**IndicXlit**

The IndicXlit model is a transformer-based multilingual transliteration architecture designed for fast and accurate mapping between Roman script and native Indic scripts across 21 languages.

## Architecture Overview

* Type: Transformer encoder-decoder architecture
* Granularity: Character-level (not word-level)
* Parameters: ~11 million
* Layers: 6 encoder and 6 decoder layers
* Input Embedding Size: 256
* Attention Heads: 4 in both encoder and decoder
* Feedforward Dimension: 1024
* Vocabulary: Joint, multilingual vocabulary supporting romanized input and script output
* Special Tokens: Uses a “target language tag” in the input sequence to inform the model which script/language to generate (akin to multilingual NMT)
* Training Data: Aksharantar dataset (parallel word pairs—26 million pairs, 21 Indic languages, 12 scripts)

## Model Flow

* Encoder: Reads input as a sequence of characters (Roman or native script, depending on direction).
* Target Language Tag: Prepend a token specifying the target language/script.
* Decoder: Generates the output character by character in the target script.
* Beam Search: Decoding uses beam search (beam size = 4) and a reranking mechanism combining transliteration scores and a language model probability.
* Training Objective: Standard cross-entropy loss for next character prediction.

## Why This Design?

* One-to-Many Setting: Single model for all script/language pairs, enabled by the target language token.
* Transfer Learning: Low-resource languages benefit from shared representation and patterns with high-resource ones.
* Canonicalization: Helps standardize spellings by learning from diverse language data.
* Efficiency: Easier deployment and maintenance as one model supports 21 languages natively.

## 

## Key Hyperparameters

| Parameter | Value |
| --- | --- |
| Encoder/Decoder layers | 6 each |
| Attention heads | 4 |
| Input embedding size | 256 |
| Feedforward dimension | 1024 |
| Model size | ~11M parameters |
| Optimizer | Adam (betas: 0.9,0.98) |
| Learning rate (peak) | 0.001 |
| Scheduler | inverse-sqrt |
| Warmup steps | 4000 |
| Temperature sampling (T) | 1.5 |

## Reference Papers and Resources

* [Aksharantar: Open Indic-language Transliteration datasets and models](https://arxiv.org/pdf/2205.03018.pdf)
* [IndicXlit GitHub + documentation](https://github.com/AI4Bharat/IndicXlit)

## Summary Table: IndicXlit Components

| Component | Description |
| --- | --- |
| Encoder | 6 layers, 4 attention heads, character-level |
| Decoder | 6 layers, 4 attention heads, character-level |
| Input Embedding | 256 dimensions |
| Feedforward Network | Size 1024 |
| Target Language Tag | Indicates output script/language |
| Training Corpus | Aksharantar: 26M pairs, 21 languages, 12 scripts |
| Multilingual | Single model covers 21 Indic languages |

IndicXlit leverages transformer architectures for robust, efficient, and highly accurate transliteration across Indian languages, with strong transfer learning benefits—particularly important for smaller, low-resource languages.

Based on my analysis of the IndicXlit research paper and related documentation, IndicXlit does not use a specific standardized Roman script scheme like ITRANS or ISO 15919. Instead, it uses a flexible, data-driven approach to Roman script representation.

## IndicXlit's Roman Script Approach

## Key Characteristics:

1. No Fixed Standard: IndicXlit is trained on the Aksharantar dataset which contains diverse romanization patterns found "in the wild"
2. Real-World Variations: The training data includes multiple romanization variants for the same word to capture natural user behavior
3. Flexible Input: Can handle various romanization styles that users naturally employ in digital communication

## Examples from the Dataset:

The paper shows how the same word can be romanized in multiple ways:

* मैं (Hindi "I") can be: main, mai, mein, mei
* में (Hindi "in") overlaps with some of these variations

## Data Collection Strategy:

Manual Collection Process:

* Human annotators provided up to 4 romanization variants per native word
* Validators could add 2 additional variants
* This captured natural diversity in how people romanize words

Mining Sources:

* Wikidata: Entity names in multiple scripts
* Parallel corpora: Samanantar dataset mining
* Monolingual corpora: IndicCorp dataset
* Existing datasets: Compilation of available resources

## Design Philosophy:

Instead of enforcing a rigid scheme like:

* ITRANS: aa, ii, uu for long vowels
* ISO 15919: ā, ī, ū with diacritics

IndicXlit learns from actual user behavior, accepting inputs like:

* namaste, namaskar, namasthe
* bharat, bharath, bhaarat

# IndicXlit can handle various romanization styles:

inputs = [

"namaste", # Simple

"namaskar", # Alternative

"namasthe", # With 'h'

"namaskaara" # Elongated vowel

]

# All would be processed appropriately

IndicXlit uses a flexible, corpus-driven romanization approach rather than adhering to any single standard like ITRANS or ISO 15919. This makes it more practical for real-world applications where users type in various natural ways, rather than following strict transliteration conventions.