The Speaker and Language Recognition Workshop (Odyssey 2024)
18-21 June 2024, Quebec City, Canada

Normalizing Flows for Speaker and Language Recognition Backend

Aleix Espuña[1], *Amrutha Prasad*[1,2], *Petr Motlicek*[1,2],
Srikanth Madikeri[1], *Christof Schuepbach*[3]

[1] Idiap Research Institute, Martigny, Switzerland
[2] Brno University of Technology, Brno, Czech Republic
[3] Armasuisse Science and Technology, Thun, Switzerland

aleix.fontcuberta@gmail.com, {amrutha.prasad, petr.motlicek, mrsrikanth}@idiap.ch
christof.schuepbach@armasuisse.ch

Abstract

In this paper, we address the Gaussian distribution assumption made in PLDA, a popular back-end classifier used in Speaker and Language recognition tasks. We study normalizing flows, which allow using non-linear transformations and still obtain a model that can explicitly represent a probability density. The model makes no assumption about the distribution of the observations. This alleviates the need for length normalization, a well known data preprocessing step used to boost PLDA performance. We demonstrate the effectiveness of this flow model on NIST SRE16, LRE17 and LRE22 datasets. We observe that when applying length normalization, both the flow model and PLDA achieve similar EERs for SRE16 (11.5% vs 11.8%). However, when length normalization is not applied, the flow shows more robustness and offers better EERs (13.1% vs 17.1%). For LRE17 and LRE22, the best classification accuracies (84.2%, 75.5%) are obtained by the flow model without any need for length normalization.

1. Introduction

Normalizing flows are a general class of machine algorithms that use the transformation theorem [1, Chapter 1.2.1] in order to model a probability distribution. This is done through the application of invertible transformations on latent variables that belong to a known distribution [2, 3]. Although they gained their popularity within the field of generative modeling [4], the fact that they represent a probability distribution makes it theoretically possible to use them for classification and regression tasks. In this work, we study normalizing flows for speaker recognition (SRE) and language recognition (LRE) tasks.

In speaker recognition, the goal is to develop a model capable of verifying a speaker from a set of recordings. In language recognition, the goal is to determine the language of an utterance. Both speaker and language recognition tasks typically use a 2-step approach: (i) One first uses the training set to train a deep learning model such that it learns to transform each utterance into a single vector, which we call an embedding. The embeddings must satisfy clustering properties: those belonging to the same class must be close together in the Euclidean space, while those belonging to different classes must be separated. In speaker recognition, a class is a speaker. In language recognition, a class is a language. (ii) Once these training embeddings have been produced, they form a new training set, which is used to develop simple recognition models such as cosine distance or Probabilistic Linear Discriminant Analysis (PLDA) [5]. During evaluation, the recognition algorithm receives two embedded utterances whose classes are unknown. The goal is to provide a likelihood ratio between the same class or different class hypotheses.

1.1. Motivation

PLDA is a useful model deployed for classification tasks that assumes an infinite number of classes can exist. It defines the process that generates an observation $\boldsymbol{x} \in \mathcal{R}^d$ as the following:

1. A latent variable $\boldsymbol{v} \in \mathcal{R}^d$ is sampled from a Gaussian with zero mean and diagonal covariance matrix Ψ, i.e., $\boldsymbol{v} \sim \mathcal{N}(0, \Psi)$. This represents the creation of a new class.
2. The latent variable $\boldsymbol{v}$ is used as the center of a new Gaussian with identity covariance matrix. Once fixed, a latent variable $\boldsymbol{u}$ is sampled: $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{v}, I)$.
3. Finally, the latent variable $\boldsymbol{u}$ is linearly transformed to produce the observation $\boldsymbol{x} := \boldsymbol{m} + A\boldsymbol{u}$, where the vector $\boldsymbol{m} \in \mathcal{R}^d$ is a bias, and $A \in \mathcal{R}^{d \times d}$ is a matrix.

The linear transformation $\boldsymbol{x} := \boldsymbol{m} + A\boldsymbol{u}$ allows for separately modeling the within class covariance matrix and between class covariance matrix of the data. Hence, PLDA is often known as the two-covariance model. However, one of the first consequences of this transformation is that PLDA models observations as Gaussian variables. This can be a strong assumption when the observations are for instance embedding vectors extracted by a neural network. Consequently, length normalization [6] is often applied to preprocess the vector embeddings and make them approximate better the Gaussian assumption of PLDA. In [7], the authors explored heavy-tailed PLDA [8], which replaces the Gaussian distributions with Student's t distributions. This approach is computationally expensive. In later work [9], the authors explored applying a non-linear transformation to the embedding vectors before passing them to PLDA. This requires an additional preprocessing step.

In this work, we relax this assumption by using an invertible non-linear transformation $\boldsymbol{x} := g(\boldsymbol{u})$ instead of a linear one. The main advantage is the explicit representation of an arbitrary probability density that makes no assumption about the distribution of the observations. Hence, we expect to observe more robustness with respect to PLDA when not applying length normalization in our vector embeddings.

In [10, 11], this idea is already explored using a composition of affine and non-linear transformations based on the $\sinh$ non-linearity. The resulting model requires a scaling factor that

10.21437/odyssey.2024-11

complicates the computation of likelihood ratios during evaluation. The authors train the model via Expectation Maximization (EM) [12, Chapter 9] using i-vector [13] and e-vector [14] embeddings. In our study, we present a different model whose transformation is based on the composition of a single non-linearity. The model is trained via gradient descent and it allows using the original PLDA equations during evaluation. Another difference is that we conduct our experiments using embeddings from the x-vector [15] and XLS-R [16] models.

Section 2 describes the flow model applied as the backend for SRE and LRE tasks. Section 3 provides an overview of the training, development, and test sets. The experiments for SRE and LRE are presented in Section 4 and the results are discussed in Section 4.4. The overview of our findings in this study is provided in Section 5.

2. Model

2.1. The Density

According to PLDA, observations that have been generated from the same latent variable $\boldsymbol{v}$ are dependent and considered to belong to the same class. The joint distribution of a set of dependent observations $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\theta)$ depends on the parameters θ. They are the matrix A, the intercept $\boldsymbol{m}$, and the diagonal covariance matrix Ψ. The optimal θ is found by maximizing the likelihood of the whole dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = \sum_{k=1}^{K} \log p(\boldsymbol{x}_i : i \in \mathcal{C}_k \mid \theta), \qquad (1)$$

where K is the total number of classes, N is the total number of observations, and $\mathcal{C}_k$ is the set of indexes of the observations that belong to class k. Because PLDA just applies a linear transformation, Equation 1 can be maximized analytically if all classes have the same number of observations, or by using EM if it is not the case.

Now, instead of applying a linear transform $\boldsymbol{x} = A\boldsymbol{u} + \boldsymbol{m}$, we are defining a parameterized, invertible, and non-linear transformation $\boldsymbol{x} = g(\boldsymbol{u}, \lambda)$. The transformation g involves a new set of parameters λ and it has inverse $h = g^{-1}$. At the same time, we keep intact the PLDA assumptions regarding the latent variables, i.e. $p(\boldsymbol{u}|\boldsymbol{v}) = \mathcal{N}(\boldsymbol{v}, I)$ and $p(\boldsymbol{v}|\Psi) = \mathcal{N}(0, \Psi)$. Consequently, we have defined a new model with parameters λ and Ψ. The remaining PLDA parameters A and $\boldsymbol{m}$ have disappeared. In order to train this model via maximum likelihood, we need to find how it represents the joint distribution $p(\boldsymbol{x}_i : i \in \mathcal{C}_k \mid \lambda, \Psi)$ of a set of dependent observations that belong to a class k. In the following derivation, for simplicity, we use the notation $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\lambda, \Psi)$ in order to refer to $p(\boldsymbol{x}_i : i \in \mathcal{C}_k \mid \lambda, \Psi)$.

From PLDA, we already know which is the joint distribution $p(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n|\Psi)$ of a set of latent variables that belong to the same class [5, Equation 6]. If we now transform each variable with $\boldsymbol{x}_i = g(\boldsymbol{u}_i, \lambda)$ with inverse $\boldsymbol{u}_i = h(\boldsymbol{x}_i, \lambda)$, the transformation theorem claims that the new joint distribution $p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n|\lambda, \Psi)$ takes the following form:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n|\lambda, \Psi) = p(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \boldsymbol{u}_n|\Psi) \prod_{i=1}^{n} \left| \frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i} \right|. \qquad (2)$$

Where $|\frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i}|$ represents the absolute value of the determinant of the Jacobian $\frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i}$. It is not hard to prove that if we choose $\boldsymbol{x} = g(\boldsymbol{u}) = A\boldsymbol{u} + \boldsymbol{m}$ we obtain the PLDA distribution. Therefore, our equations are a general case of the PLDA model. We can take the logarithm of Equation 2 to obtain an explicit expression for the log-likelihood:

$$\begin{aligned} \log p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\lambda, \Psi) &= -\frac{nd}{2} \log(2\pi) \qquad (3) \\ &+ \sum_{i=1}^{n} \log \left| \frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i} \right| - \frac{1}{2} \sum_{t=1}^{d} \log(\Psi_t + \frac{1}{n}) \\ &- \frac{1}{2} \sum_{t=1}^{d} \frac{\bar{u_t}^2}{(\Psi_t + \frac{1}{n})} - \frac{1}{2} \sum_{i=1}^{n} \sum_{t=1}^{d} (u_{it} - \bar{u}_t)^2. \end{aligned}$$

The data dimensionality is d and the scalar u_{it} is just component t of the vector $\boldsymbol{u}_i = h(\boldsymbol{x}_i, \lambda)$. We also denote by $\bar{\boldsymbol{u}} := \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{u}_i$ the average of all vectors $\boldsymbol{u}_i$ and $\bar{u}_t$ is component t of vector $\bar{\boldsymbol{u}}$.

Our density assumes that the parameterized transformation $\boldsymbol{u} = h(\boldsymbol{x}, \lambda)$ generates latent variables of the same class that are normally distributed with unit variance around the center $\bar{\boldsymbol{u}}$. Moreover, the center $\bar{\boldsymbol{u}}$ is normally distributed with zero mean and covariance matrix controlled by the diagonal matrix Ψ and the number of observations within the class, just like in PLDA.

2.2. The Flow Transformation

The transformation $\boldsymbol{u} = h(\boldsymbol{x}, \lambda)$ that we use is called coupling layer [17]. It is a non-volume preserving transformation that transforms the vector $\boldsymbol{x} \in \mathcal{R}^d$ into a new vector $\boldsymbol{u} \in \mathcal{R}^d$. It splits the vector $\boldsymbol{x}$ into two halves: $\boldsymbol{x}_1 \in \mathcal{R}^{\frac{d}{2}}$ and $\boldsymbol{x}_2 \in \mathcal{R}^{\frac{d}{2}}$. The first half $\boldsymbol{x}_1$ remains unmodified, forming $\boldsymbol{u}_1 := \boldsymbol{x}_1$. The second half is transformed according to:

$$\boldsymbol{u}_2 := (\boldsymbol{x}_2 - \boldsymbol{t}) \exp(-\boldsymbol{s}), \qquad (4)$$

where the variables $\boldsymbol{s}$ and $\boldsymbol{t}$ have been produced by a neural network function f that receives $\boldsymbol{x}_1$ as input: $\boldsymbol{s}, \boldsymbol{t} = f(\boldsymbol{x}_1, \lambda)$. The vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are concatenated to form the new vector $\boldsymbol{u}$. It is easy to check that this transformation is non-linear and invertible. The parameters λ of the coupling layer transformation are the parameters of the neural network f that has been used. The most common choice is to use a Convolutional Neural Network (CNN) [18].

2.3. The overall flow model architecture

Because the composition of invertible transformations is still an invertible transformation, we can create a powerful non-linear function by composing several coupling layers. Mathematically, we define $h(\boldsymbol{x})$ as $h(\boldsymbol{x}) = h_N \circ h_{N-1} \circ \ldots \circ h_1$, where $h_1, h_2, \ldots, h_N$ are N coupling layer transformations and $\circ$ denotes function composition.

When composing multiple layers, it is important to vary the way in which the vector $\boldsymbol{x}$ is split at each layer. This is done to ensure that all components of the vector $\boldsymbol{x}$ are modified after the full transformation. To achieve this, we follow the simple approach of just splitting in half the vector $\boldsymbol{x}$ and inverting the splitting at each layer. We are aware there are clever ways of achieving this, such as introducing additional invertible layers based on rotation matrices, as done in [19]. In this first study we decide to start with the simplest possible flow. A summary of the presented ideas is illustrated in the block diagram of Figure 1, which also establishes a comparison between the flow and PLDA architectures.

Finally, each coupling layer has its own CNN network with a well defined goal: it must receive one half of the vector

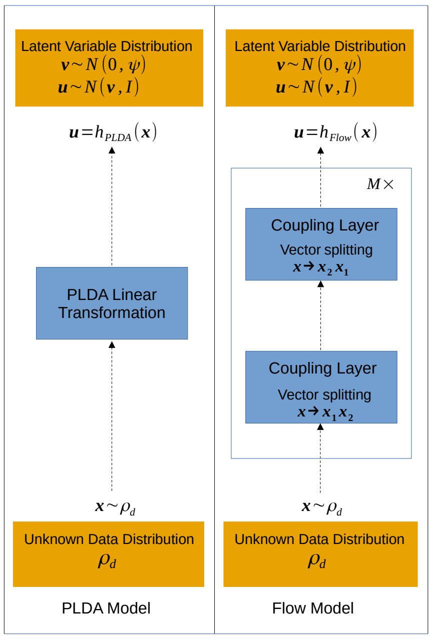
Latent Variable Distribution
$\boldsymbol{v} \sim N(0, \psi)$
$\boldsymbol{u} \sim N(\boldsymbol{v}, I)$
$\boldsymbol{u} = h_{PLDA}(\boldsymbol{x})$
PLDA Linear Transformation
$\boldsymbol{x} \sim \rho_d$
Unknown Data Distribution
ρ_d
PLDA Model
Latent Variable Distribution
$\boldsymbol{v} \sim N(0, \psi)$
$\boldsymbol{u} \sim N(\boldsymbol{v}, I)$
$\boldsymbol{u} = h_{Flow}(\boldsymbol{x})$
$M\times$
Coupling Layer
Vector splitting
$\boldsymbol{x} \rightarrow \boldsymbol{x}_2 \boldsymbol{x}_1$
Coupling Layer
Vector splitting
$\boldsymbol{x} \rightarrow \boldsymbol{x}_1 \boldsymbol{x}_2$
$\boldsymbol{x} \sim \rho_d$
Unknown Data Distribution
ρ_d
Flow Model

Figure 1: Overview of the PLDA and flow model backends in SRE and LRE. PLDA receives a sample $\boldsymbol{x}$ from the unknown data distribution ρ_d, applies a linear transformation $h_{\text{PLDA}}(\boldsymbol{x})$ and assumes the resulting latent variable $\boldsymbol{u}$ belongs to a Gaussian. The flow model instead performs multiple non-linear transformations on the data sample.

$\boldsymbol{x} \in \mathcal{R}^d$, namely $\boldsymbol{x}_1 \in \mathcal{R}^{\frac{d}{2}}$, and it must output the scale and intercept vectors $\boldsymbol{s}, \boldsymbol{t}$ that are needed in Equation 4. Our implementation achieves this by using a CNN that starts with a linear transform from $\frac{d}{2}$ to d units. It then uses 1D convolutions with padding that preserve the d dimensions during forward propagation. The network f returns a vector $f(\boldsymbol{x}_1) \in \mathcal{R}^d$ that is split in two to obtain $\boldsymbol{s}$ and $\boldsymbol{t}$.

2.4. Optimization

The model parameters that need to be optimized are the covariance matrix Ψ and the CNN parameters λ of each coupling layer. In this work, we take a simplified approach and we train first a PLDA model via EM. Then, we take the optimal matrix Ψ^* and we fix it for our flow model.

The optimization method that we use for learning λ is stochastic gradient descent. We randomly select a batch of data and we perform an update in order to maximize its likelihood. We repeat the procedure until the likelihood saturates. More specifically, we randomly select B classes $k_1, k_2, \ldots, k_B$. Our batch loss $l(\lambda, \Psi^*)$ is the Negative Log-Likelihood (NLL) of the data involved in the B classes:

$$l(\lambda, \Psi^*) = -\frac{1}{B}\sum_{b=1}^{B} \log p(\boldsymbol{x}_i : i \in C_{k_b} | \lambda, \Psi^*). \tag{5}$$

2.5. Evaluation

It is straightforward to do verification with the model. Verification consists in receiving a vector $\boldsymbol{x}^{\text{enroll}}$ and a vector $\boldsymbol{x}^{\text{test}}$ and computing the log-likelihood ratio between the target and non-target hypotheses. Following our independence assumptions, the ratio takes the following form:

$$R(\boldsymbol{x}^{\text{enroll}}, \boldsymbol{x}^{\text{test}}) := \log\left(\frac{p(\boldsymbol{x}^{\text{enroll}}, \boldsymbol{x}^{\text{test}}|\lambda, \Psi)}{p(\boldsymbol{x}^{\text{enroll}}|\lambda, \Psi)p(\boldsymbol{x}^{\text{test}}|\lambda, \Psi)}\right). \tag{6}$$

At this point we notice that when using Equation 2 in order to compute Equation 6, we keep a desirable property of PLDA: there is no need to compute the Jacobians, as they cancel each other out in the fraction. This means our model only requires evaluating the likelihood of the Gaussian latent variables, obtaining the PLDA equation:

$$R(\boldsymbol{x}^{\text{enroll}}, \boldsymbol{x}^{\text{test}}) := \log\left(\frac{p(\boldsymbol{u}^{\text{enroll}}, \boldsymbol{u}^{\text{test}}|\Psi)}{p(\boldsymbol{u}^{\text{enroll}}|\Psi)p(\boldsymbol{u}^{\text{test}}|\Psi)}\right), \tag{7}$$

where $\boldsymbol{u}^{\text{enroll}} = h(\boldsymbol{x}^{\text{enroll}}, \lambda)$ and $\boldsymbol{u}^{\text{test}} = h(\boldsymbol{x}^{\text{test}}, \lambda)$.

3. Datasets

This section describes the datasets used for training and evaluating the flow model on SRE and LRE tasks.

3.1. Speaker Recognition Evaluation (SRE)

The NIST 2016 Speaker Recognition Evaluation Plan (SRE16 [1]) [20] consists of telephone calls in 4 different languages obtained from the Call My Net [21] corpus. It comprises training, development, enrollment, and evaluation sets. The training data is obtained from the previous NIST SREs 04, 05, 06, 08, 10, Switchboard [22], Fisher [23] and Mixer 6 [24] corpora. It has a total duration of 5700 h. The development, enrollment, and evaluation splits contain 52 h, 50 h, and 206 h of data respectively. Energy based voice activity detection (VAD) is applied to these datasets.

3.2. Language Recognition Evaluation (LRE)

The **LRE17** [25] dataset consists of 14 languages and 3 parts: train, dev, and eval. The splits contain 2061 h, 21 h, and 236 h of data, respectively. We filter out the LRE17 dev data to use only audio files shorter than 100 s. On the LRE17 eval data, silence removal is applied on all audios that have length more than 100 s.

The **LRE22** [26] dataset comprises 14 languages from African countries. The dev and test splits contain 30 h and 193 h of audio, respectively. The evaluation focused on developing technologies to improve LID for low-resource languages with an average of 2 h of dev data for each language.

4. Experiments

Our experiments compare the performance of PLDA and the flow model for two different data preprocessing conditions. We either apply or skip length normalization [6] as the last step of our data preprocessing. More specifically, we always follow the subsequent steps for each dataset:

1. Center the data.
2. Reduce the data dimensionality with Linear Discriminant Analysis (LDA) [29, 30].
3. Train and evaluate both PLDA and the flow model.
4. Repeat steps 1-3 but now adding length normalization after LDA in step 2 and compare the results.

[1] https://www.nist.gov/system/files/documents/2016/10/07/sre16_eval_plan_v1.3.pdf

Table 1: Overview of the training datasets used in the front-end for embedding generation. The datasets are used in x-vector system training and XLS-R system fine-tuning for SRE and LRE tasks.

Task	Front-end	Train data
SRE	x-vector	NIST SREs 04, 05, 06, 08, 10, Switchboard [22], Fisher [23] and Mixer 6 [24]
LRE	x-vector	NIST LRE 2009– 2017, Babel [27], NIST SRE16 [28] and Fisher English [23]
	XLS-R	LRE17

Table 2: Equal Error Rate (EER)(%) comparison between PLDA and the flow model for x-vector embeddings on SRE16. The flow shows more robustness than PLDA to the application or not of length normalization.

SRE16		EER (%)	
Embedding type	**Length Norm**	**PLDA**	**Flow**
x-vector	Yes	11.8	**11.5**
x-vector	No	17.1	13.0

4.1. Common Flow Hyperparameters

For all experiments we choose a flow model with $N = 4$ coupling layer transformations. Each coupling layer uses a CNN with a linear transform and 3 1D convolutions, all of them with kernel size 3. The number of output channels of each convolution are 8, 8, and 1 respectively. The ReLU non-linearity is applied between each convolution. We implement the model in Pytorch [31]. Adam optimizer is used with learning rate 0.001 and momentum. We choose a class size of $B = 64$ (Equation 5).

4.2. Methodology for SRE16

For SRE16 the x-vector [15] model is trained with the kaldi SRE16 [32] recipe, with the model architecture extended to have 10 layers of Time delay neural networks (TDNN) [33]. The training data used for x-vector model is given in Table 1. The embeddings are preprocessed to train the PLDA model as mentioned at the beginning of Section 4. We apply the same data preprocessing methodology for our flow model as well. The large unlabeled dev-set is used for centering the evaluation data and the small labeled dev-set is ignored.

In order to choose the number of training epochs, a highly representative labeled dev-set is created by splitting the training data into a new train-set with 80% of the speakers and a dev-set with the remaining 20%. The flow model is then optimized on the Negative Log-Likelihood (NLL) loss using the train-set. We stop training the model when the loss on the dev-set grows or saturates. During evaluation, when a speaker has multiple utterances for enrollment, the embeddings are averaged (see i-vector averaging in [34]).

4.3. Methodology for LRE

For LRE, we use two distinct embedding extractors to demonstrate the generalizabiltiy of our observations: (i) x-vector model from [35], which was trained from scratch with labeled data, and (ii) XLS-R model [16] fine-tuned for language identification. The datasets used for training the two embedding extractors are detailed in Table 1. Note that the XLS-R model only uses LRE17 train data for fine-tuning. Hence, it is not expected to perform as well as the x-vector system, which uses significantly more data.

As mentioned earlier, x-vector embeddings – with a dimension of 256 – are obtained from the system described in [35]. The model uses a 18 layer ResNet [36] architecture and 64 filter bank channels as the acoustic features. The XLS-R embeddings – with a dimension of 1024 - are obtained from the system described in [37]. The XLS-R [16] model is pre-trained following wav2vec 2.0 [38] style self-supervision with 128 languages and 430,000 h of data using fairseq [39]. We fine-tune with the LRE17 train split described in Section 3.2 with a learning rate of 3e-05 for 42’000 steps with a batch size of 8 with the cross-entropy loss using the espresso [40] toolkit following [41]. The Orthonormal Linear layer proposed in [37] is used as the final layer classifier for the XLS-R model. This layer is implemented using [42] and adapted with a learning rate factor of 20.

LDA dimensionality of 13 is chosen given that both LRE17 and LRE22 datasets have only 14 languages. The dev split for each LRE dataset is used to compute the mean of embeddings required for centering. We train the flow model for 10 epochs using the dev splits of each dataset.

4.4. Results

Table 2 shows the EERs obtained on SRE16 for the PLDA and flow models when varying the application of length normalization. The results indicate that the flow model slightly outperforms PLDA when length normalization is applied (11.51% vs 11.83% respectively). However, when length normalization is not applied PLDA performance degrades to 17.12% while the flow shows more robustness with an EER of 13.05%. As already reported by several other studies [6, 43], we note how the application of length normalization strongly boosts PLDA performance for SRE16. We do not claim our flow model is in general better than PLDA for SRE16, as PLDA can offer an excellent EER of 8.5% when used with domain adaptation techniques [32].

It is also revealing to visualize the Detection Error Tradeoff (DET) curves in Figures 2 and 3. When length normalization is applied, both models present very similar performance curves. When it is not applied, we get a clear separation gap between the DET curves of each model, indicating a substantial performance improvement of the flow with respect to PLDA.

When analyzing LRE17 in Table 3, we observe PLDA accuracy on x-vector embeddings drops by 1.4% absolute (from 77.7% to 76.3%) when not applying length normalization. On the contrary, our flow model not only performs better than PLDA for both x-vector and XLS-R systems (improving by 1.3% with length normalization), but also length normalization has no effect on its performance (difference of only 0.1% in accuracy with and without length normalization). The results for LRE22 point towards the same direction: the flow model results are independent to the application of length normalization (constant to 75.7% for x-vector and to 69.2% for XLS-R).

5. Conclusions

We conclude it is possible to combine the theory of PLDA and normalizing flows in order to obtain a non-linear version of PLDA that can be efficiently trained and evaluated for speaker verification and language recognition. While PLDA strongly

Table 3: Accuracy (%) comparison between PLDA and the flow model for x-vector and XLS-R embeddings on LRE17 and LRE22 evaluation sets. The best accuracy is achieved by the flow model when length normalization is not applied. The XLS-R model is fine-tuned with only LRE17 data (see Table 1), thus resulting in a performance drop on the LRE22 setup.

Embedding type	Length normalization	LRE17 Accuracy (%)		LRE22 Accuracy (%)	
		PLDA	Flow	PLDA	Flow
x-vector	Yes	77.7	79.0	73.3	75.7
	No	76.3	**79.1**	73.6	**75.7**
XLS-R	Yes	83.7	84.1	67.0	69.2
	No	83.2	**84.2**	66.9	**69.2**

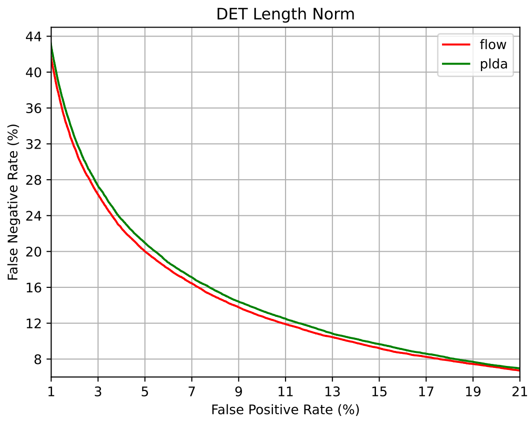

Figure 2: Detection Error Tradeoff (DET) curves for PLDA and the flow model on SRE16 are similar when length normalization is applied in the data preprocessing.

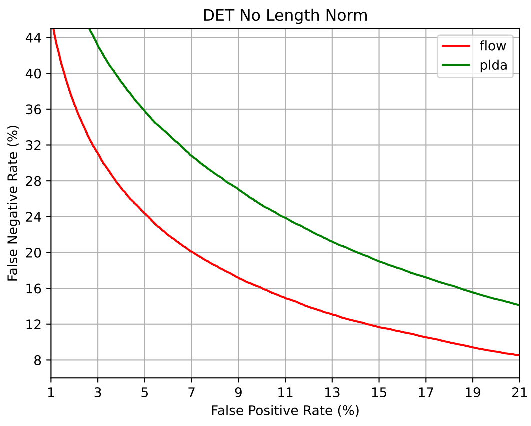

Figure 3: Clear gap between the DET curves of PLDA and the flow model when not applying length normalization.

relies on length normalization to boost its performance, the flow model can offer more robustness when the data distribution varies and it does not match the Gaussian assumption of PLDA. In some problems the flow can even offer the best results without any need for length normalization. We emphasize we do not present the flow model as a replacement of PLDA, as one can obtain excellent results by using PLDA with domain adaptation techniques. We rather present the flow as a non-linear version of PLDA which offers the possibility to compose many invertible non-linear transformations in order to model non-Gaussian data distributions. Theoretically, the flow can be applied to the same variety of tasks as PLDA (classification, clustering, and verification) while keeping desirable properties such as Jacobian cancellation in likelihood ratios and simple Gaussian distributions for latent variables.

6. Future Work

Our model currently relies on PLDA for obtaining a matrix Ψ and fixing it. Our next step will be to derive an optimization method for updating the diagonal matrix Ψ that is independent of PLDA. We will also focus on extending our techniques to flow models from [44, 19] to allow using simple (and cheaper) models for generating the vector embeddings.

7. Acknowledgements

This work was supported by Armasuisse Science and Technology.

8. References

[1] Allan Gut, *An Intermediate Course in Probability*, Springer Publishing Company, Incorporated, 2009.

[2] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker, “Normalizing Flows: An Introduction and Review of Current Methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, 2021.

[3] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan, “Normalizing Flows for Probabilistic Modeling and Inference,” *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[4] Lucas Theis, Aäron van den Oord, and Matthias. Bethge, “A note on the evaluation of generative models,” in *International Conference on Learning Representations*, April 2016.

[5] Sergey Ioffe, “Probabilistic Linear Discriminant Analysis,” in *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 2006, pp. 531–542.

[6] Daniel Garcia-Romero and Carol Y Espy-Wilson, “Analysis of i-vector Length Normalization in Speaker Recognition Systems,” in *12th Annual Conference of the International Speech Communication Association*, 2011.

[7] Pavel Matějka, Ondřej Glembek, Fabio Castaldo, Md Jahangir Alam, Oldřich Plchot, Patrick Kenny, Lukáš Burget, and Jan Černocky, “Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4828–4831.

[8] Patrick Kenny, “Bayesian Speaker Verification with Heavy Tailed Priors,” *Proc. Odyssey 2010*, 2010.

[9] Sandro Cumani and Pietro Laface, “Nonlinear I-Vector Transformations for PLDA-Based Speaker Recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 908–919, 2017.

[10] Sandro Cumani and Pietro Laface, “Joint Estimation of PLDA and Nonlinear Transformations of Speaker Vectors,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 10, pp. 1890–1900, 2017.

[11] Oldřich Plchot, Pavel Matějka, Ondřej Novotný, Sandro Cumani, Alicia Lozano-Diez, Josef Slavíček, Mireia Diez, František Grézl, Ondřej Glembek, Mounika Kamsali, Anna Silnova, Lukáš Burget, Lucas Ondel, Santosh Kesiraju, and Johan Rohdin, “Analysis of BUT-PT Submission for NIST LRE 2017,” in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2018)*, 2018, pp. 47–53.

[12] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2016.

[13] Najim Dehak et al., “Front-End Factor Analysis for Speaker Verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[14] Sandro Cumani and Pietro Laface, “e-vectors: JFA and i-vectors revisited,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5435–5439.

[15] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, “X-Vectors: Robust DNN Embeddings for Speaker Recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.

[16] Arun Babu et al., “XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale,” in *Proc. of Interspeech*, 2022, pp. 2278–2282.

[17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio, “Density estimation using Real NVP,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, Toulon, April 2017.

[18] Yann LeCun, Yoshua Bengio, et al., “Convolutional Networks for Images, Speech, and Time-Series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, pp. 1995, 1995.

[19] Durk P Kingma and Prafulla Dhariwal, “Glow: Generative Flow with Invertible 1x1 Convolutions,” in *Advances in Neural Information Processing Systems*, 2018.

[20] Douglas Reynolds, Elliot Singer, Seyed O Sadjadi, Timothee Kheyrkhah, Audrey Tong, Craig Greenberg, Lisa Mason, Jaime Hernandez-Cordero, and MIT Lincoln Laboratory Lexington United States, “The 2016 NIST Speaker Recognition Evaluation,” 2017.

[21] Karen Jones, Stephanie Strassel, Kevin Walker, David Graff, and Jonathan Wright, “Call My Net Corpus: A Multilingual Corpus for Evaluation of Speaker Recognition Technology,” 08 2017, pp. 2621–2624.

[22] J.J. Godfrey, E.C. Holliman, and J. McDaniel, “SWITCHBOARD: telephone speech corpus for research and development,” in *1992 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1992, vol. 1, pp. 517–520 vol.1.

[23] Christopher Cieri, David Miller, and Kevin Walker, “The fisher corpus: A resource for the next generations of speech-to-text.,” in *LREC*, 2004, vol. 4, pp. 69–71.

[24] Linda Brandschain, David Graff, Chris Cieri, Kevin Walker, Chris Caruso, and Abby Neely, “The Mixer 6 Corpus: Resources for Cross-Channel and Text Independent Speaker Recognition,” in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*, Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, Eds., Valletta, Malta, may 2010, European Language Resources Association (ELRA).

[25] Seyed Omid Sadjadi et al., “The 2017 NIST language recognition evaluation.,” in *Odyssey*, 2018, pp. 82–89.

[26] Yooyoung Lee et al., “NIST 2022 language recognition evaluation plan,” 2022.

[27] Peter Roach, Simon Arnfield, William Barry, Julia Baltova, Marian Boldea, Adrian Fourcin, Wiktor Gonet, Ryszard Gubrynowicz, Elisabeth Hallum, Lori Lamel, et al., “Babel: An eastern european multi-language database,” in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP’96*. IEEE, 1996, vol. 3, pp. 1892–1893.

[28] National Institute of Standards and Technology, “NIST 2016 Speaker Recognition Evaluation Plan,” Accessed 2023, `https://www.nist.gov/system/files/documents/2016/10/07/sre16_eval_plan_v1.3.pdf`.

[29] T. Hastie, R. Tibshirani, and J.H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[30] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak, “Language Recognition via Ivectors and Dimensionality Reduction,” in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., “Pytorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] David Snyder, Daniel Garcia-Romero, Gregory Sell, and Daniel Povey, "Kaldi Recipe for NIST SRE 2016," Accessed 2023, `https://github.com/kaldi-asr/kaldi/blob/master/egs/sre16/v2/run.sh`.

[33] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[34] Srikanth Madikeri, Marc Ferras, Petr Motlicek, and Subhadeep Dey, "Intra-class covariance adaptation in PLDA back-ends for speaker verification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5365–5369.

[35] Igor Szoke, Santosh Kesiraju, Ondrej Novotny, Martin Kocour, Karel Vesely, Jan Cernocky, et al., "Detecting English Speech in the Air Traffic Control Voice Communication," *arXiv preprint arXiv:2104.02332*, 2021.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[37] Amrutha Prasad, Andrés Carofilis, Geoffroy Vanderreydt, Driss Khalil, Srikanth Madikeri, Petr Motlicek, and Christof Schuepbach, "Fine-tuning self-supervised models for language identification using orthonormal constraint," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 11921–11925.

[38] Alexei Baevski et al., "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12449–12460, 2020.

[39] Myle Ott et al., "fairseq: A Fast, Extensible Toolkit for Sequence Modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[40] Yiming Wang et al., "Espresso: A Fast End-to-end Neural Speech Recognition Toolkit," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 136–143.

[41] Apoorv Vyas, Srikanth Madikeri, and Hervé Bourlard, "Comparing CTC and LFMMI for out-of-domain adaptation of wav2vec 2.0 acoustic model," *arXiv preprint arXiv:2104.02558*, 2021.

[42] Srikanth Madikeri et al., "Pkwrap: a Pytorch Package for LF-MMI Training of Acoustic Models," *arXiv preprint arXiv:2010.03466*, 2020.

[43] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md Jahangir Alam, and Pierre Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7649–7653.

[44] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville, "Neural Autoregressive Flows," in *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause, Eds. 10–15 Jul 2018, vol. 80 of *Proceedings of Machine Learning Research*, pp. 2078–2087, PMLR.