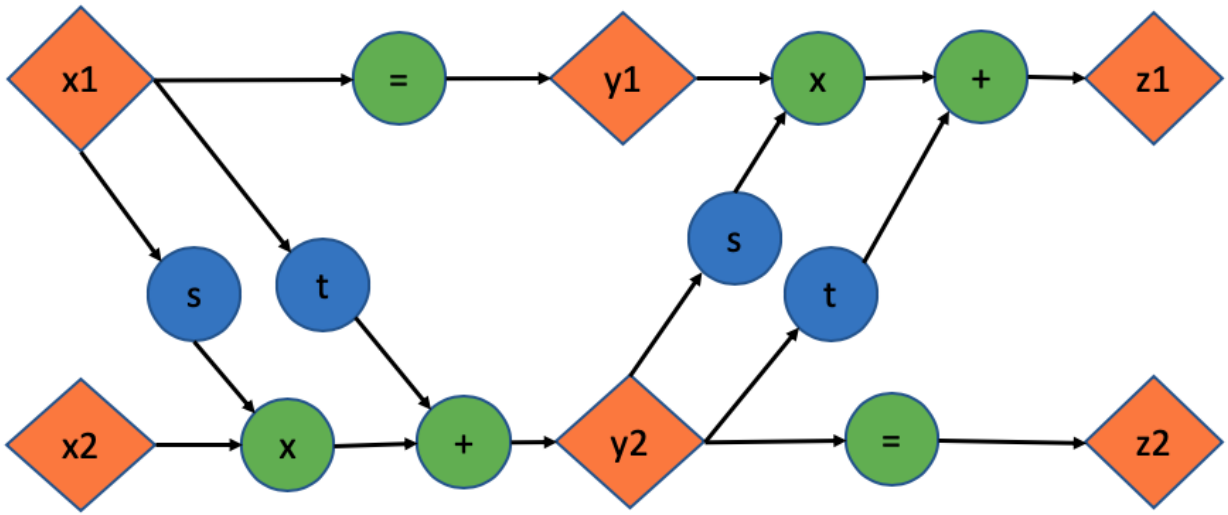

Figure 1: Illustrating the alternate switching in a RealNVP flow network with 2 coupling layers in the generative direction [2]

Using maximum likelihood approach

A sequential signal is thus represented by a set of feature vector such as $\bar{\mathbf{x}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N]$ (N denoting the length of the sequential signal) and each component $\mathbf{x}_i \in \mathrm{R}^K$ (where K denotes the dimensionality of the feature vector).

$$\mathbf{H}_{opt.} = \arg\max_{\mathbf{H} \in \mathcal{H}} \frac{1}{M} \sum_{m=1}^{M} \log\left(p\left(\bar{\mathbf{x}}^{(m)}; \mathbf{H}\right)\right) \tag{6}$$

The models were trained using the Expectation Maximization (EM) algorithm [1].

- A batch of training data for a particular class of **observed** features consisted of M such signals so the concatenated set consisted of $\mathbf{X} = \left[\bar{\mathbf{x}}^{(\mathbf{i})}\right]_{i=1}^{M}$ signals.
- The **hidden** variables were the sequences of state vectors (denoted by $\bar{\mathbf{s}}$) and the sequences of mixture component indexes (denoted by $\bar{\mathbf{k}}$).

EM formulation

Expectation step

$$\mathcal{L}\left(\mathbf{H};\mathbf{H}^{old}\right)=\mathbf{E}_{p\left(\bar{\mathbf{s}},\bar{\mathbf{k}}|\bar{\mathbf{x}};\mathbf{H}^{old}\right)}\log\left(p\left(\bar{\mathbf{x}},\bar{\mathbf{s}},\bar{\mathbf{k}};\mathbf{H}\right)\right) \tag{7}$$

Maximization step

$$\max_{\mathbf{H}}\mathcal{L}\left(\mathbf{H};\mathbf{H}^{old}\right)=\max_{\mathbf{q}}\mathcal{L}\left(\mathbf{q};\mathbf{H}^{old}\right)+\max_{\mathbf{A}}\mathcal{L}\left(\mathbf{A};\mathbf{H}^{old}\right)+\max_{\mathbf{\Psi}}\mathcal{L}\left(\mathbf{\Psi};\mathbf{H}^{old}\right) \tag{8}$$

where,

$$\mathcal{L}\left(\mathbf{q};\mathbf{H}^{old}\right)=\mathbf{E}_{p(\bar{\mathbf{s}}|\bar{\mathbf{x}};\mathbf{H}^{old})}\log\left(p\left(s_1;\mathbf{H}\right)\right) \tag{9}$$

$$\mathcal{L}\left(\mathbf{A};\mathbf{H}^{old}\right)=\mathbf{E}_{p(\bar{\mathbf{s}}|\bar{\mathbf{x}};\mathbf{H}^{old})}\sum_{t=2}^{T}\log\left(p\left(s_t|s_{t-1};\mathbf{H}\right)\right) \tag{10}$$

$$\mathcal{L}\left(\mathbf{\Psi};\mathbf{H}^{old}\right)=\mathbf{E}_{p\left(\bar{\mathbf{s}},\bar{\mathbf{k}}|\bar{\mathbf{x}};\mathbf{H}^{old}\right)}\log\left(p\left(\bar{\mathbf{x}},\bar{\mathbf{k}}|\bar{\mathbf{s}};\mathbf{H}\right)\right) \tag{11}$$

$$\mathcal{L}\left(\mathbf{\Phi};\mathbf{H}^{old}\right)=\mathbf{E}_{p\left(\bar{\mathbf{s}},\bar{\mathbf{k}}|\bar{\mathbf{x}};\mathbf{H}^{old}\right)}[\log\left(p_{\bar{\mathbf{s}},\bar{\mathbf{k}}}\left(\mathbf{f}_{s,k}\left(\bar{\mathbf{x}}\right)\right)\right)+\log\left(\left|\det\left(\nabla\mathbf{f}_{s,k}\right)\right|\right)] \tag{12}$$

Overview of the learning algorithm

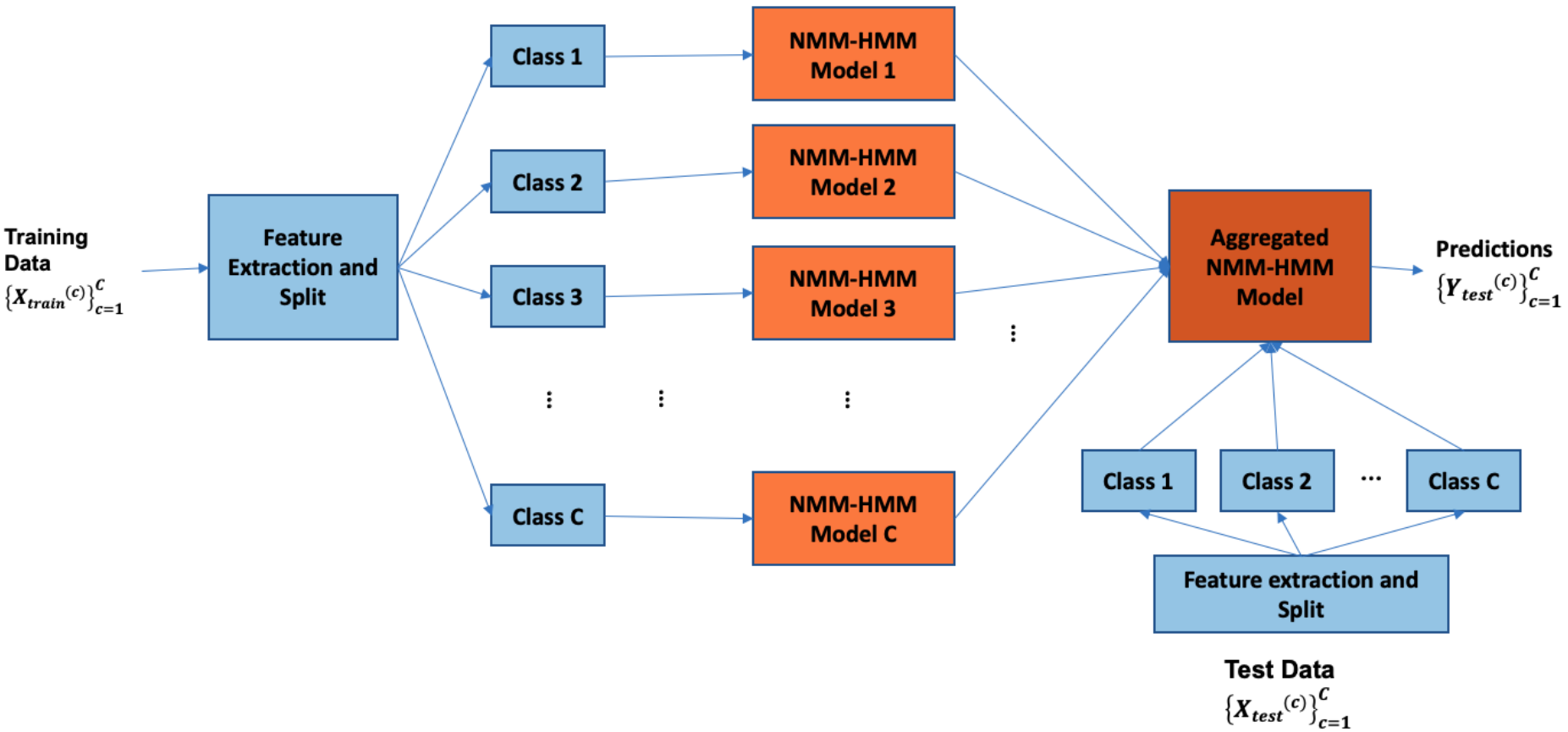

Figure 2: A schematic depicting the training and testing of NMM-HMM models on data having C classes

Dataset used for Phone recognition

Phone Recognition

- A phone is a distinct speech sound and is universal irrespective of the language under consideration.
- Understanding the spoken utterance at the phone-level is known as **Phone Recognition** (*Language model* not required!)

TIMIT Dataset

- Phoneme-level labelled utterances.
  - # training sequences: 4620
  - # testing sequences: 1680
- Utterances sampled at 16 kHz
- Originally 61 phones, but a smaller folded set of 39 phones used.
- Noise types: white, babble, pink, hfchannel (16 kHz) from NOISEX-92 dataset [7].
  - Random Noise snippets *added* to Clean utterances
  - Additionally Signal to Noise ratio (SNR) was varied while adding noise

Details of training

- Features used were Mel-frequency cepstral coefficients (MFCCs) including dynamic features like differential (Δ) and acceleration ($\Delta\Delta$) features to capture temporal information.
- The code for the flow based HMM models (RealNVP) was written using Python, PyTorch and run on GPU. The code for the GMM-HMM model was written purely using *hmmlearn* [6].
- NMM-HMMs trained using Adam [3]. $\eta_{NVP} = 4e-3$ for NMM-HMM along with step-wise adaptive learning rate decay.
- Maximization of the log-likelihood (or minimization of the negative log-likelihood), and monitoring relative change of the same (*used as a convergence criterion*).
- The evaluation was done using a metric known as *accuracy* (or equivalently $100 - PER\%$) computed on the test set.
  - Accuracy was computed as a simple percentage of the number of correctly predicted phones (phone label) among the total number of phones (phone labels) in a given dataset.

Training and Testing on Clean data

Table 1: Test accuracy (in %) for GMM-HMM at varying number of mixture components (K_g) on clean data

Model-Type	No. of components (K_g)			
	K_g=3	K_g=10	K_g=15	K_g=20
GMM-HMM	66.7	70.8	71.9	72.8

Table 2: Test accuracy (in %) for NMM-HMM at varying number of mixture components (K_g) on clean data

Model-Type	No. of components (K_g)	
	K_g=1	K_g=3
NMM-HMM	76.7	77.6

Training on Clean and Testing on Noisy data

Table 3: Chosen Model configurations for GMM-HMM and NMM-HMM

Model Type	Configuration details
GMM-HMM	$K_g = 20$, *Covariance Matrix type*: Diagonal
NMM-HMM	$K_g = 3$, No. of *flow-blocks:* 4, Dimension of hidden channel: 24, Learning rate (η): 4e-3

Table 4: Test accuracy (in %) for clean and various noise conditions. We compare GMM-HMM and NMM-HMM for folded 39-phone classification. We use the notations GMM and NMM to represent GMM-HMM and NMM-HMM, respectively. The performance drop is shown in parenthesis with respect to the clean train and clean test scenario as in Tables 1 and 2.

Performance for clean data training and testing as a reference: GMM: 72.8 and NMM: 77.6								
Type of Noise	SNR levels for different kinds of noises							
	25dB		20dB		15dB		10dB	
	GMM	NMM	GMM	NMM	GMM	NMM	GMM	NMM
white	55.6 (17.2)	67.1 (10.5)	46.8 (26.0)	60.0 (17.6)	36.8 (36.0)	49.4 (28.2)	27.9 (44.9)	37.7 (39.9)
babble	65.7 (7.1)	70.7 (6.9)	59.3 (13.5)	65.8 (11.8)	49.3 (23.5)	56.2 (21.4)	37.4 (35.4)	42.3 (35.3)
hfchannel	62.3 (10.5)	67.9 (9.7)	54.4 (18.4)	63.4 (14.2)	44.1 (28.7)	55.8 (21.8)	33.3 (39.5)	44.9 (32.7)
pink	59.9 (12.9)	69.3 (8.3)	51.9 (20.9)	61.7 (15.9)	42.3 (30.5)	48.6 (29)	32.2 (40.6)	33.7 (43.9)

Training on Clean and Testing on Noisy data

Table 3: Chosen Model configurations for GMM-HMM and NMM-HMM

Model Type	Configuration details
GMM-HMM	$K_g = 20$, *Covariance Matrix type*: Diagonal
NMM-HMM	$K_g = 3$, No. of *flow-blocks:* 4, Dimension of hidden channel: 24, Learning rate (η): 4e-3

Table 4: Test accuracy (in %) for clean and various noise conditions. We compare GMM-HMM and NMM-HMM for folded 39-phone classification. We use the notations GMM and NMM to represent GMM-HMM and NMM-HMM, respectively. The performance drop is shown in parenthesis with respect to the clean train and clean test scenario as in Tables 1 and 2.

Performance for clean data training and testing as a reference: GMM: 72.8 and NMM: 77.6								
Type of Noise	SNR levels for different kinds of noises							
	25dB		20dB		15dB		10dB	
	GMM	NMM	GMM	NMM	GMM	NMM	GMM	NMM
white	55.6 (17.2)	67.1 (10.5)	46.8 (26.0)	60.0 (17.6)	36.8 (36.0)	49.4 (28.2)	27.9 (44.9)	37.7 (39.9)
babble	65.7 (7.1)	70.7 (6.9)	59.3 (13.5)	65.8 (11.8)	49.3 (23.5)	56.2 (21.4)	37.4 (35.4)	42.3 (35.3)
hfchannel	62.3 (10.5)	67.9 (9.7)	54.4 (18.4)	63.4 (14.2)	44.1 (28.7)	55.8 (21.8)	33.3 (39.5)	44.9 (32.7)
pink	59.9 (12.9)	69.3 (8.3)	51.9 (20.9)	61.7 (15.9)	42.3 (30.5)	48.6 (29)	32.2 (40.6)	33.7 (43.9)

Training on Clean and Testing on Noisy data

Table 3: Chosen Model configurations for GMM-HMM and NMM-HMM

Model Type	Configuration details
GMM-HMM	$K_g = 20$, *Covariance Matrix type*: Diagonal
NMM-HMM	$K_g = 3$, No. of *flow-blocks:* 4, Dimension of hidden channel: 24, Learning rate (η): 4e-3

Table 4: Test accuracy (in %) for clean and various noise conditions. We compare GMM-HMM and NMM-HMM for folded 39-phone classification. We use the notations GMM and NMM to represent GMM-HMM and NMM-HMM, respectively. The performance drop is shown in parenthesis with respect to the clean train and clean test scenario as in Tables 1 and 2.

Performance for clean data training and testing as a reference: GMM: 72.8 and NMM: 77.6								
Type of Noise	SNR levels for different kinds of noises							
	25dB		20dB		15dB		10dB	
	GMM	NMM	GMM	NMM	GMM	NMM	GMM	NMM
white	55.6 (17.2)	67.1 (10.5)	46.8 (26.0)	60.0 (17.6)	36.8 (36.0)	49.4 (28.2)	27.9 (44.9)	37.7 (39.9)
babble	65.7 (7.1)	70.7 (6.9)	59.3 (13.5)	65.8 (11.8)	49.3 (23.5)	56.2 (21.4)	37.4 (35.4)	42.3 (35.3)
hfchannel	62.3 (10.5)	67.9 (9.7)	54.4 (18.4)	63.4 (14.2)	44.1 (28.7)	55.8 (21.8)	33.3 (39.5)	44.9 (32.7)
pink	59.9 (12.9)	69.3 (8.3)	51.9 (20.9)	61.7 (15.9)	42.3 (30.5)	48.6 (29)	32.2 (40.6)	33.7 (43.9)

Training on Clean and Testing on Noisy data

Table 3: Chosen Model configurations for GMM-HMM and NMM-HMM

Model Type	Configuration details
GMM-HMM	$K_g = 20$, *Covariance Matrix type*: Diagonal
NMM-HMM	$K_g = 3$, No. of *flow-blocks:* 4, Dimension of hidden channel: 24, Learning rate (η): 4e-3

Table 4: Test accuracy (in %) for clean and various noise conditions. We compare GMM-HMM and NMM-HMM for folded 39-phone classification. We use the notations GMM and NMM to represent GMM-HMM and NMM-HMM, respectively. The performance drop is shown in parenthesis with respect to the clean train and clean test scenario as in Tables 1 and 2.

Performance for clean data training and testing as a reference: GMM: 72.8 and NMM: 77.6								
Type of Noise	SNR levels for different kinds of noises							
	25dB		20dB		15dB		10dB	
	GMM	NMM	GMM	NMM	GMM	NMM	GMM	NMM
white	55.6 (17.2)	67.1 (10.5)	46.8 (26.0)	60.0 (17.6)	36.8 (36.0)	49.4 (28.2)	27.9 (44.9)	37.7 (39.9)
babble	65.7 (7.1)	70.7 (6.9)	59.3 (13.5)	65.8 (11.8)	49.3 (23.5)	56.2 (21.4)	37.4 (35.4)	42.3 (35.3)
hfchannel	62.3 (10.5)	67.9 (9.7)	54.4 (18.4)	63.4 (14.2)	44.1 (28.7)	55.8 (21.8)	33.3 (39.5)	44.9 (32.7)
pink	59.9 (12.9)	69.3 (8.3)	51.9 (20.9)	61.7 (15.9)	42.3 (30.5)	48.6 (29)	32.2 (40.6)	33.7 (43.9)

Training on Clean and Noisy data

Table 5: Test accuracy (in %) using noisy training. The performance drop is shown in parenthesis with respect to the clean train and clean test scenario. The models were trained using a mixture of clean data and white-noise corruputed data at 10dB SNR

Model-Type	Clean	white noise	
		15dB	10dB
GMM-HMM	72.0	55.9 (16.1)	53.7 (18.3)
NMM-HMM	76.8	69.2 (7.6)	65.7 (11.1)

Conclusion and Future work

Conclusion:

- It is possible to use neural networks for improving maximum-likelihood based classification performance and robustness against noise
- methods are able to use time-tested signal processing based features as MFCCs
- Tractable machine learning techniques as expectation-maximization for training the models

Future Work:

- Use of other input features such as logarithm of the power spectrum.
- Use of other normalizing flow models for modeling output distributions

References I

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real NVP". In: *Proc. of ICLR 2017 - Conference Track*. 2019. arXiv: 1605.08803.

[3] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *Proc. of ICLR 2015*. 2014. arXiv: 1412.6980.

[4] Ivan Kobyzev, Simon Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: i (2019), pp. 1–16. arXiv: 1908.09257. URL: http://arxiv.org/abs/1908.09257.

[5] Dong Liu et al. "Neural network based explicit mixture models and expectation-maximization based learning". In: *International Joint Conference on Neural Networks (IJCNN)*. 2020.

[6] Dong Liu et al. "Powering Hidden Markov Model by Neural Network based Generative Models". In: *European Conference on Artificial Intelligence (ECAI)*. 2020.

[7] Andrew Varga and Herman J.M. Steeneken. "Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems". In: *Speech Communication* 12.3 (1993), pp. 247–251. ISSN: 01676393. DOI: 10.1016/0167-6393(93)90095-3.

Thanks for Listening!

Questions ? Comments ?