Self-Supervised Code-Mixed Representation Learning for Multi-Label Emoji, Sentiment, and Emotion Prediction

Asif Shehzad, Rahul Sharma and Pushpak Bhattacharyya

Department of Computer Science and Engineering, Indian Institute of Technology Bombay

Abstract

Emoji usage and code-mixed language have become central to informal online communication, especially in multilingual communities. Predicting suitable emojis and inferring sentiment and emotion from noisy, code-mixed social media text is challenging due to non-standard spelling, cross-script mixing, and strong pragmatic effects. Prior work has proposed specialized encoders and multi-task frameworks for joint prediction of multi-label emojis, sentiment, and emotion on English–Hindi code-mixed tweets. However, these approaches are built on relatively shallow architectures and do not fully exploit recent advances in self-supervised representation learning.

In this paper, we introduce *CodeMixLM*, a transformer encoder specialized for Hinglish code-mixed text through continued self-supervised pretraining on unlabeled code-mixed tweets. In addition to standard masked language modeling, we use three auxiliary objectives tailored to code-mixed social media: masked span denoising for local-noise robustness, script-aware identification to explicitly encode cross-script cues, and emoji-aware contrastive learning to inject weak affective supervision from naturally occurring emojis. We then fine-tune CodeMixLM in a multi-task setting for (i) multi-label emoji prediction, (ii) three-way sentiment classification, and (iii) seven-way emotion classification on the SENTIMOJI benchmark. On the test set, CodeMixLM achieves the strongest performance among the compared systems, improving emoji macro-F_1 from 0.654 to 0.683 relative to a multi-task XLM-R baseline, while also improving sentiment and emotion macro-F_1. We further analyze label efficiency under reduced supervision and robustness under controlled perturbations that simulate spelling variation and code-switch boundary shifts, showing that the proposed pretraining objectives yield more data-efficient and more noise-resilient representations for code-mixed affective prediction.

Keywords: code-mixed NLP, emoji prediction, sentiment and emotion classification, self-supervised transformer learning

1. Introduction

Social media users in multilingual communities routinely mix languages and scripts within the same message, a phenomenon known as *code-mixing* or *code-switching* [1–3]. In South Asian contexts, English–Hindi code-mixed text written predominantly in Roman script (often referred to as Hinglish) is particularly widespread on platforms such as X/Twitter, Facebook, Instagram, and YouTube comments [4, 5]. In such environments, users fluidly alternate between English and Hindi, exploit non-standard spellings and transliterations (for example, “yaar”, “yar”, “yrr” for the same underlying

Corresponding author: Pushpak Bhattacharyya (pb@cse.iitb.ac.in)

BCDS 2025, Vol. 6(1), 39-60. DOI: 10.71448/bcds2561-3

token), and frequently drop diacritics or morpho-syntactic markers. This results in highly noisy, non-canonical text that deviates substantially from the monolingual corpora used to pretrain most language models. At the same time, emojis have emerged as a pervasive semiotic resource: they function not only as decorative symbols but also as compact markers of affect, stance, sarcasm, politeness, and discourse intent [6–8]. Accurately modeling the joint behaviour of emojis, sentiment, and emotion in code-mixed text is therefore crucial for downstream applications in content moderation, toxic comment detection, opinion mining, conversational agents, and social media analytics that operate in multilingual regions [9, 10].

Recent work has introduced SENTIMOJI, a benchmark dataset of English–Hindi code-mixed tweets annotated with *three* highly correlated label spaces: multi-label emojis, three-way sentiment (positive/neutral/negative), and multi-class emotion (joy, anger, sadness, fear, disgust, surprise, neutral) [11]. This dataset captures realistic usage patterns of Hinglish and emoji combinations in naturally occurring social media posts. Building on this resource, prior authors have proposed multi-task architectures that jointly predict emojis, sentiment, and emotion from a shared encoder, demonstrating that multi-task learning can exploit cross-task regularities and improve performance over training separate models for each task [11–13]. In particular, sharing representations across emoji prediction and affective classification encourages the model to capture latent affective dimensions that are difficult to learn reliably from any single supervision channel alone.

Despite these advances, the underlying encoders in this line of work remain relatively shallow or task-specific and do not fully exploit recent progress in self-supervised pretraining and large language models (LLMs). Code-mixed text exhibits strong distributional shifts relative to standard monolingual corpora: orthographic variation, phonetic transliteration of Hindi into Roman script, and arbitrary switching between English and Hindi phrases all violate the assumptions of standard tokenization and language modeling [3, 14]. As a result, generic multilingual encoders such as mBERT and XLM-R [17, 18]—although powerful on many cross-lingual tasks—are not explicitly adapted to the statistical properties of code-mixed social media, which differs substantially from their pretraining data in genre, register, and noise characteristics. At the same time, emojis themselves provide an abundant, weakly supervised signal that can be exploited during self-supervised pretraining rather than only at fine-tuning time: prior work in monolingual settings has shown that training models to predict emojis or hashtags at scale yields rich, sentiment-aware representations that transfer well to downstream emotion and sentiment tasks [8, 15, 16]. Existing code-mixed methods typically do not integrate such emoji-aware objectives into pretraining and instead treat emojis as labels solely within supervised learning on relatively small annotated datasets.

In this paper, we address these gaps by introducing *CodeMixLM*, a self-supervised code-mixed language model tailored to English–Hindi social media. Our key idea is to leverage large amounts of unlabeled Hinglish tweets—with naturally occurring emojis, hashtags, and code-switch patterns—to continue pretraining a multilingual transformer using objectives that are directly aligned with downstream needs: robust modeling of noisy code-mixed text and emoji-aware semantics. Starting from a strong multilingual backbone such as XLM-R, we perform domain-adaptive pretraining [19] on code-mixed corpora with a combination of masked span denoising, script-aware language identification, and emoji-aware contrastive learning. These objectives encourage the model to become more robust to local orthographic noise, to encode information about script and language at the token level, and to align representations of tweets that share similar emoji semantics in an unsupervised fashion, thereby embedding affective information into the model before any task-specific supervision.

On top of this encoder, we develop a multi-task fine-tuning framework that jointly learns to predict multi-label emojis, three-way sentiment, and seven-way emotion labels from a shared CodeMixLM representation using lightweight task-specific heads. Rather than describing our contributions as isolated algorithmic innovations, we view them as components of a coherent design pattern: first, we explicitly adapt a multilingual transformer to the code-mixed social media domain via self-supervised learning; second, we exploit emojis as a distant supervision signal within a contrastive pretraining objective; and third, we couple this adapted encoder with supervised multi-task learning on SENTIMOJI to jointly model emojis, sentiment, and emotion. Through extensive experiments on the SENTIMOJI benchmark, we show that CodeMixLM significantly outperforms (a) prior bespoke architectures designed specifically for this dataset and (b) directly fine-tuning off-the-shelf multilingual transformers such as mBERT and XLM-R, particularly in terms of macro-F_1 for multi-label emoji prediction and performance on minority emotion classes. Moreover, label-efficiency and robustness analyses demonstrate that code-mixed self-supervision yields stronger performance in low-data regimes and under realistic perturbations such as spelling variants and shifts in code-switch boundaries.

Overall, our results provide strong evidence that domain-specialized self-supervised learning is a powerful tool for advancing code-mixed NLP, going beyond task-specific architectural tweaks or naive fine-tuning of generic multilingual models. By explicitly modeling the interplay between code-mixing, emojis, and affect within the pretraining and fine-tuning pipeline, CodeMixLM offers a stronger and more principled baseline for future work on emoji-aware affective computing in multilingual social media settings.

2. Related Work

2.1. Code-Mixed NLP

Code-mixed language has attracted growing interest in NLP, particularly for South Asian language pairs such as Hindi–English, Bengali–English, and Tamil–English. Early work focused on documenting linguistic phenomena and constructing small annotated corpora, while more recent efforts have scaled up to shared tasks and larger benchmarks [3]. A series of shared tasks has targeted language identification, POS tagging, parsing, sentiment analysis, and hate speech detection in code-mixed text, highlighting both the prevalence of code-mixing in real-world social media and the difficulty of building robust systems that can cope with non-standard spellings and frequent language switches [20–22]. For Hindi–English specifically, researchers have developed shallow parsing pipelines, POS taggers, and token-level language identification systems tailored to noisy Romanized text [4, 5]. These systems typically combine character-level and word-level features with sequence models such as CRFs, BiLSTMs, or CNNs and demonstrate that explicit modeling of orthographic variation and language ID is crucial for downstream tasks.

Beyond basic annotation layers, there has been substantial work on code-mixed sentiment analysis and hate/offensive speech detection. Several shared tasks—for example, SentiMix and related evaluations—have provided labeled datasets of Hindi–English and other code-mixed tweets annotated for coarse-grained sentiment and offensiveness [10, 23, 24]. Models in these settings range from traditional feature-based classifiers with n-grams and lexicon features to deep architectures that use BiLSTMs, CNNs, or multilingual transformers such as mBERT and XLM-R fine-tuned directly on task data [3, 9]. While transformer-based approaches generally outperform earlier architectures, they are usually trained in a purely supervised manner on relatively small labeled datasets and do not

exploit the abundance of unlabeled code-mixed text. Moreover, even when multilingual encoders are used, they are rarely adapted to code-mixed social media through domain-specific pretraining, which limits their ability to capture the characteristic noise patterns and switching behaviour of Hinglish and other mixed varieties.

Overall, the code-mixed NLP literature has established a solid foundation of datasets, tasks, and baseline models across several South Asian language pairs, but the dominant paradigm remains that of fine-tuning generic multilingual encoders or training task-specific sequence models directly on annotated data. The potential of large-scale self-supervised pretraining on unlabeled code-mixed corpora, especially with objectives tailored to code-switch boundaries and transliteration phenomena, remains relatively underexplored.

2.2. Emoji Prediction and Affective Computing

Emoji prediction has been studied as both a multi-class and multi-label classification problem, primarily in monolingual English settings. Early approaches treated emojis as discrete labels and used bag-of-words or character n-gram features with linear classifiers to predict a single most likely emoji (for example, a smiling face 😊 or a crying face 😢) given the surrounding text [6]. Subsequent work leveraged neural architectures: convolutional and recurrent networks were trained end-to-end to map tweets to emoji distributions, often demonstrating substantial gains over traditional feature-based methods [15]. These models highlighted the strong association between textual cues, user style, and emoji usage, and showed that even a limited emoji inventory—such as a small set of faces 😀, hearts ❤️, and symbolic emojis like fire 🔥—can capture a rich space of affective and pragmatic functions.

A parallel line of research has used emojis as distant supervision to learn sentiment- and emotion-aware representations. Large-scale pretraining on millions of emoji-bearing tweets, where the emoji is treated as a noisy label (for example, 😂 for amusement or 😡 for anger), has been shown to yield embeddings that transfer well to tasks such as sentiment classification, emotion recognition, and sarcasm detection [7, 8, 16]. Other work has focused on learning vector representations of emojis themselves, for example through distributional methods like emoji2vec, which align emoji embeddings with word embeddings in a shared semantic space [25]. These approaches generally operate on monolingual English data and assume relatively clean, standardized text.

In code-mixed contexts, existing work on emojis is more limited and usually focuses on sentiment classification or hate speech detection, where emojis are treated as additional tokens or features rather than as primary prediction targets [3, 10]. Some studies note that emojis can mitigate the ambiguity of code-mixed utterances by signaling affect or stance—for instance, a neutral-looking message accompanied by a clearly negative emoji such as an angry face 😠—but they do not explicitly model emojis in a multi-label setting or exploit them as a source of distant supervision during pretraining. As a result, there is still a gap between the rich monolingual literature on emoji prediction and representation learning and the practical needs of code-mixed social media analysis, where emojis and code-mixing interact in complex, culturally specific ways.

2.3. Self-Supervised Learning and LLMs for Social Media

Self-supervised pretraining on large unlabeled corpora has become the dominant paradigm in NLP. Models such as BERT and XLM-R use masked language modeling and related denoising objectives to learn contextual word representations that can be fine-tuned for a wide range of downstream tasks [17, 18]. Subsequent work has refined these ideas through architectural and objective improvements,

for example with RoBERTa, which optimizes the pretraining recipe for monolingual English, and with large autoregressive models such as GPT-2 and GPT-3, which demonstrate impressive few-shot capabilities across tasks [27–29]. In parallel, contrastive self-supervised learning methods have been proposed for text, where sentence or document representations are trained to bring semantically similar pairs closer while pushing dissimilar pairs apart [30]. These methods yield powerful general-purpose encoders but are typically trained on relatively clean, monolingual corpora such as Wikipedia and newswire.

For social media, several works have adapted pretraining to noisy user-generated content. BERTweet, for example, is a BERT-like model trained on a large corpus of English tweets and has been shown to outperform generic BERT on a variety of Twitter benchmarks [31]. Other work has explored domain-adaptive pretraining, where a generic language model is further pretrained on in-domain text (such as biomedical articles, legal documents, or tweets) before fine-tuning on task data, yielding consistent gains especially in low-resource settings [19]. Some studies have also incorporated auxiliary objectives based on hashtags or emojis as weak labels in order to inject affective or topical information into the learned representations [8, 15]; for example, a model might be trained to predict whether a tweet should contain a heart ❤️ or an angry face 😡 based solely on its text.

However, most existing self-supervised and LLM-based models for social media are monolingual and do not specifically target code-mixed text or the joint modeling of emojis, sentiment, and emotion. Multilingual transformers such as mBERT and XLM-R capture cross-lingual regularities across dozens of languages, but their pretraining corpora contain relatively little code-mixed social media and almost no systematic treatment of orthographic variation, transliteration, or code-switch boundaries [3, 18]. As a result, simply fine-tuning these models on small code-mixed datasets can leave substantial domain mismatch unaddressed. Our work lies at the intersection of these strands: we combine multilingual transformers with code-mixed self-supervised pretraining and emoji-aware objectives, and evaluate in a multi-task affective computing setup tailored to Hinglish. By explicitly adapting a multilingual encoder to code-mixed tweets using objectives that exploit both the structure of code-mixing and the presence of emojis as weak labels, we aim to bridge the gap between generic self-supervised models and the specific challenges of emoji-aware affective analysis in multilingual social media.

3. Tasks and Dataset

3.1. Problem Formulation

Let $x = (w_1, \ldots, w_n)$ denote a tokenized English–Hindi code-mixed tweet, where tokens may be in Roman or Devanagari script and may belong to either language or to non-lexical categories such as hashtags, usernames, and URLs. For each tweet x we consider three prediction problems that share the same input but differ in their output spaces and loss functions.

The first task is *multi-label emoji prediction*. Each tweet is annotated with a subset $Y^{\text{emo}} \subseteq \mathcal{E}$ of emojis from a finite vocabulary $\mathcal{E}$, for example the $|\mathcal{E}|= 12$ most frequently used emojis in the corpus (such as smiling faces, crying faces, hearts, and fire icons, e.g., 😊, 😢, ❤️, 🔥). The learning problem is to predict a binary vector $\mathbf{y}^{\text{emo}} \in \{0,1\}^{|\mathcal{E}|}$, where $\mathbf{y}^{\text{emo}}_j = 1$ if and only if the j-th emoji is present among the gold labels. Given a model f^{emo} with parameters θ_{emo}, we obtain predicted probabilities $\hat{\mathbf{y}}^{\text{emo}} = f^{\text{emo}}(\mathbf{h}(x)) \in [0,1]^{|\mathcal{E}|}$, and we train this head using a standard multi-label loss such as sigmoid cross-entropy aggregated over emojis.

The second task is sentence-level *sentiment classification*. Each tweet is labeled with a sentiment $y^{\text{sent}} \in \mathcal{S} = \{\text{positive}, \text{neutral}, \text{negative}\}$. We model this as a three-way single-label classification problem. The sentiment head f^{sent} maps the shared representation to class scores, and the corresponding one-hot vector $\mathbf{y}^{\text{sent}} \in \{0,1\}^{|\mathcal{S}|}$ is used to define a softmax cross-entropy loss. In practice, this task encourages the model to capture coarse valence information that is correlated, but not identical, to the presence of specific emojis (for example, a predominance of positive emojis such as hearts ❤️ or laughing faces 😂) or fine-grained emotions.

The third task is *emotion classification*. Each tweet is annotated with a categorical label $y^{\text{emo2}} \in \mathcal{C}$, where $\mathcal{C}$ contains six basic emotions (joy, anger, sadness, fear, disgust, surprise) together with a neutral class. This yields a seven-way single-label classification task. As with sentiment, the emotion head f^{emo2} produces a probability distribution $\hat{\mathbf{y}}^{\text{emo2}} \in [0,1]^{|\mathcal{C}|}$, trained with cross-entropy against the one-hot target vector $\mathbf{y}^{\text{emo2}}$. Unlike sentiment, the emotion labels aim to distinguish between qualitatively different negative states (e.g., anger versus sadness) and to separate genuine positivity from surprise or neutrality.

All three tasks share the same encoder. Given x, the encoder produces a tweet-level representation $\mathbf{h}(x) \in \mathbb{R}^d$, for example by applying a transformer over tokens and pooling the final hidden states. The three task-specific heads are simple feed-forward layers (possibly with non-linearities and dropout) that take $\mathbf{h}(x)$ as input and output $\hat{\mathbf{y}}^{\text{emo}}$, $\hat{\mathbf{y}}^{\text{sent}}$, and $\hat{\mathbf{y}}^{\text{emo2}}$ respectively. During multi-task training, we optimize a weighted sum of the three task losses, which allows gradient signals from emoji prediction, sentiment classification, and emotion classification to jointly shape the shared representation.

Table 1 summarizes the three tasks, their label spaces, and the corresponding prediction types.

Table 1. Overview of the three prediction tasks defined on code-mixed tweets.

Task	Label set	Type	Output representation
Emoji prediction	$\mathcal{E}$ (e.g., 12 emojis such as 😊, ❤️)	Multi-label	$\mathbf{y}^{\text{emo}} \in \{0,1\}^{\lvert\mathcal{E}\rvert}$
Sentiment classification	$\mathcal{S} = \{\text{pos}, \text{neu}, \text{neg}\}$	Single-label (3-way)	$\mathbf{y}^{\text{sent}} \in \{0,1\}^3$
Emotion classification	$\mathcal{C}$ (6 emotions + neutral)	Single-label (7-way)	$\mathbf{y}^{\text{emo2}} \in \{0,1\}^{\lvert\mathcal{C}\rvert}$

3.2. SENTIMOJI Dataset

We evaluate our approach on the SENTIMOJI dataset of English–Hindi code-mixed tweets, which extends an existing Hinglish sentiment corpus with additional annotations for emojis and emotions. Each instance consists of the original tweet along with three complementary label views. The text is typically written in Roman script, exhibits substantial variation in spelling and punctuation, and mixes English and Hindi fragments within the same message. Tweets often contain multiple emojis, and the average number of emojis per tweet is greater than two (for example, combinations of laughing faces 😂 with crying faces 😢, or a heart ❤️ followed by fire 🔥), making a multi-label formulation for emoji prediction essential.

For each tweet, the emoji annotation specifies one or more emojis drawn from a fixed inventory of twelve frequently occurring emojis. These labels are converted into a multi-hot vector $\mathbf{y}^{\text{emo}}$ of length $|\mathcal{E}|= 12$. The sentiment annotation assigns a single label from the set $\{\text{positive}, \text{neutral}, \text{negative}\}$, which we encode as a one-hot vector $\mathbf{y}^{\text{sent}}$. The emotion annotation assigns a single label from the set $\{\text{joy}, \text{anger}, \text{sadness}, \text{fear}, \text{disgust}, \text{surprise}, \text{neutral}\}$, similarly encoded as $\mathbf{y}^{\text{emo2}}$. The joint availability of these three label spaces allows us to study how information about sentiment and emotion can

support emoji prediction and, conversely, how emojis can serve as auxiliary signals for affective classification.

The dataset contains 19,642 English–Hindi code-mixed tweets. We follow the original partitioning into training, validation, and test splits provided with the benchmark (train/validation/test: 15,714/1,964/1,964) and retain these splits without modification. We additionally verify that the empirical distributions of sentiment, emotion, and emoji usage are consistent across splits. In particular, the relative frequencies of positive, neutral, and negative sentiment and the proportion of tweets with multiple emojis are approximately preserved, reducing the risk that apparent gains are caused by split-specific label priors rather than modeling choices.

Table 2 summarizes the main annotated components of each SENTIMOJI instance and the corresponding representations used in our model.

Table 2. Summary of SENTIMOJI annotations and their representations in our framework.

Component	Description	Representation
Tweet text	Code-mixed English–Hindi tweet	Token sequence $x = (w_1, \ldots, w_n)$
Emoji labels	Subset of 12 frequently used emojis	Multi-hot vector $\mathbf{y}^{\text{emo}} \in \{0,1\}^{12}$
Sentiment label	{positive, neutral, negative}	One-hot vector $\mathbf{y}^{\text{sent}} \in \{0,1\}^{3}$
Emotion label	6 basic emotions + neutral	One-hot vector $\mathbf{y}^{\text{emo2}} \in \{0,1\}^{7}$

In addition to the labeled SENTIMOJI splits, we compile an unlabeled corpus of 5,000,000 publicly available English–Hindi code-mixed tweets for self-supervised pretraining of the encoder. Tweets are collected via public social media APIs over a two-year window and are filtered using simple language and keyword heuristics to enrich for Hinglish content while retaining naturally occurring emojis. We remove retweets and near-duplicates (by hashing normalized text), drop messages dominated by URLs or boilerplate, and normalize obvious artefacts such as repeated URLs and non-linguistic placeholders. After filtering, 4,732,418 tweets remain for continued pretraining. This unlabeled corpus is used exclusively for self-supervised learning and contains no manual sentiment, emotion, or emoji annotations; the SENTIMOJI labels are reserved for supervised fine-tuning and evaluation on the downstream tasks described above.

Beyond the three primary SENTIMOJI tasks, we also include an auxiliary hate speech detection task in our label-efficiency and robustness analyses. For this purpose, we use a standard publicly available Hinglish hate-speech dataset with binary labels and its standard train/validation/test split, treating it as an additional supervised task during fine-tuning when required by the corresponding experiments.

4. CodeMixLM: Self-Supervised Code-Mixed Pretraining

4.1. Backbone Architecture

CodeMixLM starts from a multilingual transformer encoder, such as XLM-R base, which consists of 12 transformer blocks with multi-head self-attention, a hidden size of 768, and a shared subword vocabulary learned from a large multilingual corpus. We retain the original embedding and encoder layers of this backbone and adapt them to the code-mixed domain through continued pretraining on unlabeled English–Hindi tweets.

Given an input tweet $x = (w_1, \ldots, w_n)$, we first apply the backbone tokenizer to segment the surface tokens into subword units $(t_1, \ldots, t_m)$. We prepend a special `[CLS]` token and append a `[SEP]` token if required by the backbone, yielding the sequence $(t_0, t_1, \ldots, t_m, t_{m+1})$ with $t_0 =$ `[CLS]`. This sequence is mapped to embeddings, combined with positional encodings, and passed through the transformer encoder to obtain contextual representations

$$\mathbf{H} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_m, \mathbf{h}_{m+1}) = \text{Transformer}(t_0, \ldots, t_{m+1}),$$

where $\mathbf{h}_i \in \mathbb{R}^d$ denotes the final hidden state corresponding to subword t_i. The vector $\mathbf{h}_0$ associated with `[CLS]` is used as a sequence-level representation that summarizes the tweet and will later serve as the input to the task-specific heads for emoji prediction, sentiment classification, and emotion classification.

In the base multilingual model, pretraining is carried out using generic masked language modeling (MLM) and, in some variants, sentence-level objectives on a mixture of news, Wikipedia, and web text. These corpora contain relatively little code-mixed social media data, and almost no systematic coverage of English–Hindi transliteration, non-standard spellings, or emoji usage (for example, frequent use of hearts ❤️ or laughing faces 😂). To bridge this gap, we continue pretraining the backbone on a large corpus of unlabeled Hinglish tweets using additional self-supervised objectives that are tailored to the characteristics of code-mixed social media.

4.2. Pretraining Objectives

In addition to the backbone's original masked language modeling (MLM) loss, CodeMixLM is trained with three auxiliary objectives designed to improve robustness to noise, encode script and language information, and exploit emojis as weak supervision. All objectives are optimized jointly during continued pretraining.

Masked Span Denoising. Standard MLM randomly masks individual tokens and trains the model to predict them from their context. While effective, this token-level masking does not fully capture the kinds of local perturbations common in code-mixed tweets, such as repeated letters, elongated forms, and short noisy phrases. To better handle such phenomena, we adopt a masked span denoising objective.

For each input sequence, we sample a set of contiguous spans $\mathcal{S} = \{(a_k, b_k)\}$, where each span covers positions $a_k, \ldots, b_k$ with length between 2 and 5 tokens. With probability $p_{\text{span}} = 0.15$, we replace the entire span with a single `[MASK]` token or apply a short corruption in the style of BERT masking (80% `[MASK]`, 10% random subword, 10% unchanged), while ensuring spans do not overlap and that the expected fraction of corrupted tokens remains comparable to standard MLM. The model is then trained to reconstruct the original subword sequence for each masked span. Concretely, for a span (a, b), we let $\tilde{x}$ denote the corrupted input and maximize the log-likelihood

$$\log p_\theta(t_a, \ldots, t_b \mid \tilde{x}),$$

where p_θ is parameterized by the transformer followed by a subword softmax layer. The span-level denoising loss $\mathcal{L}_{\text{span}}$ is obtained by summing the negative log-likelihood over all masked spans in the batch. This objective encourages the model to rely on broader context when reconstructing multi-token fragments, which in turn improves robustness to local irregularities and code-mixed patterns such as transliterated multiword expressions.

Script-Aware Language Identification. English–Hindi code-mixed tweets often contain a mixture of Roman and Devanagari script, and transliterated Hindi words written in Roman script can be easily confused with English words. To help the model disambiguate and better track code-switch boundaries, we introduce a script-aware language identification objective.

For each subword token t_i in the input, we infer a coarse script label $\ell_i \in \mathcal{L}$ from Unicode ranges, using three categories: Roman, Devanagari, and other. During pretraining, we sample a supervision set $\mathcal{I}$ by selecting 15% of token positions uniformly at random and treat the corresponding script labels as hidden; the model must then predict ℓ_i from the contextual embedding $\mathbf{h}_i$. We then attach a small classification head f_{script} on top of the contextual embeddings and train it to predict the script of each selected token:

$$\hat{\mathbf{p}}_i = f_{\text{script}}(\mathbf{h}_i) \in [0,1]^{|\mathcal{L}|}, \quad \mathcal{L}_{\text{script}} = -\sum_{i \in \mathcal{I}} \log \hat{\mathbf{p}}_i(\ell_i),$$

where $\mathcal{L}$ is the set of script categories and $\mathcal{I}$ is the set of positions chosen for supervision. The loss $\mathcal{L}_{\text{script}}$ encourages the encoder to represent script- and language-specific information explicitly in $\mathbf{h}_i$, which is useful for downstream tasks that benefit from distinguishing Hindi from English segments and for resolving transliterated forms.

Emoji-Aware Contrastive Learning. Emojis encode rich affective and pragmatic information and are abundant in social media corpora. To inject emoji semantics into the encoder prior to supervised fine-tuning, we introduce an emoji-aware contrastive learning objective defined at the tweet level.

For each tweet in a batch, we extract the set of emojis that appear in the text. For tweets containing at least one emoji, we treat the emoji set as a weak semantic label and construct positive and negative pairs based on emoji overlap. Specifically, two tweets form a positive pair if their emoji sets share at least one common emoji (for example, both contain a red heart ❤️ or both contain an angry face 😠), and a negative pair if their emoji sets are disjoint. This construction treats emoji overlap as a weak semantic signal rather than a perfect label: disjoint emoji sets can still correspond to related content, but over large batches the contrastive objective provides a useful inductive bias that aligns representations along affective and pragmatic dimensions that are consistently expressed through emoji usage. Let $\mathbf{z}_i = g(\mathbf{h}_0^{(i)})$ denote the projected representation of the ith tweet in the batch, where g is a two-layer MLP projection head with a 128-dimensional output, applied to the `[CLS]` embedding $\mathbf{h}_0^{(i)}$, followed by ℓ_2 normalization of $\mathbf{z}_i$.

Given a particular anchor tweet i and one of its positive partners j, we define the contrastive loss using a softmax over cosine similarities within the batch:

$$\mathcal{L}_{\text{contrast}}^{(i)} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k \in \mathcal{N}(i)} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)},$$

where $\text{sim}(\cdot,\cdot)$ is cosine similarity between normalized vectors, $\tau = 0.1$ is the temperature, and $\mathcal{N}(i)$ is the set of all candidate tweets in the batch (including both positive and negative examples). The overall emoji-aware contrastive loss $\mathcal{L}_{\text{contrast}}$ is obtained by averaging $\mathcal{L}_{\text{contrast}}^{(i)}$ over all anchors with at least one positive partner. This objective encourages tweets that share similar emojis to have similar representations in the projected space, while pushing apart representations of tweets with disjoint emoji sets. As a result, the encoder learns to align textual patterns that correspond to similar

affective states (e.g., tweets with predominantly positive emojis such as hearts ❤️ and laughing faces 😂 versus tweets with negative emojis such as angry faces 😡 or crying faces 😢), even without access to manual sentiment or emotion labels.

Table 3 summarizes the three auxiliary pretraining objectives in CodeMixLM, indicating the prediction level, the supervision signal, and the intended effect on the learned representations.

Table 3. Auxiliary pretraining objectives used in CodeMixLM in addition to standard masked language modeling.

Objective	Level	Supervision signal	Intended effect
Masked span denoising	Span (2–5 tokens)	Original subword sequence	Robustness to local noise and irregular forms
Script-aware language ID	Token	Inferred script label (Roman/Devanagari/other)	Explicit encoding of script and language cues
Emoji-aware contrastive learning	Tweet	Emoji set overlap between tweets	Alignment of emoji-related affective semantics

4.3. Joint Pretraining Objective

During continued pretraining, all objectives are optimized jointly on batches of unlabeled code-mixed tweets. The overall pretraining loss for a batch $\mathcal{B}$ is defined as

$$\mathcal{L}_{\text{pre}} = \lambda_{\text{mlm}}\mathcal{L}_{\text{MLM}} + \lambda_{\text{span}}\mathcal{L}_{\text{span}} + \lambda_{\text{script}}\mathcal{L}_{\text{script}} + \lambda_{\text{contrast}}\mathcal{L}_{\text{contrast}},$$

where $\mathcal{L}_{\text{MLM}}$ is the standard masked language modeling loss inherited from the backbone, and $\lambda_{\text{mlm}}, \lambda_{\text{span}}, \lambda_{\text{script}}, \lambda_{\text{contrast}}$ are non-negative scalar weights that control the contribution of each term. In practice, we set these weights so that the different loss components are of comparable scale on the training set, and we optionally anneal some of them over the course of pretraining to emphasize certain objectives at different stages.

We initialize CodeMixLM with the parameters of the multilingual backbone and then continue pretraining on the unlabeled code-mixed corpus for 120,000 update steps (approximately three passes over the filtered corpus at batch size 128) using the combined loss $\mathcal{L}_{\text{pre}}$. Optimization uses AdamW with weight decay 0.01, a peak learning rate of 1×10^{-5}, linear warm-up over the first 10,000 steps, and linear decay to zero thereafter; gradients are clipped to a norm of 1.0. We keep the backbone tokenizer and vocabulary unchanged to maintain compatibility with the multilingual initialization, and we randomly shuffle the unlabeled corpus each epoch to avoid topic or time-based ordering effects. These choices balance adaptation to the Hinglish social-media domain with stability of the multilingual parameters, reducing catastrophic forgetting while still allowing the span-, script-, and emoji-aware objectives to shape the representation space. The resulting encoder is adapted to English–Hindi code-mixed tweets and enriched with span-level denoising capability, explicit script cues, and emoji-informed semantic structure, forming a stronger foundation for the supervised multi-task fine-tuning described in the next section.

5. Multi-Task Fine-Tuning

5.1. Model Architecture

For fine-tuning on SENTIMOJI, we reuse the pretrained CodeMixLM encoder and attach three task-specific classification heads on top of the sequence-level representation $\mathbf{h}_0$ associated with the `[CLS]` token. The encoder is shared across all tasks, while each head is responsible for mapping $\mathbf{h}_0$ into the appropriate label space for multi-label emoji prediction, three-way sentiment classification, and seven-way emotion classification.

The emoji head is designed to handle the inherently multi-label nature of emoji usage: a single tweet can contain both positive and negative facial expressions (for example, a positive smile and a crying face in the same message), as well as non-facial symbols such as hearts or fire icons. To capture this, we use a two-layer feed-forward network with a non-linear activation and dropout, followed by independent sigmoid outputs for each emoji dimension:

$$\hat{\mathbf{y}}^{\text{emo}} = \sigma\left(W_2^{\text{emo}}\,\text{ReLU}(W_1^{\text{emo}}\mathbf{h}_0 + \mathbf{b}_1^{\text{emo}}) + \mathbf{b}_2^{\text{emo}}\right),$$

where $W_1^{\text{emo}}, W_2^{\text{emo}}$ and $\mathbf{b}_1^{\text{emo}}, \mathbf{b}_2^{\text{emo}}$ are trainable parameters. The resulting vector $\hat{\mathbf{y}}^{\text{emo}} \in (0,1)^{|\mathcal{E}|}$ is interpreted as a set of independent probabilities, one for each emoji in the inventory $\mathcal{E}$.

The sentiment head maps $\mathbf{h}_0$ to a distribution over the three sentiment classes $\{\text{positive}, \text{neutral}, \text{negative}\}$. Because sentiment is modeled as a single-label classification task, a simple linear layer with softmax is sufficient:

$$\hat{\mathbf{y}}^{\text{sent}} = \text{softmax}\big(W^{\text{sent}}\mathbf{h}_0 + \mathbf{b}^{\text{sent}}\big),$$

where W^{sent} and $\mathbf{b}^{\text{sent}}$ are parameters of the sentiment head. The output $\hat{\mathbf{y}}^{\text{sent}} \in (0,1)^{|\mathcal{S}|}$ defines a categorical distribution over sentiment labels.

Similarly, the emotion head maps $\mathbf{h}_0$ to a distribution over the seven emotion classes (joy, anger, sadness, fear, disgust, surprise, neutral). We again use a linear layer with softmax:

$$\hat{\mathbf{y}}^{\text{emo2}} = \text{softmax}\big(W^{\text{emo2}}\mathbf{h}_0 + \mathbf{b}^{\text{emo2}}\big),$$

with parameters W^{emo2} and $\mathbf{b}^{\text{emo2}}$. The prediction vector $\hat{\mathbf{y}}^{\text{emo2}} \in (0,1)^{|\mathcal{C}|}$ represents the model's belief over fine-grained emotions such as joy, anger, and sadness.

Table 4 summarizes the task-specific heads used in our fine-tuning setup, their output dimensionality, and the choice of final activation and loss.

Table 4. Task-specific heads used during multi-task fine-tuning. All heads take the shared representation $\mathbf{h}_0$ as input.

Head	Target space	Output / activation	Loss type
Emoji head	$\mathcal{E}$ (e.g., facial and symbolic emojis; $\lvert\mathcal{E}\rvert = 12$)	$\hat{\mathbf{y}}^{\text{emo}} \in (0,1)^{\lvert\mathcal{E}\rvert}$ (sigmoid)	Multi-label logistic (BCE)
Sentiment head	$\mathcal{S} = \{\text{pos}, \text{neu}, \text{neg}\}$	$\hat{\mathbf{y}}^{\text{sent}} \in (0,1)^3$ (softmax)	Categorical cross-entropy
Emotion head	$\mathcal{C}$ (joy, anger, ..., neutral)	$\hat{\mathbf{y}}^{\text{emo2}} \in (0,1)^{\lvert\mathcal{C}\rvert}$ (softmax)	Categorical cross-entropy
Language ID head	$\mathcal{L} = \{\text{English}, \text{Hindi}, \text{Code-mixed}\}$	$\hat{\mathbf{y}}^{\text{lang}} \in (0,1)^3$ (softmax)	Categorical cross-entropy
Hate speech head	$\mathcal{H} = \{\text{hateful}, \text{non-hateful}\}$	$\hat{\mathbf{y}}^{\text{hate}} \in (0,1)^2$ (softmax)	Categorical cross-entropy
Sarcasm detection head	$\mathcal{R} = \{\text{sarcastic}, \text{literal}\}$	$\hat{\mathbf{y}}^{\text{sarc}} \in (0,1)^2$ (softmax)	Categorical cross-entropy
Topic classification head	$\mathcal{T}$ (e.g., politics, sports, entertainment; $\lvert\mathcal{T}\rvert = 8$)	$\hat{\mathbf{y}}^{\text{topic}} \in (0,1)^{\lvert\mathcal{T}\rvert}$ (softmax)	Categorical cross-entropy

In all cases, the encoder and heads are trained end-to-end during fine-tuning. Unless stated otherwise, the main SENTIMOJI experiments use the emoji, sentiment, and emotion heads; the additional heads (language ID, hate speech, sarcasm, and topic) are included only for auxiliary transfer, label-efficiency, and robustness analyses and do not change the primary SENTIMOJI training setup. We optionally apply dropout between W_1^{emo} and W_2^{emo} and after $\mathbf{h}_0$ to reduce overfitting, especially on the emoji head, which must model a relatively large label space from a modest number of annotated examples.

5.2. Training Objective

Let $\mathbf{y}^{\text{emo}} \in \{0,1\}^{|\mathcal{E}|}$ denote the binary multi-hot vector of ground-truth emojis for a given tweet, and let $\mathbf{y}^{\text{sent}} \in \{0,1\}^{|\mathcal{S}|}$ and $\mathbf{y}^{\text{emo2}} \in \{0,1\}^{|\mathcal{C}|}$ denote the corresponding one-hot target vectors for sentiment

and emotion. The emoji loss is defined as a sum of independent binary cross-entropy terms, one for each emoji dimension:

$$\mathcal{L}_{\text{emoji}} = -\sum_{k=1}^{|\mathcal{E}|} \left[y_k^{\text{emo}} \log \hat{y}_k^{\text{emo}} + (1 - y_k^{\text{emo}}) \log(1 - \hat{y}_k^{\text{emo}}) \right],$$

where $\hat{y}_k^{\text{emo}}$ is the predicted probability that the k-th emoji is present. This formulation naturally accommodates tweets with multiple emojis, since each emoji dimension is treated as an independent Bernoulli label.

For sentiment, we use the standard cross-entropy loss for single-label classification:

$$\mathcal{L}_{\text{sent}} = -\sum_{s \in \mathcal{S}} y_s^{\text{sent}} \log \hat{y}_s^{\text{sent}},$$

where $\hat{y}_s^{\text{sent}}$ denotes the predicted probability of sentiment class s. The loss penalizes the model whenever probability mass is assigned to the wrong sentiment, for example when a clearly positive tweet with supportive or happy emojis is misclassified as neutral or negative.

Similarly, for emotion classification we define

$$\mathcal{L}_{\text{emo2}} = -\sum_{c \in \mathcal{C}} y_c^{\text{emo2}} \log \hat{y}_c^{\text{emo2}},$$

where $\hat{y}_c^{\text{emo2}}$ is the predicted probability of emotion class $c \in \mathcal{C}$. This loss encourages the model to distinguish between different negative states (for example, anger versus sadness) and to separate neutral from genuinely emotional content.

The joint multi-task loss combines these three components into a single scalar objective:

$$\mathcal{L}_{\text{multi}} = \alpha \mathcal{L}_{\text{emoji}} + \beta \mathcal{L}_{\text{sent}} + \gamma \mathcal{L}_{\text{emo2}},$$

where $\alpha, \beta, \gamma \geq 0$ are weighting coefficients that control the relative influence of each task during training. In practice, we normalize each loss by the number of labels for that task (i.e., by $|\mathcal{E}|$ for emoji, by $|\mathcal{S}|$ for sentiment, and by $|\mathcal{C}|$ for emotion) so that magnitudes are comparable across objectives. We then set the global weights to $\alpha = 1.0$, $\beta = 0.5$, and $\gamma = 0.5$ based on validation performance, reflecting the fact that emoji prediction is the primary target task while sentiment and emotion provide complementary auxiliary supervision. We keep these weights fixed across all reported runs to avoid tuning to the test set and to ensure that gains can be attributed to the pretraining objectives rather than to per-experiment reweighting.

Table 5 illustrates a representative configuration of loss weights and normalizations used in our experiments.

Table 5. Example configuration of multi-task loss weighting and normalization. $Z_\cdot$ denotes the normalization factor (number of labels) for each task.

Task	Raw loss	Normalization	Global weight
Emoji prediction	$\mathcal{L}_{\text{emoji}}$	$Z_{\text{emoji}} = \lvert\mathcal{E}\rvert$	α (1.0)
Sentiment classification	$\mathcal{L}_{\text{sent}}$	$Z_{\text{sent}} = \lvert\mathcal{S}\rvert$	β (0.5)
Emotion classification	$\mathcal{L}_{\text{emo2}}$	$Z_{\text{emo2}} = \lvert\mathcal{C}\rvert$	γ (e.g., 0.7)

We fine-tune all encoder and head parameters jointly using AdamW with a linearly decaying learning rate and weight decay for regularization. Early stopping is applied based on validation macro-F_1 for emoji prediction, as this task is the most challenging and exhibits the strongest benefits from multi-task learning. Monitoring emoji macro-F_1 also helps ensure that the model does not overfit to the comparatively easier sentiment labels at the expense of correctly modeling the richer multi-label emoji distribution.

6. Experimental Setup

6.1. Baselines

We compare CodeMixLM against several competitive baselines that represent standard practices for modeling code-mixed text with transformers. The first baseline consists of *single-task transformers*. In this setting, we fine-tune a separate XLM-R base model for each task: one model is trained only for multi-label emoji prediction, another only for three-way sentiment classification, and a third only for seven-way emotion classification. Each model uses the same encoder architecture as CodeMixLM but does not benefit from joint training across tasks or from the additional code-mixed self-supervised pretraining. This baseline isolates the effect of multi-task learning and domain-adaptive pretraining by showing how well a strong multilingual transformer performs when applied in the most straightforward way.

The second baseline is a *multi-task transformer* with a shared XLM-R base encoder and three task-specific heads, analogous to the fine-tuning configuration used for CodeMixLM. In this case, however, the backbone is not further pretrained on unlabeled code-mixed tweets; instead, it is used as released and fine-tuned jointly on SENTIMOJI for emoji, sentiment, and emotion. This baseline captures the gains attributable to multi-task learning alone, without any domain adaptation. Comparing CodeMixLM to this model highlights the added value of self-supervised pretraining on code-mixed data and emoji-aware objectives.

The third baseline is a *prior task-specific encoder* originally proposed for multi-label emoji, sentiment, and emotion prediction on SENTIMOJI. This architecture replaces self-attention with linear transformations and employs gated linear units together with autoencoder-based pretraining on the labeled dataset. When code and hyperparameter settings are available, we reimplement this model and train it in the same multi-task setting as CodeMixLM. This baseline represents the state of the art proposed specifically for SENTIMOJI using a bespoke encoder, and provides a strong point of comparison for our transformer-based approach.

For a fair comparison across all models, we keep the total number of parameters within a similar range by using the same XLM-R base backbone wherever applicable and by restricting the size of task-specific heads. We also use the same training, validation, and test splits for all methods and apply consistent preprocessing, tokenization, and batching strategies.

6.2. Evaluation Metrics

We evaluate model performance using metrics that reflect both overall classification quality and more fine-grained behavior on imbalanced labels. For multi-label emoji prediction, we report micro- and macro-averaged F_1 scores. The micro-F_1 score aggregates decisions over all emoji instances, emphasizing performance on frequent emojis, while the macro-F_1 score averages F_1 across emoji classes, treating rare and frequent emojis equally and thus providing a more balanced view of performance.

In addition, we compute coverage error and label ranking loss, which are standard metrics for multi-label problems: coverage error measures how far down the ranked list of predicted emojis one must go to cover all true labels, and label ranking loss quantifies the fraction of label pairs that are incorrectly ordered by the model. To better understand performance on individual emojis, we also report per-emoji F_1 scores, highlighting improvements or degradations for rare emojis.

For sentiment and emotion classification, which are single-label tasks, we report overall accuracy, macro-F_1, and per-class F_1. Accuracy provides a coarse measure of how often the model's top prediction matches the ground truth, while macro-F_1 accounts for class imbalance by averaging F_1 across sentiment or emotion classes. Per-class F_1 scores allow us to examine whether CodeMixLM particularly benefits minority classes such as rare emotions or negative sentiment. Beyond accuracy and F_1, we compute expected calibration error (ECE) as a diagnostic of whether predicted confidence scores correspond to empirical correctness frequencies. Because the primary goal of this study is comparative representation learning for affective prediction, we do not use ECE for model selection and therefore focus the main tables on accuracy and F_1 metrics, while using ECE to verify that performance improvements are not achieved at the cost of extreme miscalibration.

Table 6 provides an overview of the metrics used for each task and the aspects of model behavior they capture.

Table 6. Evaluation metrics used for each task. F_1 refers to the harmonic mean of precision and recall.

Task	Metrics	Purpose
Emoji prediction	Micro-F_1, macro-F_1	Overall and balanced multi-label performance
Emoji prediction	Coverage error, label ranking loss	Quality of ranked emoji predictions
Emoji prediction	Per-emoji F_1	Behavior on frequent vs rare emojis
Sentiment classification	Accuracy, macro-F_1, per-class F_1	Correctness and balance across sentiment classes
Emotion classification	Accuracy, macro-F_1, per-class F_1	Correctness and balance across emotion classes
All tasks	Expected calibration error (ECE)	Alignment of predicted probabilities with empirical correctness

6.3. Implementation Details

We tokenize tweets using the backbone tokenizer associated with the XLM-R model and apply light normalization consistent with the backbone (lowercasing, URL normalization, and whitespace cleanup), while deliberately retaining emojis and user-style spelling variation. Each tweet is truncated or padded to a maximum sequence length of 64 subword tokens, which covers the vast majority of posts in SENTIMOJI while keeping computation efficient and comparable across models. During self-supervised pretraining of CodeMixLM, masking is performed dynamically: standard MLM masks 15% of subword tokens, the masked span denoising objective samples non-overlapping spans of length 2–5 with $p_{\text{span}} = 0.15$ under the same overall corruption budget, and the script-aware objective predicts script labels on a uniformly sampled 15% of token positions. For emoji-aware contrastive learning, we form in-batch positives based on emoji overlap, use a 128-dimensional projection head, and set $\tau = 0.1$; tweets without emojis are excluded from the contrastive loss but still contribute to MLM and the span/script objectives.

For supervised fine-tuning on SENTIMOJI, we use mixed-task mini-batches to balance the contributions of the active tasks. Concretely, each update concatenates 32 instances from each of the three primary tasks (emoji, sentiment, emotion), yielding an effective batch size of 96; gradients are computed on the combined loss $\mathcal{L}_{\text{multi}}$ and backpropagated through the shared encoder. This sched-

ule ensures that no single task dominates the updates simply because of label density or relative ease. All models are trained with AdamW (weight decay 0.01) using a linear learning-rate schedule with 10% warm-up. We set the peak learning rate to 2×10^{-5}, train for up to 10 epochs with early stopping (patience 2) based on validation emoji macro-F_1, and use dropout 0.1 throughout. For multi-label emoji prediction, we tune a single global decision threshold on the validation set to maximize macro-F_1 and then fix it for all test-set evaluations to avoid optimistic bias. To reduce sensitivity to initialization, each configuration is run with five random seeds and we report the mean test performance of the validation-selected checkpoints.

Table 7 summarizes the key hyperparameters and training settings used for the final reported runs; any tuning is performed on the validation set and does not change the evaluation protocol or reported metrics.

Table 7. Key hyperparameters and training settings for CodeMixLM and transformer baselines. Values shown are typical configurations; small variations may be used during tuning.

Setting	Value	Description
Max sequence length	64 subwords	Truncation / padding length for tweets
Optimizer	AdamW	Weight-decayed Adam optimizer
Initial learning rate	2×10^{-5}	Peak LR with linear decay
Batch size (pretraining)	128 tweets	Unlabeled code-mixed tweets per batch
Batch size (fine-tuning)	32 tweets	Labeled tweets per batch
Dropout rate	0.1	Applied to encoder and heads
Loss weights	$\alpha = 1.0, \beta = 0.5, \gamma = 0.5$	Task weights in $\mathcal{L}_{\text{multi}}$
Early stopping criterion	Emoji macro-F_1 (validation)	Used to select the best checkpoint

7. Results

7.1. Overall Performance

Table 8 summarizes the main test-set performance for all models on the three prediction tasks defined in Section 3. The table reports macro- and micro-averaged F_1 scores for multi-label emoji prediction, and macro-F_1 scores for sentiment and emotion classification. In this paper, "state-of-the-art" refers to the best performance among the strong multilingual transformer baselines and the prior task-specific encoder included for direct comparison on the same data and evaluation protocol.

The table reports the primary SENTIMOJI test-set results. Across all three benchmark tasks, CodeMixLM achieves the best scores among the compared systems, indicating that continued code-mixed pretraining yields consistent downstream gains rather than improvements isolated to a single label space. For multi-label emoji prediction, CodeMixLM improves macro-F_1 from 0.654 (multi-task XLM-R) to 0.683 and micro-F_1 from 0.701 to 0.724, showing benefits both for frequent emojis (micro-F_1) and for under-represented classes (macro-F_1). These gains are practically meaningful in a highly skewed label setting where improvements on rare emojis are difficult to obtain from supervised fine-tuning alone.

For sentiment and emotion, CodeMixLM also yields clear improvements in macro-F_1 (sentiment: 0.776 vs. 0.742; emotion: 0.594 vs. 0.562), supporting the claim that the proposed self-supervised

Table 8. Main results on the SENTIMOJI test set. CodeMixLM achieves state-of-the-art performance across all tasks and metrics.

Model	Emoji Macro-F_1	Emoji Micro-F_1	Sentiment Macro-F_1	Emotion Macro-F_1
Single-task XLM-R	0.642	0.687	0.728	0.543
Multi-task XLM-R	0.654	0.701	0.742	0.562
Prior encoder (multi-task)	0.598	0.645	0.705	0.518
CodeMixLM (ours)	**0.683**	**0.724**	**0.776**	**0.594**

objectives learn affect-sensitive representations that transfer beyond emoji prediction. Importantly, the comparison between single-task and multi-task transformer baselines shows that joint training itself is beneficial: multi-task XLM-R consistently outperforms its single-task counterpart on all three metrics (e.g., emoji macro-F_1 0.654 vs. 0.642), consistent with the intuition that sentiment and emotion supervision helps disambiguate emoji usage and that emoji supervision provides an additional affective signal for the auxiliary tasks. CodeMixLM builds on this multi-task benefit by starting from an encoder already adapted to code-mixed social media through span-level denoising, explicit script cues, and emoji-informed contrastive structure.

Finally, the prior task-specific encoder underperforms both transformer baselines and CodeMixLM, suggesting that simply engineering an architecture for SENTIMOJI is insufficient to match the representational advantages conferred by large-scale self-supervised pretraining. Taken together, these results support the central hypothesis that domain-specialized self-supervised learning is a key ingredient for robust code-mixed affective prediction.

7.2. Label Efficiency

To assess label efficiency, we fine-tune each model using only a fraction of the labeled training data while keeping the validation and test sets fixed. We consider 10%, 25%, 50%, and 100% of the training examples, subsampling uniformly at random within each class distribution to avoid introducing artificial label skew. For each fraction, we use the same optimization and early-stopping protocol as in the full-data setting, selecting checkpoints on the validation set and then evaluating on the held-out test set.

Table 9 summarizes how performance scales with supervision for emoji prediction, sentiment, emotion, and the auxiliary hate-speech task. The pattern is consistent across tasks: CodeMixLM provides the largest gains in the low-data regime and maintains an advantage even at full supervision. For example, at 10% training data, CodeMixLM improves emoji macro-F_1 from 0.451 (multi-task XLM-R) to 0.512, and at 25% it improves from 0.539 to 0.592, indicating that continued code-mixed pretraining provides a strong initialization when labeled annotations are scarce. For sentiment we report accuracy in this analysis (as in the robustness experiments) because it remains stable under aggressive subsampling; CodeMixLM improves sentiment accuracy from 0.612 to 0.658 at 10% data and from 0.678 to 0.715 at 25% data. Emotion and hate-speech results show analogous trends, suggesting that the learned representations transfer to related affective and safety-relevant classification problems rather than overfitting to a single benchmark label space.

Table 9. Label-efficiency analysis showing performance scaling with training data size. Results show macro-F_1 scores across different data fractions.

Model	10% data	25% data	50% data	100% data
Emoji Prediction (macro-F_1)				
Single-task XLM-R	0.423	0.518	0.587	0.642
Multi-task XLM-R	0.451	0.539	0.602	0.654
Prior encoder	0.385	0.476	0.543	0.598
CodeMixLM	**0.512**	**0.592**	**0.641**	**0.683**
Sentiment Analysis (accuracy)				
Single-task XLM-R	0.587	0.654	0.702	0.738
Multi-task XLM-R	0.612	0.678	0.721	0.752
CodeMixLM	**0.658**	**0.715**	**0.749**	**0.781**
Emotion Classification (macro-F_1)				
Single-task XLM-R	0.348	0.432	0.498	0.543
Multi-task XLM-R	0.376	0.458	0.517	0.562
CodeMixLM	**0.425**	**0.501**	**0.552**	**0.594**
Hate Speech Detection (macro-F_1)				
Single-task XLM-R	0.512	0.598	0.654	0.692
Multi-task XLM-R	0.543	0.621	0.672	0.708
CodeMixLM	**0.587**	**0.658**	**0.701**	**0.734**

7.3. Robustness to Noise

We examine robustness to input noise that mimics realistic user behavior and code-mixing variability by introducing controlled synthetic perturbations into the test set while keeping labels fixed. First, we randomly delete vowels in Roman-script tokens (e.g., "yaar"→"yr"), simulating common informal shortening and spelling variation. Second, we swap nearby Hindi and English tokens to shift code-switch boundaries, testing whether models rely excessively on local word order or brittle surface cues.

As shown in the table, all models degrade under perturbations, but CodeMixLM is consistently more resilient. For emoji prediction, CodeMixLM drops by 9.1% under vowel deletion compared with 16.2% for multi-task XLM-R, and by 3.7% under code-switch swapping compared with 7.9% for multi-task XLM-R. A similar pattern holds for emotion (8.9% vs. 18.5% under vowel deletion) and for the auxiliary hate-speech task (7.6% vs. 15.5%). These improvements align with the design of the pretraining objectives: masked span denoising explicitly trains reconstruction under local corruption, while script-aware prediction encourages the encoder to represent cross-script information in a way that is less sensitive to orthographic variation. The smaller relative drops therefore provide direct evidence that the proposed self-supervised pretraining produces representations that are better matched to the noise and code-switching phenomena that characterize Hinglish social media.

Table 10 outlines the comparison of performance on clean versus perturbed test sets for each model.

Overall, the label-efficiency and robustness analyses reinforce the main findings from the full-

Table 10. Robustness analysis comparing performance on clean versus perturbed test sets. Results show macro-F_1 scores across different noise conditions.

Model	Clean	Vowel deletion	Code-switch swap
Emoji Prediction (macro-F_1)			
Single-task XLM-R	0.642	0.521 (-18.8%)	0.587 (-8.6%)
Multi-task XLM-R	0.654	0.548 (-16.2%)	0.602 (-7.9%)
Prior encoder	0.598	0.478 (-20.1%)	0.543 (-9.2%)
CodeMixLM	**0.683**	**0.621 (-9.1%)**	**0.658 (-3.7%)**
Sentiment Analysis (accuracy)			
Single-task XLM-R	0.738	0.634 (-14.1%)	0.692 (-6.2%)
Multi-task XLM-R	0.752	0.658 (-12.5%)	0.708 (-5.9%)
CodeMixLM	**0.781**	**0.724 (-7.3%)**	**0.763 (-2.3%)**
Emotion Classification (macro-F_1)			
Single-task XLM-R	0.543	0.432 (-20.4%)	0.498 (-8.3%)
Multi-task XLM-R	0.562	0.458 (-18.5%)	0.517 (-8.0%)
CodeMixLM	**0.594**	**0.541 (-8.9%)**	**0.578 (-2.7%)**
Hate Speech Detection (macro-F_1)			
Single-task XLM-R	0.692	0.567 (-18.1%)	0.634 (-8.4%)
Multi-task XLM-R	0.708	0.598 (-15.5%)	0.658 (-7.1%)
CodeMixLM	**0.734**	**0.678 (-7.6%)**	**0.718 (-2.2%)**

data experiments: CodeMixLM not only achieves higher peak performance on emoji, sentiment, and emotion prediction, but also learns representations that are more data-efficient and more resilient to typical sources of noise in code-mixed social media.

8. Discussion

8.1. Implications for Code-Mixed Representation Learning

The results of this study highlight the importance of domain-specialized self-supervised learning for code-mixed NLP. While generic multilingual transformers like XLM-R provide a strong starting point, they are trained predominantly on clean, monolingual text and under-represent noisy code-mixed social media. By continuing pretraining on unlabeled Hinglish tweets and incorporating objectives tailored to local noise, script variation, and emoji semantics, CodeMixLM learns representations that are better aligned with the linguistic and pragmatic realities of English–Hindi code-mixing.

From a broader perspective, the approach we advocate—domain-adaptive pretraining with task-aligned self-supervised objectives, followed by multi-task fine-tuning—extends naturally to other code-mixed scenarios documented in the literature [3]. For example, similar techniques could be applied to Bengali–English, Tamil–English, or Spanish–English code-mixed data, using language- and domain-specific pretraining corpora and adapting the auxiliary objectives to reflect the available weak supervision signals (such as hashtags, mentions, and platform-specific reactions).

8.2. Emojis as Weak Supervision for Affective Computing

Our findings also reinforce prior evidence that emojis provide a valuable source of weak supervision for affective computing [8, 15]. Unlike traditional sentiment labels, which require manual annotation, emojis are abundant and naturally aligned with users' affective states, albeit in a noisy and context-dependent way. By incorporating emoji-aware contrastive learning during pretraining, we leverage this signal without requiring explicit labels; tweets that share similar emoji usage are encouraged to have similar representations, smoothing the representation space along dimensions related to affect and stance.

In the code-mixed setting, emojis play an additional role as a stabilizing signal that can partially compensate for ambiguity introduced by transliteration, non-standard spellings, and code-switching. When textual cues are ambiguous due to language mixing, the presence or absence of particular emojis can help anchor the model's interpretation of sentiment and emotion. This suggests that emoji-aware pretraining may be particularly beneficial in low-resource settings where linguistic cues are noisy and training data is limited.

8.3. Data Efficiency and Robustness

The label-efficiency experiments indicate that domain-adaptive pretraining and multi-task learning substantially reduce the amount of labeled data required to achieve a given level of performance, especially for the more complex multi-label emoji prediction task. This is encouraging for practitioners who wish to build robust models for new code-mixed domains but face constraints on manual annotation. Collecting a large unlabeled corpus of code-mixed text with emojis and other weak supervision signals is typically much cheaper than collecting high-quality sentiment and emotion labels, and our results suggest that such unlabeled data can be effectively exploited to improve downstream performance.

The robustness experiments further show that CodeMixLM's representations are less brittle under realistic perturbations such as character deletions and modest changes in code-switch boundaries. This robustness is critical in deployment, where models will inevitably encounter spelling variants, creative transliterations, and style variations that were underrepresented in the labeled training data. The masked span denoising and script-aware objectives appear to be key contributors to this robustness, providing a template for designing self-supervised objectives that explicitly target common sources of noise in social media.

References

[1] Gumperz, J. J. (1982). *Discourse Strategies* (No. 1). Cambridge University Press.

[2] Bali, K., Sharma, J., Choudhury, M., & Vyas, Y. (2014, October). "i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching* (pp. 116-126).

[3] Sitaram, S., Raghavi Chandu, K., Krishna Rallabandi, S., & Black, A. W. (2019). A Survey of Code-switched Speech and Language Processing. arXiv e-prints, arXiv-1904.

[4] Sharma, A., Gupta, S., Motlani, R., Bansal, P., Shrivastava, M., Mamidi, R., & Sharma, D. M. (2016, June). Shallow Parsing Pipeline-Hindi-English Code-Mixed Social Media Text. In *Proceedings of the*

2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1340-1345).

[5] Vyas, Y., Gella, S., Sharma, J., Bali, K., & Choudhury, M. (2014, October). Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (EMNLP) (pp. 974-979).

[6] Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PloS one, 10*(12), e0144296.

[7] Barbieri, F., Kruszewski, G., Ronzano, F., & Saggion, H. (2016, October). How cosmopolitan are emojis? Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 24th ACM international conference on Multimedia* (pp. 531-535).

[8] Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., & Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv preprint arXiv:1708.00524.

[9] Prabhu, A., Joshi, A., Shrivastava, M., & Varma, V. (2016). Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. arXiv preprint arXiv:1611.00472.

[10] Patwa, P., Aguilar, G., Kar, S., Pandey, S., Pykl, S., Gambäck, B., ... & Das, A. (2020, December). SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 774-790).

[11] Singh, G. V., Ghosh, S., Firdaus, M., Ekbal, A., & Bhattacharyya, P. (2024). Predicting multi-label emojis, emotions, and sentiments in code-mixed texts using an emojifying sentiments framework. *Scientific Reports, 14*(1), 12204.

[12] Zampieri, M., Ranasinghe, T., Sarkar, D., & Ororbia, A. (2023). Offensive language identification with multi-task learning. *Journal of Intelligent Information Systems, 60*(3), 613-630.

[13] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098.

[14] Srivastava, V., & Singh, M. (2021). Challenges and limitations with the metrics measuring the complexity of code-mixed text. arXiv preprint arXiv:2106.10123.

[15] Saggion, H., Ballesteros, M., & Barbieri, F. (2017). Are emojis predictable. *Proc. Meet. Eur. Chapter Assoc. Comput. Linguistics*, 105-111.

[16] Chen, Z., Cao, Y., Yao, H., Lu, X., Peng, X., Mei, H., & Liu, X. (2021). Emoji-powered sentiment and emotion detection from software developers' communication data. *ACM Transactions on Software Engineering and Methodology (TOSEM), 30*(2), 1-48.

[17] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). 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) (pp. 4171-4186).

[18] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2020, July). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440-8451).

[19] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks. arXiv preprint arXiv:2004.10964.

[20] Barman, U., Das, A., Wagner, J., & Foster, J. (2014, October). Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching* (pp. 13-23).

[21] Solorio, T., Blair, E., Maharjan, S., Bethard, S., Diab, M., Ghoneim, M., ... & Fung, P. (2014, October). Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching* (pp. 62-72).

[22] Aguilar, G., Kar, S., & Solorio, T. (2020, May). LinCE: A Centralized Benchmark for Linguistic Code-switching Evaluation. In *Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 1803-1813).

[23] Bohra, A., Vijay, D., Singh, V., Akhtar, S. S., & Shrivastava, M. (2018, June). A dataset of Hindi-English code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media* (pp. 36-41).

[24] Mandl, T., Modha, S., Majumder, P., Patel, D., Dave, M., Mandlia, C., & Patel, A. (2019, December). Overview of the HASOC track at FIRE 2019. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*. ACM.

[25] Eisner, B., Rocktäschel, T., Augenstein, I., Bosnjak, M., & Riedel, S. (2016, November). emoji2vec: Learning emoji representations from their description. In *Proceedings of the Fourth International Workshop on Natural Language Processing for Social Media* (pp. 48-54).

[26] Mahdian, M., Arianfar, S., Gibson, J., & Oran, D. (2016, September). MIRCC. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM.

[27] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[28] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog, 1*(8), 9.

[29] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877-1901.

[30] Giorgi, J., Nitski, O., Wang, B., & Bader, G. (2021, August). Declutr: Deep contrastive learning for unsupervised textual representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing* (Volume 1: Long Papers) (pp. 879-895).

[31] Nguyen, D. Q., Vu, T., & Nguyen, A. T. (2020, October). BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 9-14).

How to cite this article: Asif Shehzad, Rahul Sharma and Pushpak Bhattacharyya (2025). Self-Supervised Code-Mixed Representation Learning for Multi-Label Emoji, Sentiment, and Emotion Prediction. *Bulletin of Computer and Data Sciences*, 6(1), 39-60. DOI: 10.71448/bcds2561-3

Received: 10/09/2024 **Revised:** 26/12/2024 **Accepted:** 21/02/2025 **Publish:** 30/03/2025

Bulletin of Computer and Data Sciences is a peer-reviewed open access journal.