Anlirika: an LSTM–CNN Flow Twister for Spoken Language Identification

Andrei Shcherbakov[♯] **Liam Whittle**[◁] **Ritesh Kumar**[•]
Siddharth Singh[•] **Matthew Coleman**[◁] **Ekaterina Vylomova**[♯]
[♯]University of Melbourne [◁]Monash University [•]Bhim Rao Ambedkar University

ultrasparc@yandex.ru

Abstract

The paper presents Anlirika's submission to SIGTYP 2021 Shared Task on Robust Spoken Language Identification. The task aims at building a robust system that generalizes well across different domains and speakers. The training data is limited to a single domain only with predominantly single speaker per language while the validation and test data samples are derived from diverse dataset and multiple speakers. We experiment with a neural system comprising a combination of dense, convolutional, and recurrent layers that are designed to perform better generalization and obtain speaker-invariant representations. We demonstrate that the task in its constrained form (without making use of external data or augmentation the train set with samples from the validation set) is still challenging. Our best system trained on the data augmented with validation samples achieves 29.9% accuracy on the test set.

1 Introduction

Among approximately 7,000 world languages, over 43% are oral only and do not exhibit any writing system. Still, even in less exotic cases language processing systems may have to solely rely on vocal representations. Spoken language identification (SLI) is essential sub-task in many approaches to multilingual automated speech recognition and machine translation. In addition, it also has practical applications as a standalone task. Automated assignment of a call center operator to a client is one of possible use case scenarios.

The paper provides a description of "Anlirika" system[1] that was submitted to SIGTYP 2021 Shared Task on *Robust* SLI (Salesky et al., 2021). In terms of the task, systems are trained to predict a language class (id) from an audio signal. Importantly, the task aims at development of robust systems that can generalize well to new domains and speakers. Many languages are under-resourced, and the situation when the language data exist only for a very limited number of speakers or domains is common. For instance, the largest multilingual SLI dataset, namely CMU Wilderness (Black, 2019), has been derived from the Bible in ≈ 700 languages and lacks speaker diversity. Therefore, it is essential for a system to be speaker-invariant and robust.

2 Related Work

Most work on SLI focused on Indo-European languages such as English, German, Russian, French, Hindi. It is also common to transform raw audio signal into the log-Mel spectra or MFCC features. Recent approaches such as Bartz et al. (2017), Revay and Teschke (2019), and Shukla et al. (2019) make use of various convolution-based neural architectures. For instance, Bartz et al. (2017) proposed a hybrid model that used convolutional layers to extract spatial features and recurrent units (bidirectional LSTMs) tp capture temporal characteristics. Revay and Teschke (2019) explored the ResNet-50 (He et al., 2016) architecture dynamically adapting learning rate.

3 Dataset

The dataset comprises 16 typologically diverse languages from Afro-Asiatic, Austronesian, Basque, Dravidian, Indo-European, Niger-Congo, and Tai-Kadai families. The training data is derived from the CMU Wilderness dataset (Black, 2019) which represents a single domain (speech utterances from the Bible) and has predominantly a single speaker per language. The validation and test sets were collected from multiple corpora such as Common Voice (Ardila et al., 2019) and present a variety of recording conditions with multiple speakers per language. The length of each speech utterance ranged

[1]The code is available at https://github.com/andreas-softwareengineer-pro/speech-language-classifier

Proceedings of the Third Workshop on Computational Typology and Multilingual NLP, pages 145–148

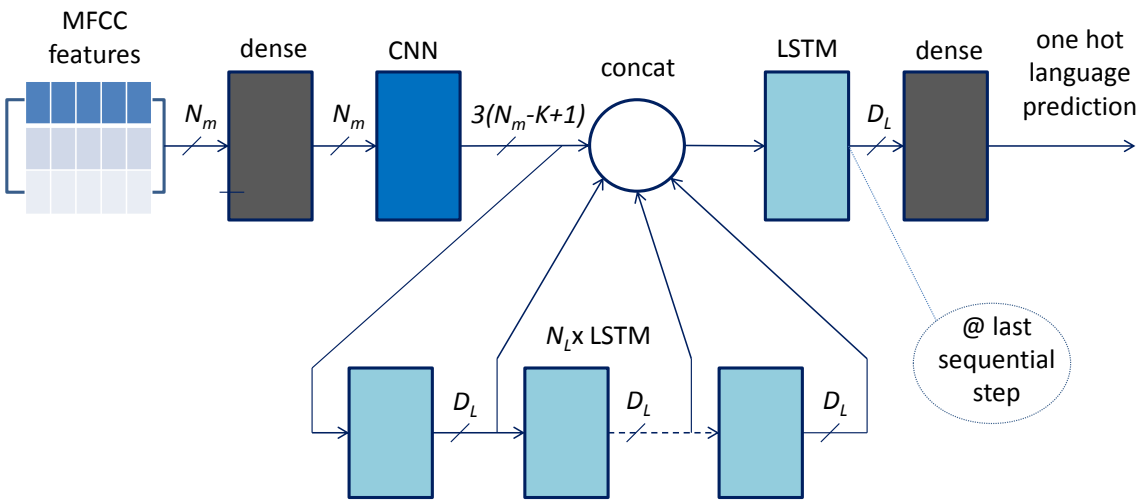

Figure 1: Architecture used in language classifier

between 3 and 7 seconds. The training data contained 4,000 utterances per language, while validation and test sets comprised 500 samples each. Importantly, the utterances were provided in the form of Mel-Frequency Cepstral Coefficients (MFCC) features rather then raw audio signal.

4 Architecture

As illustrated on Figure 1, we used a multi-layer neural network solution with two dense layers, one CNN and 1–7 LSTM layers. The design of neural layer stack is motivated by the following general vision of how a sample should be processed:

- We suggest that a raw spectral pattern first needs to be multiplied by a square matrix in order to remove sound harmonics. That is why we are using a dense layer as the front one;
- Then we try to recognize features related to the spectral line shape. Therefore, we use a one-dimensional CNN (convolving by input feature vector index [frequency]);
- Then we recognize "local" temporal constructs with a stack of LSTMs;
- We use yet another LSTM to reduce temporal patterns into a single-vector representation (only the final time step output goes to the next layer);
- Finally, we classify it into one of 16 languages with a dense layer.

The layer stack we used is summarized in Table 1.

4.1 Batching mechanism

We employed a batched learning process with a fixed number of processed samples per batch (64) and with variable number of time steps. Such a mechanism works as follows. An initial batch is filled with randomly chosen samples. The number of temporal steps in the batch is determined by the shortest sample currently present in a given batch. Once a batch is fed forward through the neural network layers:

1. The samples which ends happen to be aligned with the end of the batch, are done now (within a given epoch). We replace them with next randomly chosen training samples when forming the next batch. If a sequential layer contains hidden states (which is true for the LSTMs in our model), zero hidden states are supplied to the respective threads of batch. Final prediction values for such threads are used to calculate the loss;
2. The samples which do not fit within the batch length, are passed to the next batch for further processing, having their already-processed prefixes removed. Start hidden states for the respective threads are initialized with the values of final hidden states computed in the preceding batch, as shown with blue arrows in Figure 2.

This process repeats until all the samples are pro-

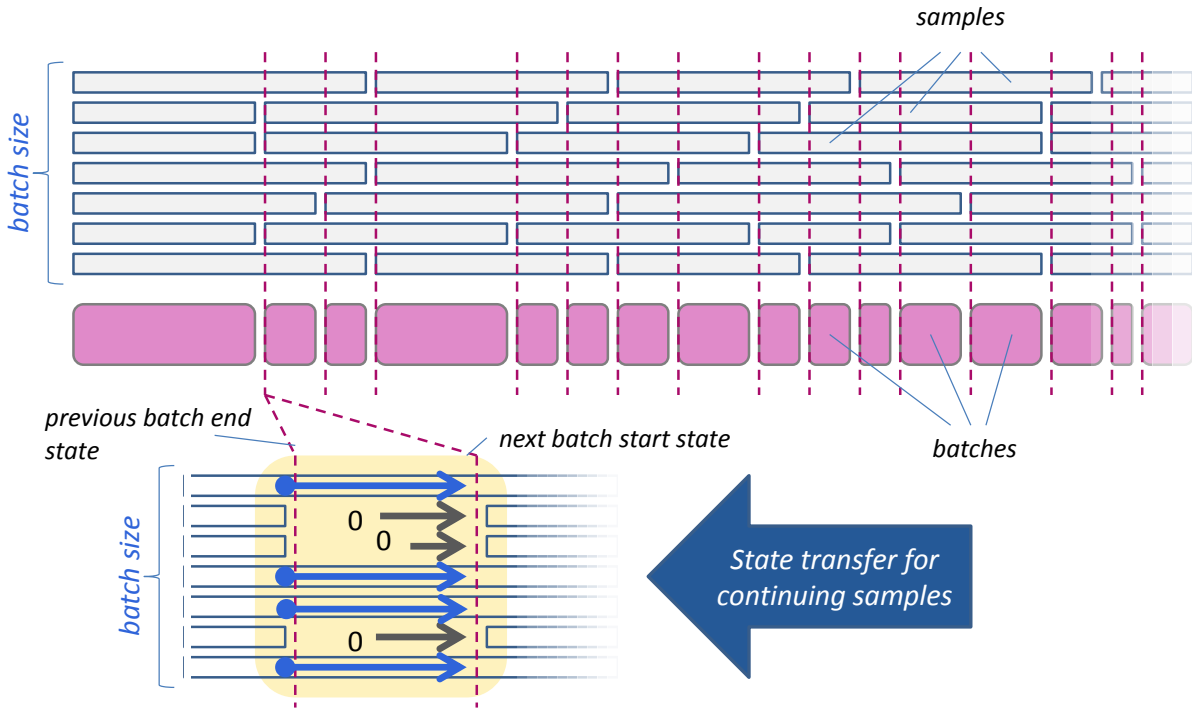

Figure 2: Batch shaping

Layer Type	Output Size	Output Type	Hyperparamers
dense	N_m	per-timestep	$N_m = 39$
1D CNN	$D_{CNN} = 3(N_m - K + 1)$	per-timestep	K=4 – kernel size
$N_L\times$ LSTM	D_L ea.	per-timestep	-
concat	$D_{CNN} + N_L D_L$	per-timestep	-
LSTM	D_L	per-sample	-
dense	Num. languages=16	per-sample	-

Table 1: Layer stack summary

cessed, i.e. a training epoch is done. Some trailing batches may be underpopulated with threads, in which cases output values of unused threads are ignored. Figure 2 summarises the description above.

A drawback of such a batching technique is constraining of temporal depth when the backpropagation through time takes place in LSTM layers. A batch is typically much shorter in time steps than a sample. Therefore, a single backpropagation operation (that cannot run across batches) may modify less weights than it would be expected without batching. We regarded this effect as minor, however, its influence to the overall learning capability is yet to be investigated.

5 Experiments

We varied N_L, the number of extra sequential LSTM layers (which outputs were concatenated to the output of the CNN layer). We tried the following options: {0,2,4,6}. A number of units in each LSTM layer was chosen from {200,300}. We used equal numbers of units across all the LSTM layers present in the model.

Using the original train set. A learning dynamic we observed in our experiment was generally slow. In most trials the model failed to learn with the learning rate value greater than $4 \cdot 10^{-4}$. With lower learning rates, it trained at an extremely slow pace gaining about 0.1% train set accuracy per epoch. At the time of this report writing, we achieved an overall accuracy value of about 12% at validation set. It is curious to note that accuracy figures for train and validation sets did not correlate as expected, the fact that may indicate significant difference in domains. Typically, a predicted distribution of languages was limited to 2-3 classes,

the list of which was volatile. We also noticed that the model converged much faster at small subsets of training sets (50-500 samples).

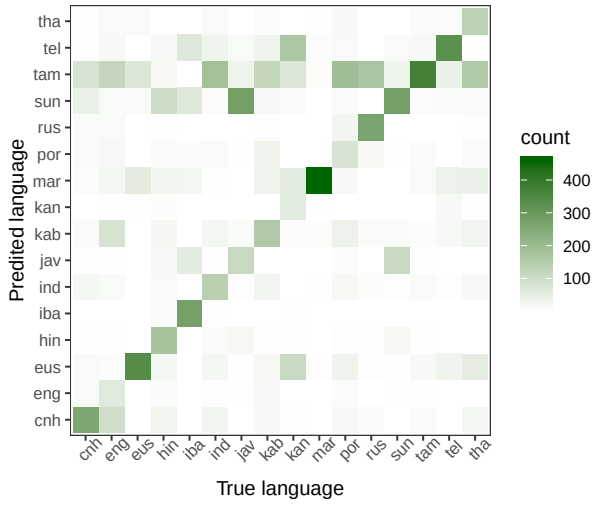

Figure 3: Confusion matrix for mixed set holdout validation ($N_L = 2, D_L = 200$)

Augmenting training data with validation set samples. A quite different picture was observed when we combined training and validations sets and randomly split them again into training and validation portions. A much superior accuracy of 74% on validation set was achieved. The confusion matrix is shown on Figure 3. Such a relatively high prediction accuracy is not surprising, as a validation holdout is likely to share speaker identities with the respective training subset, the fact that leads to a significant loosening of required generalization ability. However, a drastic improvement in convergence dynamics remains a noticeable and unexpected effect.

Tuning of hyperparameters. A choice of$N_L = 2$ was found to be producing the highest accuracy. Increasing D_L from 200 to 300 didn't lead to any significant difference in performance.

Shared task submission. The final submitted version was trained on an augmented set. The performance figures are shown in Table 2.

Set	Acc.	F1, Micro Avg	F1, Macro Avg
Test	29.9%	29.8%	28.2%
Valid.	43.6%	43.6%	42.1%

Table 2: Aggregated performance metrics for the final model version

6 Conclusion & future work

To address the task of language classification in speech samples, we implemented and explored a neural network model. The model's architecture was inspired by an idea of phoneme sequence recognition. Our experiments are yet in progress, still it is clear that the generalization across domains appears to be the main challenge.

Following a maxim of keeping model as light as possible, we are going to explore architecture modifications that directly enforce some kind of phonetic generalization, for instance, by insertion of "bottlenecks" (layers with low output size).

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. 2019. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*.

Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. 2017. Language identification using deep convolutional recurrent neural networks. In *International conference on neural information processing*, pages 880–889. Springer.

Alan W Black. 2019. Cmu wilderness multilingual speech dataset. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5971–5975. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Shauna Revay and Matthew Teschke. 2019. Multi-class language identification using deep learning on spectral images of audio signals. *arXiv preprint arXiv:1905.04348*.

Elizabeth Salesky, Badr M Abdullah, Sabrina J Mielke, Elena Klyachko, Oleg Serikov, Edoardo Maria Ponti, Ritesh Kumar, Ryan Cotterell, and Ekaterina Vylomova. 2021. SIGTYP 2021 shared task: Robust spoken language identification. In *Proceedings of the Third Workshop on Computational Research in Linguistic Typology*, pages 136–142.

Shikhar Shukla, Govind Mittal, et al. 2019. Spoken language identification using convnets. In *European Conference on Ambient Intelligence*, pages 252–265. Springer.