Towards Building ASR Systems for the Next Billion Users

Tahir Javed[1,2], Sumanth Doddapaneni[2,4], Abhigyan Raman[2], Kaushal Santosh Bhogale[2], Gowtham Ramesh[2,4], Anoop Kunchukuttan[2,3], Pratyush Kumar[2,3], Mitesh M. Khapra[1,2,4]

[1]IIT Madras,
[2]AI4Bharat,
[3]Microsoft,
[4]RBCDSAI

{tahirjmakhdoomi, dsumanth17, ramanabhigyan, kaushal98b, gowtham.ramesh1}@gmail.com
{ankunchu, kumar.pratyush}@microsoft.com
miteshk@cse.iitm.ac.in

Abstract

Recent methods in speech and language technology pretrain very large models which are fine-tuned for specific tasks. However, the benefits of such large models are often limited to a few resource rich languages of the world. In this work, we make multiple contributions towards building ASR systems for low resource languages from the Indian subcontinent. First, we curate 17,000 hours of raw speech data for 40 Indian languages from a wide variety of domains including education, news, technology, and finance. Second, using this raw speech data we pretrain several variants of wav2vec style models for 40 Indian languages. Third, we analyze the pretrained models to find key features: codebook vectors of similar sounding phonemes are shared across languages, representations across layers are discriminative of the language family, and attention heads often pay attention within small local windows. Fourth, we fine-tune this model for downstream ASR for 9 languages and obtain state-of-the-art results on 3 public datasets, including on very low-resource languages such as Sinhala and Nepali. Our work establishes that multilingual pretraining is an effective strategy for building ASR systems for the linguistically diverse speakers of the Indian subcontinent.

Introduction

The Indian subcontinent is one of the most linguistically diverse regions in the world as well as one of the most populous regions in the world - little wonder that is called a 'subcontinent'. It is home to around 2 billion people from 7 countries who speak 100+ major languages (and thousands of minority languages and dialects) belonging to four major language families[1,2]. Of these the Indo-Aryan and Dravidian languages are spoken by a large section of the population. These language families have intermingled over a large period of time giving rise to the Indian linguistic area/sprachbund where languages across these families share many features (Emeneau 1956; Subbārāo 2012; Kunchukuttan and Bhattacharyya 2020).

Building high-quality ASR models for such a large and diverse pool of languages is challenging, even if we restrict ourselves to 30 odd languages which have more than a million speakers. Many modern ASR models rely on large amounts of labeled data for each language to build high-quality ASR systems. Such approaches are expensive and not scalable, thus limiting the reach of ASR technologies to some languages and a section of the population. In addition to these challenges on availability of labeled data, Indian languages also face a set of common challenges that need to be addressed. Collection on unlabeled data for pretraining can be undertaken as a joint effort since many sources might be shared amongst these languages. Most Indic scripts have a larger character set than English which can be addressed in a uniform way. The complex inflectional systems of Indian languages would make modelling phonotactics more challenging. The rich inflectional/agglutinative nature of Indian languages result in larger vocabulary sizes, presenting challenges to incorporating language models. At the same time there are opportunities from a unified perspective. A largely overlapping phoneme inventory, logically overlapping character sets and syntactic similarity can help utilize linguistic similarity at various levels to build multilingual models where transfer learning can be effective.

In this context, it is important to take note of recent work that has demonstrated the benefits of unsupervised pretraining and multilingual fine-tuning to significantly improve ASR quality for low-resource languages (Baevski et al. 2020; Conneau et al. 2020a). In particular, the wav2vec model has established two key results for English ASR. One, an end-to-end DNN architecture borrowing the popular Transformer architecture from NLP establishes SOTA results. And two, pretraining on a large corpus of data reduces the requirement for labeled fine-tuning from hundreds of hours to few hours and to even tens of minutes. It is worthwhile to explore if these findings from English ASR transfer to Indic ASR, especially given the diversity and above mentioned challenges with Indic languages. More specifically, would a wav2vec-like model establish SOTA results on available benchmarks across Indic languages? Further, would large pretraining preclude the need for collecting large amounts of labeled data? And finally, would pretraining a multilingual model across Indian languages provide positive benefits across these related languages? In order to answer these questions, we make the following contribu-

[1]https://en.wikipedia.org/wiki/Languages_of_South_Asia

[2]https://en.wikipedia.org/wiki/Languages_of_India

tions:

• We curate 17,314 hours of raw audio data for pretraining across 40 languages from 4 language families, making it one of the largest and most diverse collections of Indic language data. This data has been curated in a short time frame from public sources which have a permissive license. It indicates that the feasibility of collecting large amount of pretraining data and further efforts can be made to significantly expand this collection.

• Starting from the wav2vec 2.0 model, we perform extensive ablation studies on architecture choices, pretraining and fine-tuning strategies, language models and choice of lexicon to arrive at a training and decoding regimen that works well for Indian languages.

• Our ASR models achieve SOTA performance on 9 Indian languages on 3 publicly available benchmarks with small fine-tuning datasets. These results indicate that end-to-end ASR systems based on multilingual pretraining with the wav2vec model hold promise for Indic languages.

• Our ablation studies reveal that the accuracy of the ASR system on Indic languages sensitively depends on size of the pretraining corpus, amount of labelled data for fine-tuning, and access to task-specific lexicon.

In summary, we establish that the recent advances of pretraining wav2vec models transfer to Indic ASR and achieve SOTA results against models proposed over multiple years. However, unlike in the reported results of English ASR, we observe that the WER reported for Indic ASR is significantly higher and sensitively depends on availability of resources: pretraining corpus, fine-tuning data, and task-specific language information. This suggests that the ASR task on Indic languages remains far from being solved and requires model innovation and continued efforts on curating resources. We have publicly released all the artifacts of our work[3] to spur further work in the area of Indic ASR. This includes: (a) sources of pretraining data along scripts for their collection and pre-processing, (b) pretraining, fine-tuning and decoding scripts, and (c) our best ASR models.

Curating Speech Data for Indian Languages

As mentioned earlier, existing wav2vec models for English (Baevski et al. 2020) and some other languages (Conneau et al. 2020a; Wang et al. 2021a) have been trained using 1000s of hours of curated raw audio data in these languages. However, for Indian languages there is no such publicly available repository of audio data. Hence, as our first task, we curate audio data from online sources such as YouTube.

We hired native speakers in each of the 22 constitutionally recognised Indian languages to help us with this task. These workers were employees of a data collection agency which specializes in creating and curating text and speech data in Indian languages. In particular, each worker had prior experience of working on similar tasks. The workers were asked to discover channels, playlists or individual videos on YouTube by using a wide variety of keywords from different domains such as education, news, technology, sports and finance. Once the content was discovered, they were asked to ensure that the audio was clean, *i.e.*, it did not contain any background music and the content was predominantly in the target language. Note that some amount of code-mixing with English is unavoidable in Indian speech content but the workers manually checked that the audio was largely in the target language. The workers were also instructed to ensure speaker diversity by not taking a large amount of content from a single channel. Lastly, the workers were asked to ensure that the video was available under a Creative Commons License. This was important to ensure that we can freely share the URLs with the research community, ensuring reproducibility of our work although this significantly limited the amount of data we could collect, especially in the low resource Indian languages. For each video which cleared the above checks, the workers were asked to copy the URL in an excel sheet and mention the target language.

Language	NoA	YT	Language	NoA	YT
assamese	667	176	**ladakhi**	121	0
balti	7	0	**lepcha**	52	0
bengali	740	295	**maithili**	37	1
bhojpuri	62	0	**malayalam**	625	232
bhutia	42	0	**manipuri**	374	90
bodo	53	11	**marathi**	387	667
chhatisgar	73	0	**mizo**	304	0
dogri	606	8	**nagamese**	210	0
garo	63	0	**nepali**	374	333
gojri	197	0	**odia**	716	302
gujarati	375	686	**pahari**	24	0
hindi	353	722	**punjabi**	562	301
english	408	0	**rajasthani**	134	0
jaintia	57	0	**sambalpuri**	192	0
kannada	425	587	**sanskrit**	152	348
karbi	56	0	**santali**	0	9
kashmiri	417	19	**sindhi**	77	30
khasi	85	0	**tamil**	815	197
kokborok	185	0	**telugu**	395	657
konkani	471	28	**urdu**	359	362

Table 1: Number of hours audio data per language after applying the preprocessing steps. Here NoA refers to `newsonair` data, YT refers to YouTube data.

Apart from YouTube, we also curated content from `newsonair`[4] which is a radio news channel run by the Govt. of India and broadcasts news in multiple Indian languages. We did not need any of the above manual checks for `newsonair` as being a national news channel, the content there was already clean (e.g., the language of the audio was clearly mentioned and the audio files were largely free of any background music). We crawled their website to get the URLs of audio files in 40 Indian languages.

Once the URLs were available, we used the youtube-dl library[5] to download the videos from YouTube. We then used the FFmpeg library[6] to extract the audio data from the videos. For `newsonair`, these two steps were not needed

[3] https://github.com/AI4Bharat/IndicWav2Vec

[4] http://newsonair.com/

[5] https://github.com/tpikonen/youtube-dl

[6] https://ffmpeg.org/

as we could simply download the audio files from the URLs. We noted that existing speech datasets typically contain mono channel audio data sampled at a sampling frequency of 16kHz. However, in our case, since the data was curated from diverse sources, some of the content was not mono channel and the sampling frequency varied from 8kHz to 44 kHz. To standardise the data, we once again used the FFmpeg library to (i) upsample (downsample) the data which was originally sampled at a frequency lesser (greater) than 16 kHz and (ii) reduce the number of audio channels to 1. We further refined the data by removing long silences in the audio clips using the py-webrtcvad[7] library which is a python interface to the popular WebRTC VAD (Voice Activity Detection) module developed by Google[8]. The VAD algorithm filters out non-speech content and allows us to set an aggressiveness parameter, which is an integer between 0 and 3 (0 is the least aggressive about filtering out non-speech, 3 is the most aggressive). We found that setting this parameter to 2 worked best for our data. Next, to avoid including highly noisy content in our data, we used Waveform Amplitude Distribution Analysis (WADA-SNR) (Kim and Stern 2008) and filtered out audio clips which had a signal-to-noise-ratio (SNR) less than 15 dB. This threshold was chosen after experimenting with a subset of the audio data. Finally, as a standard practice, we chunked audio files so that the maximum duration of any audio file was only 25 secs.

The final statistics of the data thus obtained in each language are summarised in Table 1. The uncompressed size of the data is 1.5 TB. The URLs of all the audio files as well as the scripts used for downloading, standardising and cleaning the data have been made publicly available on our GitHub repository.

IndicWav2Vec: A Multilingual ASR Model for Indian Languages

In this section, we describe the methodology followed for pretraining a Wav2Vec ASR model for Indian languages using the raw audio data described in the previous section. In particular, we describe (i) the model architecture, (ii) the pretraining procedure, and (iii) the fine-tuning and decoding procedure. Note that, for all of the above, we closely follow the procedure described in (Baevski et al. 2020) for building an English ASR system. However, we provide the details here for the sake of completeness and reproducibility.

Model Architecture

We use the same architecture as wav2vec 2.0 which consists of (i) a feature encoder for encoding raw audio into a sequence of T latent representations, (ii) a transformer for learning contextualised representations for each of the T units, and (iii) a quantizer for discretizing the representations learnt by the feature encoder to obtain the targets for self-supervised learning. The feature encoder is a multilayered convolutional neural network which takes raw audio signal as input and outputs a sequence of frames $\mathcal{Z} = \{\mathbf{z}_1, \mathbf{z_2}, \ldots, \mathbf{z}_T\}$ where T is the number of timesteps and $\mathbf{z}_i \in \mathbb{R}^d$. The output $\mathcal{Z}$ of the encoder is fed to a transformer based context network which computes a contextualised representation for each of the T units in the input as $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_T\}$, where $\mathbf{c}_i$ is the contextual representation for the i-th input (i.e., $\mathbf{z}_i$). Following standard practice, we augment the $\mathbf{z}_i$'s with positional information before feeding them to the context network using a convolutional layer similar to that used in (Baevski et al. 2020; Mohamed, Okhonko, and Zettlemoyer 2020; Baevski, Auli, and Mohamed 2020; Wu et al. 2019). The final component of the architecture is a quantizer which takes $\mathcal{Z}$ as input and discretizes it to a finite set of representations using product quantization (Jégou, Douze, and Schmid 2011). The output of the quantizer is a sequence of representations $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_T\}$ where $\mathbf{q}_i$ is the contextual representation for the i-th input (i.e., $\mathbf{z}_i$).

Pretraining Objectives and Data Sampling

To train the model, we mask a certain fraction of the T input representations before feeding them to the context network. However the inputs to the quantizer are not masked since they serve as the target for the self-supervised objective. This objective is implemented as minimizing a contrastive loss (Graves et al. 2006) that involves maximizing the similarity of the context network's output to the quantized representation for masked input at time t, while minimizing the similarity with quantized representations of other masked inputs. In addition to the above contrastive loss, we also use a diversity loss (Baevski et al. 2020) which ensures better utilisation of the codebook.

Data Sampling. While pretraining multilingual models involving a large number of languages with different amounts of training data in each language, it is important to ensure that all languages get a fair representation in the training. In particular, languages which account for a larger share of the training data should not dominate the training. We use temperature based sampling (Arivazhagan et al. 2019; Conneau et al. 2020a; Wang et al. 2021b) to form multilingual batches (Devlin et al. 2019; Conneau and Lample 2019) during pretraining. Let L be the set of all languages used for pretraining. For each language $l \in L$, we draw audio samples from a multinomial distribution $p_l \sim \left(\frac{n_l}{N}\right)^{\alpha}$ where n_l is the number of hours of pretraining data in language l, N is the total number of hours of pretraining data across all languages, and $\alpha \in [0, 1]$. Smaller the value of α, more is the upsampling for the lower resource languages.

Fine-tuning

Once the model is pretrained, we fine-tune it for ASR using task-specific supervised data. To do so, we add a randomly initialised projection layer on top of the context network. This layer maps each d dimensional output of the context network, into a C dimensional output where C is the size of the vocabulary. We then use the softmax function to compute a distribution over the vocabulary. In our case, the vocabulary contains all the unique characters in the language. The model can either be jointly fine-tuned using data from all the

[7] https://github.com/wiseman/py-webrtcvad

[8] https://webrtc.org/

languages or individually fine-tuned for a given a specific language. We tried both and for multilingual fine-tuning, we converted the text data into a single script, i.e., the Devanagari script. Though each of these scripts have their own Unicode codepoint range, it is possible to get a 1-1 mapping between characters in these different scripts since the Unicode standard accounts for the similarities between these scripts (Kunchukuttan 2020). This results in a smaller, compact output vocabulary while enabling better transfer across languages (Shetty and Umesh 2021). We fine-tune the model using the standard CTC loss function (Graves et al. 2006; Baevski et al. 2020; Baevski, Auli, and Mohamed 2020). Following (Baevski et al. 2020), we augment data with a modified version of SpecAugment (Park et al. 2019).

Decoding

To decode the emissions from the softmax layer, we use a lexicon and a separately trained word-level n-gram language model. Let $\mathbf{y} = \{y_1, y_2, \ldots, y_M\}$ be a candidate sequence. Further, let $p_{AM}(\mathbf{y})$ and $p_{LM}(\mathbf{y})$ be the probability assigned to the sequence $\mathbf{y}$ by the fine-tuned network and the language model respectively. We select the optimal word sequence $\mathbf{y}^*$ as follows:

$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}} \log p_{AM}(\mathbf{y}) + \alpha \log p_{LM}(\mathbf{y}) + \beta|\mathbf{y}| \quad (1)$$

where $|\mathbf{y}|$ is the length of the sequence and α and β are hyperparameters. We use an efficient beam search decoder (Pratap et al. 2019; Liptchinsky, Synnaeve, and Collobert 2019) to search candidates while combining network scores, LM scores and word insertion bonuses.

Experimental Setup

In this section we describe (i) the datasets used for our experiments, (ii) the hyperparameters used for pretraining and fine-tuning, and (iii) the corpora and hyperparameters used for training the language models.

Datasets We experiment with 3 ASR datasets covering 9 Indian languages. These include the MSR (Microsoft Research) dataset which was released as a part of the *Low Resource Speech Recognition Challenge for Indian Languages* (Srivastava et al. 2018), the MUCS2021 dataset which was released as a part of the *Multilingual and code-switching ASR challenges for low resource Indian languages* (Diwan et al. 2021), and a subset of the OpenSLR dataset (Kjartansson et al. 2018) obtained from the authors of Shetty and Umesh (2021).

Pretraining Setup We pretrain two variants of the model, viz., BASE and LARGE. Similar to the base model in Baevski et al. (2020), our BASE model contains 7 convolutional layers each with 512 channels, strides of (5,2,2,2,2,2,2) and kernel widths (10,3,3,3,3,2,2). The BASE model has 12 transformer blocks with model dimension 768, and FFN dimension 3072 with 8 attention heads. Similarly, same as Baevski et al. (2020), our LARGE model shares the same CNN encoder as the BASE model and differs only in the transformer architecture. The LARGE model has 24 transformer blocks with model dimension 1024 and FFN dimension 4096 with 16 attention heads. For both the architectures, we use $G = 2$ (codebooks) with $V = 320$ entries per codebook in the quantization module. Instead of pretraining the models from scratch we start with the pretrained checkpoint of the corresponding (BASE or LARGE) English wav2vec2.0 model[9]. We then further pretrain the model using the data we curated for Indian languages. As mentioned earlier, to account for the skew in the amount of data across different languages, we used temperature based sampling. We experiment with three different values of $\alpha \in \{0.5, 0.7, 1\}$ and found that 0.7 works best in terms of model accuracy in distinguishing the correct masked unit from the distractor units.

For the BASE model, we crop audio segments to 250k samples or 15.6 seconds of audio. We restrict the max tokens per GPU to 3M and train the model on 24 A100 GPUs with gradient accumulation of 2 steps making the effective batch size as 3.2 hours. We used Adam optimizer with learning rate set to 0.0005 and decayed the learning rate polynomially after a warmup for 32k steps. We trained the model for 160k steps. For the LARGE model we crop audio segments to 320k samples or 20 seconds of audio. We restrict the max tokens per GPU to 1.2M and train the model on 24 A100 GPUs with gradient accumulation of 6 steps making the effective batch size as 3 hours. We again used the Adam optimizer but with learning rate set to 0.005 and and decayed the learning rate polynomially without any warmup. We trained the model for 110k steps. We set the random seed to 1 for all experiments. All other hyper-parameters are set to the default values from the original wav2vec2.0 code base. Our BASE and LARGE models have 95M and 317M parameters respectively. Our models are implemented in fairseq (Ott et al. 2019) and we use fp16[10] operations to speed up training and reduce memory usage.

Fine-tuning Setup During fine-tuning, we update all the parameters of the network except the parameters of the convolutional feature encoder. For both, the base and the large model, we used Adam optimiser with a learning rate of 1e-4 and tri-stage learning rate schedule: the learning rate is warmed up for first 10% of the steps, then held constant for the the next 40% steps and finally exponentially decayed for the remaining steps. We set the maximum number of frames per GPU to 1M and fine-tune the models on 8 A100 GPUs without any gradient accumulation, making our effective batch size as 8M samples or 500 secs. We trained the BASE model for 80k steps and the LARGE model for 120k steps. For the first 200 steps, we only update the parameters in the final layer and then update all the parameters in the model (except those in the feature encoder). As a data augmentation, strategy, we mask the outputs of the feature encoder, similar to SpecAugment (Park et al. 2019). We set the masking probability to 0.05 and the LayerDrop rate as 0.1 for both the models. For the other hyper-parameters, default values were taken from the wav2vec2.0 code repository. Further we use early stopping with the patience set to 12 epochs. We used fairseq (Ott et al. 2019) for our fine-

[9] https://github.com/pytorch/fairseq/tree/master/examples/wav2vec#pre-trained-models

[10] https://github.com/NVIDIA/apex

tuning experiments.

Language Model Setup As mentioned earlier, we use a lexicon-based beam search decoder as implemented in Flashlight[11] library along with a language model to obtain predictions. For each language, we train 6-gram KenLM language model(s) (Heafield 2011) with pruning for singletons of order 5 and doubletons of order 6. We consider three options for the corpus used for training the language model: (i) the IndicCorp corpus (Kakwani et al. 2020) which contains 836M, 1,860M, 551M, 719M, 582M, 674M, 107M tokens in Bengali, Hindi, Marathi, Gujarati, Tamil, Telugu and Odia, respectively, (ii) the transcriptions available in the respective ASR datasets for each language, and (iii) a combination of the earlier two options. Similarly, we considered three choices for the lexicon: (i) using the top 180K most frequent words from IndicCorp, and (ii) using the task-specific lexicon provided by the challenge organizers and (iii) combination of above two. Later, in section 2, we will show that the choice of lexicon and training corpus have a significant impact on the performance of the ASR system. We tune hyperparameters α and β in *eq.1* using grid search with values ranging from -4 to +4 and 0 to 5 for α and β, respectively. During this grid search, we use a beam size of 64. Note that α and β are tuned independently for each language. Once the best value of α and β are found, we use these for the test data and increase the beam size to 1024.

All the models developed as a part of this work, *viz.*, the pretrained model, the language-specific fine-tuned models and the language models along with the Fairseq and KenLM scripts and configuration files used for building them have been publicly released. We hope that these models will help in advancing the state of the art for Indian Speech Technology.

Results and Discussions

Ablation Studies on Pretraining

We first discuss the effect of pretraining as well as various choices made during pretraining.

No Pretraining In row 1 of Table 2, we present the results for an ASR model which has the same architecture as our base model but was trained from scratch (i.e., the weights were not initialised using pretraining). Comparing this to row 2, which is a model with the same architecture but is fine-tuned after pretraining, we see that pretraining significantly improves the performance. These results are consistent with similar results reported for English models in speech (Baevski et al. 2020) as well as multilingual models in NLP (Conneau and Lample 2019). In a separate set of experiments (not included in Table 2) we found that pretraining is even more important when the model size is large. For example, training the large model from scratch was highly unstable and even after 180K steps, the WER on the validation set was significantly higher as compared to a large model which was initialised using pretraining (82.18 v/s 24.62).

Role of Pretraining Corpus Gupta et al. (2021) released a pretrained wave2vec model for 23 Indian languages using 9800 hours of speech data. In contrast, our model uses over 17,000 hours of pretraining data for 40 languages, with at least 500 hours of data in 15 languages. Comparing rows M2 and M3 of Table 2, we observe that our model trained on larger and more diverse data consistently improves over the model reported in Gupta et al. (2021). In particular, we get an average improvement of 1.44 WER across all languages with a maximum improvement of 6.75 WER for Sinhala and a minimum improvement of 0.09 WER for Hindi. The improvements on Sinhala are noteworthy given that Sinhala is not present in the pretraining data of both the models. We believe that since our pretraining data has more diversity (40 languages as opposed to 23) and a better distribution of pretraining data across languages, it generalises better for languages not seen during pretraining. This is also evident given that our model performs better on Hindi despite a much smaller amount of Hindi pretraining data in our model (1075 v/s 4564 hours). These results establish importance of the larger pretraining dataset released as a part of this work.

Effect of Model Size Several works (Conneau et al. 2020b; Baevski et al. 2020) have shown that large models pretrained on large amount of pretraining data outperform smaller models in downstream tasks. However, in the context of speech models for Indian languages, the right thresholds for data size and model size are not known. For example, with 16000 hours of pretraining data would a large model (317M parameters) outperform a base model (95M parameters). To answer this question we compare rows 3 and 4 of Table 2 which only differ in the model size. We observe that the large model consistently outperforms the base model with an average improvement of 2.08 WER across all languages, a maximum improvement of 7.22 WER for Marathi and a minimum improvement of 1.9 WER for Hindi.

Effect of Model Initialization As mentioned earlier, for pretraining our models we started with the pretrained English models released by (Baevski et al. 2020). This is in line with similar efforts in multilingual NLP where models pretrained for a larger set of languages are initialised with weights from a model pretrained for a smaller set of languages (Tang et al. 2020). To assess if this is needed for Indian speech models, we pretrained a large model from scratch instead of initialising with the weights from the English model. We found this to be very expensive as even after 120k steps (which corresponds to 95 hours of training on 24 GPUs) this model had a loss of 2.17 on the validation data as opposed to the english-initialised model which had a loss of 1.82 after just 110k steps. This is an important finding as it suggests that as we expand to more low resource languages (beyond the 40 covered in our model), we should be able to leverage the pretrained checkpoints of our model and save costs on expensive GPU resources. This may seem trivial but is extremely important in the context of low resource languages such languages typically have scarce computational resources.

Ablation Studies on Fine-tuning and Decoding

The previous section focused on discovering the right choices for pretraining. In this section, we focus on different choices for fine-tuning and decoding.

Choice of Language Model As mentioned earlier, language

[11] https://github.com/flashlight/flashlight

	MSR			MUCS						OpenSLR		
	gu	ta	te	gu	hi	mr	or	ta	te	bn	ne	si
M0: No pretraining	46.0	37.5	35.5	53.2	48.1	87.1	73.4	44.8	44.7	36.0	78.8	37.0
M1: IndicW2V$_b$ (EkStep data)	23.4	24.0	25.8	30.3	18.0	26.5	28.7	29.9	33.2	19.7	14.4	31.0
M2: IndicW2V$_b$ (our data)	22.8	23.7	24.9	29.4	17.8	24.3	27.2	29.3	31.9	18.1	13.8	24.3
M3: IndicW2V$_l$ (our data)	20.5	22.1	22.9	26.2	16.0	19.3	25.6	27.3	29.3	16.6	11.9	24.8
M4: + LM$_{small}$	16.6	14.9	14.4	18.0	16.3	14.8	19.0	25.4	22.4	14.3	13.0	**18.6**
M5: + LM$_{large}$	**11.7**	**13.6**	**11.0**	17.2	14.7	13.8	**17.2**	25.0	20.5	13.6	13.6	-
M6: + augmented lexicon	12.3	15.1	12.4	**14.8**	**10.5**	**12.2**	21.9	**20.0**	**15.2**	**10.6**	**9.7**	-
M7: + Multi. FT	16.7	22.0	18.5	15.2	12.6	20.7	27.4	21.3	15.6	-	-	-

Table 2: Comparison of different choices for pretraining, fine-tuning, and decoding. IndicW2V$_b$ and IndicW2V$_l$ refer to our base and large models respectively. LM$_{small}$ refers to the language model trained using transcripts from the training and validation data and LM$_{large}$ refers to the one trained using IndicCorp in addition to the transcripts.

	hi	gu	mr	or	ta	te
Baseline	27.45	25.98	20.41	31.28	35.82	29.35
CSTR	14.33	20.59	15.79	25.34	23.16	21.88
BSA	16.59	21.30	15.65	17.81	28.59	25.37
EM	17.54	20.11	20.15	19.99	28.52	26.08
EkStep	12.24	30.65	39.74	27.10	27.20	22.43
Uniphore	22.79	22.79	14.9	29.55	**18.8**	28.69
Lottery	17.81	23.62	58.78	17.74	30.69	27.67
IIITH	31.11	26.94	33.8	37.19	35.03	17.00
M5:	14.7	17.2	13.8	**17.2**	25.0	20.5
M6:	**10.5**	**14.8**	**12.2**	21.9	20.0	**15.2**

Table 3: Comparison of our best models (M5, M6) with the the top performers from the MUCS 2021 leaderboard. Individual best models from the leaderboard are underlined.

	gu	ta	te
Baseline (Srivastava et al. 2018)	19.8	19.4	22.6
Jilebi (Pulugundla et al. 2018)	14.0	13.9	14.7
Cogknit (Fathima et al. 2018)	17.7	16.0	17.1
CSALT-LEAP (Srivastava et al. 2018)	-	16.3	17.6
ISI-Billa (Billa 2018)	19.3	19.6	20.9
MTL-SOL (Sailor and Hain 2020)	18.4	16.3	18.6
Reed (Sen et al. 2021)	16.1	19.9	20.2
CNN + Context temporal features (Sen et al. 2020)	18.4	24.3	25.2
EkStep model*	19.5	22.1	21.9
M5:	**11.7**	**13.6**	**11.0**
M6:	12.3	15.1	12.4

Table 4: Comparison of our best models (M5, M6) with the the top performers from the MSR 2018 leaderboard as well as other recent state of the art methods.

model (LM) and an accompanying lexicon play a crucial role in decoding the emissions of the acoustic model. Keeping the lexicon fixed (as extracted from the transcripts for the training and validation set), we consider two choices for the language model: (i) a LM trained using only the transcripts for the training and validation data and (ii) a LM trained on much larger data from IndicCorp in addition to the transcripts for the training and validation data. Comparing rows 5 and 6 of Table 2 we observe that integrating a LM trained on larger generic data outperforms a LM trained on smaller task specific data. We also observe the significant improvement in the WER by integrating a LM (using small data or large data) as compared to the WER of the acoustic model (row 4). These two observations combined together lead to an interesting insight for developing ASR systems for low resource languages: *build better LMs using (relatively) easily available raw text corpora in these languages*.

Effect of Lexicon Having established the importance of using LMs trained on larger data, we now fix the LM and change the lexicon. In particular, instead of using a lexi-

Fine-tuning data	**ta**	**gu**	**te**
1 Hour	38.8	35.9	42.6
+LM$_{large}$+aug. lex.	20.6	19.9	20.3
10 Hours	26.5	25.4	28.1
+LM$_{large}$+aug. lex.	16.9	15.1	15.2
20 Hours	24.0	23.0	25.4
+LM$_{large}$+aug. lex.	16.2	13.8	13.8
40 Hour	22.1	20.9	22.9
+LM$_{large}$+aug. lex.	15.5	13.1	12.9

Table 5: Comparison of model performance across varying amounts of fine-tuning data. We used a beam size of 128 keeping all other hyperparameters same as M6.

	bn	ne	si
Baseline (Shetty and Umesh 2021)	17.9	12.9	21.8
Ekstep model*	15.2	13.8	20.0
M5:	14.0	13.6	-
M6:	**10.6**	**9.7**	-

Table 6: Comparison of our best models (M4, M6) with state-of-the-art results reported in the literature. * The Ekstep model was fine-tuned by us.

con which is only extracted from the transcripts of the train and validation data, we augment the lexicon with top 180K most frequent words from IndicCorp. The hope is that this augmented lexicon may reduce the number of OOVs during testing. Comparing rows 6 and 7 of Table 2, we observe that results are a bit mixed with a drop in WER for the MUCS and OpenSLR dataset but an increase in WER for the MSR dataser. To investigate this further, we did an oracle experiment where we further augmented the lexicon with words from the test set. We refer to this as an oracle experiment, because in practice we would not have access to the lexicon from the test set. We observed that this further reduces the WER with an average reduction of 1.36 WER across all languages and all 3 datasets. We do not report these numbers in the Table 2 as it would be unfair to compare this with the other results. This leads to another insight: *the right lexicon for the target domain can significantly help in low resource scenarios*. Of course, the lexicon should not be taken from the test set but independently curated.

Monolingual vs Multilingual Fine-tuning Some works (Conneau et al. 2020b) have shown that jointly fine-tuning a multilingual model using the task specific data in all languages leads to better performance. Motivated by these results, we jointly fine-tune the pretrained model by combining the training data from all the languages. We used temperature based sampling to ensure that all languages get a fair representation during fine-tuning. We did this experiment only for the MUCS and MSR datasets. During decoding we use individual language specific LMs and lexicons. Comparing the last two rows of Table 2, which only differ in the fine-tuning strategy (monolingual v/s multilingual), we observe that there is a consistent drop in the performance of the multilingual model across all the languages. In multilingual NLP, we typically see good gains due to a shared encoder (e.g. many-to-one translation) and less gains due to shared decoder (e.g. one-to-many translation) (Arivazhagan et al. 2019). Note that in our ASR models, encoder and decoder components are conflated. This difference in the behaviour of speech and NLP models needs further investigation.

Impact of Fine-tuning Dataset Size A significant result reported in Baevski et al. (2020) was that pretraining a model using a large amount of unlabeled data significantly reduces the amount of fine-tuning data needed for building ASR models. In particular, they showed that starting with a pretrained model, using just 10 mins of finet-tuning data, they were able to obtain single digit WERs for English. This is a significant result in the context of low resource languages. To check if the same holds true for Indian languages, we vary the amount of training data provided to the model from 1 to 10 to 40 hours. Quite disappointingly, our results as reported in Table 5, show that using just 1 hour of fine-tuning data leads to very poor performance. Further, we see gains in increasing the data from 20 hours to 40 hours, suggesting that adding more fine-tuning data would be beneficial. Lastly, note that we do not get a single digit WER for any of the languages compared to the results on English ASR in (Baevski et al. 2020). This indicates that developing ASR systems for Indic languages with their richer phoneme/character sets and vocabulary is more challenging than English.

Comparison with Existing Baselines

We now compare with existing models on the three datasets. For each dataset, we compare with the following models:

• **Baseline.** This is the baseline model as reported by authors of the dataset (OpenSLR) or the organisers of the challenge (MUCS/MSR). Often, this is a Time delay neural network (TDNN) model (Peddinti, Povey, and Khudanpur 2015).

• **Top3 Entries from the Leaderboard.** For the MUCS[12] and MSR[13] challenge, we include results for the Top 3 entries from the leaderboard of the challenge. Note that the MUCS challenge has only recently concluded and the system papers written by the participating teams are not yet available.

• **Best per Language.** For the MUCS challenge, we observed that some systems did poorly when considering the average performance across all languages, but did very well on a specific language (and hence did not appear in the top3 entries on the leaderboard). For example, one participant Uniphore got an average WER of 22.92 ranking 4th on the leadeboard but had the best performance for Tamil WER of 18.8. For a comprehensive comparison, for every language we report the *best individual system* for that language even if its performance was very poor on other languages.

• **Other SOTA Systems.** Given that the MSR challenge was launched in 2018, there are some follow-up works (Sailor and Hain 2020; Sen et al. 2021, 2020) which have reported results on this dataset. We report the numbers as it is from these works. Similarly, for OpenSLR, we compare with the results reported in Shetty and Umesh (2021).

• **EkStep.** As mentioned earlier, this is the recently released pretrained model covering 23 Indian languages (Gupta et al. 2021).

The above results are summarised in Table 3, 4, 6. Across all the datasets and all the languages (except MUCS-Tamil), we establish new state of the art results. Even for MUCS-Tamil, we outperform the top3 models on the leaderboard. These results clearly establish the recipe for developing ASR systems for Indian languages: *multilingual pretraining followed by monolingual fine-tuning combined with language models trained on large corpora with good lexicons.*

Conclusion

We report results of applying two recent and successful ideas from English ASR to Indic ASR: use of wav2vec like model architecture and use of unlabelled data to pretrain the model. We implement this with a curated dataset on Indic languages and a range of ablation studies on architecture, pretraining, fine-tuning, and decoding choices. Through this, we obtain state-of-the-art results on 9 Indic languages across 3 datasets. While advancing ASR systems for the next billion users from the sub-continent, our results highlight the need for larger resources and benchmarks across more languages.

[12] https://navana-tech.github.io/MUCS2021/leaderboard.html

[13] https://www.microsoft.com/en-us/research/event/interspeech-2018-special-session-low-resource-speech-recognition-challenge-indian-languages/#!leaderboard

Acknowledgments

We would like to thank the Ministry of Electronics and Information Technology (MeitY[14]) of the Government of India and the Centre for Development of Advanced Computing (C-DAC[15]), Pune for generously supporting this work and providing us access to multiple GPU nodes on the Param Siddhi Supercomputer. We would like to thank the EkStep Foundation for their generous grant which went into hiring human resources as well as cloud resources needed for this work. We would also like to thank the Robert Bosch Center for Data Science and Artificial Intelligence for supporting Sumanth and Gowtham through their Post Baccalaureate Fellowship Program.

References

Arivazhagan, N.; Bapna, A.; Firat, O.; Lepikhin, D.; Johnson, M.; Krikun, M.; Chen, M. X.; Cao, Y.; Foster, G. F.; Cherry, C.; Macherey, W.; Chen, Z.; and Wu, Y. 2019. Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges. *CoRR*, abs/1907.05019.

Baevski, A.; Auli, M.; and Mohamed, A. 2020. Effectiveness of self-supervised pre-training for speech recognition. arXiv:1911.03912.

Baevski, A.; Zhou, H.; Mohamed, A.; and Auli, M. 2020. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. arXiv:2006.11477.

Billa, J. 2018. ISI ASR System for the Low Resource Speech Recognition Challenge for Indian Languages. In *INTERSPEECH*.

Conneau, A.; Baevski, A.; Collobert, R.; Mohamed, A.; and Auli, M. 2020a. Unsupervised Cross-lingual Representation Learning for Speech Recognition. arXiv:2006.13979.

Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; and Stoyanov, V. 2020b. Unsupervised Cross-lingual Representation Learning at Scale. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J. R., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 8440–8451. Association for Computational Linguistics.

Conneau, A.; and Lample, G. 2019. Cross-lingual Language Model Pretraining. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 7057–7067.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

[14]https://www.meity.gov.in/

[15]https://www.cdac.in/index.aspx?id=pune

Diwan, A.; Vaideeswaran, R.; Shah, S.; Singh, A.; Raghavan, S.; Khare, S.; Unni, V.; Vyas, S.; Rajpuria, A.; Yarra, C.; Mittal, A.; Ghosh, P. K.; Jyothi, P.; Bali, K.; Seshadri, V.; Sitaram, S.; Bharadwaj, S.; Nanavati, J.; Nanavati, R.; Sankaranarayanan, K.; Seeram, T.; and Abraham, B. 2021. Multilingual and code-switching ASR challenges for low resource Indian languages. *Proceedings of Interspeech*.

Emeneau, M. B. 1956. India as a lingustic area. *Language*.

Fathima, N.; Patel, T.; C, M.; and Iyengar, A. 2018. TDNN-based Multilingual Speech Recognition System for Low Resource Indian Languages. In Yegnanarayana, B., ed., *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, 3197–3201. ISCA.

Graves, A.; Fernández, S.; Gomez, F.; and Schmidhuber, J. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, 369–376. New York, NY, USA: Association for Computing Machinery. ISBN 1595933832.

Gupta, A.; Chadha, H. S.; Shah, P.; Chimmwal, N.; Dhuriya, A.; Gaur, R.; and Raghavan, V. 2021. CLSRIL-23: Cross Lingual Speech Representations for Indic Languages. *CoRR*, abs/2107.07402.

Heafield, K. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 187–197. Edinburgh, Scotland: Association for Computational Linguistics.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1): 117–128.

Kakwani, D.; Kunchukuttan, A.; Golla, S.; N.C., G.; Bhattacharyya, A.; Khapra, M. M.; and Kumar, P. 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4948–4961. Online: Association for Computational Linguistics.

Kim, C.; and Stern, R. M. 2008. Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis. In *Proc. Interspeech 2008*, 2598–2601.

Kjartansson, O.; Sarin, S.; Pipatsrisawat, K.; Jansche, M.; and Ha, L. 2018. Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali. In *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*, 52–55. Gurugram, India.

Kunchukuttan, A. 2020. The IndicNLP Library. https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf.

Kunchukuttan, A.; and Bhattacharyya, P. 2020. Utilizing Language Relatedness to improve Machine Translation: A Case Study on Languages of the Indian Subcontinent. *arXiv preprint arXiv:2003.08925*.

Liptchinsky, V.; Synnaeve, G.; and Collobert, R. 2019. Letter-Based Speech Recognition with Gated ConvNets. arXiv:1712.09444.

Mohamed, A.; Okhonko, D.; and Zettlemoyer, L. 2020. Transformers with convolutional context for ASR. arXiv:1904.11660.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Park, D. S.; Chan, W.; Zhang, Y.; Chiu, C.-C.; Zoph, B.; Cubuk, E. D.; and Le, Q. V. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Peddinti, V.; Povey, D.; and Khudanpur, S. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, 3214–3218. ISCA.

Pratap, V.; Hannun, A.; Xu, Q.; Cai, J.; Kahn, J.; Synnaeve, G.; Liptchinsky, V.; and Collobert, R. 2019. Wav2Letter++: A Fast Open-source Speech Recognition System. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Pulugundla, B.; Baskar, K. M.; Kesiraju, S.; Egorova, E.; Karafiát, M.; Burget, L.; and Černocký, J. 2018. BUT system for low resource Indian language ASR. In *Proceedings of Interspeech 2018*, volume 2018, 3182–3186. International Speech Communication Association.

Sailor, H. B.; and Hain, T. 2020. Multilingual Speech Recognition Using Language-Specific Phoneme Recognition as Auxiliary Task for Indian Languages. In *INTERSPEECH*.

Sen, B.; Agarwal, A.; Ganesh, M. S.; and Vuppala, A. K. 2020. Short Term Context-based Fast Multilingual Acoustic Model for Low Resource Languages. In *Machine Learning, AI & Data Science Conference (MLADS)*. Microsoft.

Sen, B.; Agarwal, A.; Ganesh, M. S.; and Vuppala, A. K. 2021. Reed: An Approach Towards Quickly Bootstrapping Multilingual Acoustic Models. In *IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, China, January 19-22, 2021*, 272–279. IEEE.

Shetty, V. M.; and Umesh, S. 2021. Exploring the use of Common Label Set to Improve Speech Recognition of Low Resource Indian Languages. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7228–7232.

Srivastava, B. M. L.; Sitaram, S.; Kumar Mehta, R.; Doss Mohan, K.; Matani, P.; Satpal, S.; Bali, K.; Srikanth, R.; and Nayak, N. 2018. Interspeech 2018 Low Resource Automatic Speech Recognition Challenge for Indian Languages. In *Proc. 6th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)*, 11–14.

Subbārāo, K. V. 2012. *South Asian languages: A syntactic typology*. Cambridge University Press.

Tang, Y.; Tran, C.; Li, X.; Chen, P.; Goyal, N.; Chaudhary, V.; Gu, J.; and Fan, A. 2020. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning. *CoRR*, abs/2008.00401.

Wang, C.; Rivière, M.; Lee, A.; Wu, A.; Talnikar, C.; Haziza, D.; Williamson, M.; Pino, J.; and Dupoux, E. 2021a. VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation. arXiv:2101.00390.

Wang, C.; Wu, Y.; Qian, Y.; Kumatani, K.; Liu, S.; Wei, F.; Zeng, M.; and Huang, X. 2021b. UniSpeech: Unified Speech Representation Learning with Labeled and Unlabeled Data. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 10937–10947. PMLR.

Wu, F.; Fan, A.; Baevski, A.; Dauphin, Y. N.; and Auli, M. 2019. Pay Less Attention with Lightweight and Dynamic Convolutions. arXiv:1901.10430.