Inceptive Transformers: Enhancing Contextual Representations through Multi-Scale Feature Learning Across Domains and Languages

Asif Shahriar[1,2], Rifat Shahriyar[1], M Saifur Rahman[1]
[1]Bangladesh University of Engineering and Technology, [2]BRAC University
asif.shahriar@bracu.ac.bd, {rifat, mrahman}@cse.buet.ac.bd

Abstract

Encoder transformer models compress information from all tokens in a sequence into a single `[CLS]` token to represent global context. This approach risks diluting fine-grained or hierarchical features, leading to information loss in downstream tasks where local patterns are important. To remedy this, we propose a lightweight architectural enhancement: an inception-style 1-D convolution module that sits on top of the transformer layer and augments token representations with multi-scale local features. This enriched feature space is then processed by a self-attention layer that dynamically weights tokens based on their task relevance. Experiments on five diverse tasks show that our framework consistently improves general-purpose, domain-specific, and multilingual models, outperforming baselines by 1% to 14% while maintaining efficiency. Ablation studies show that multi-scale convolution performs better than any single kernel and that the self-attention layer is critical for performance.

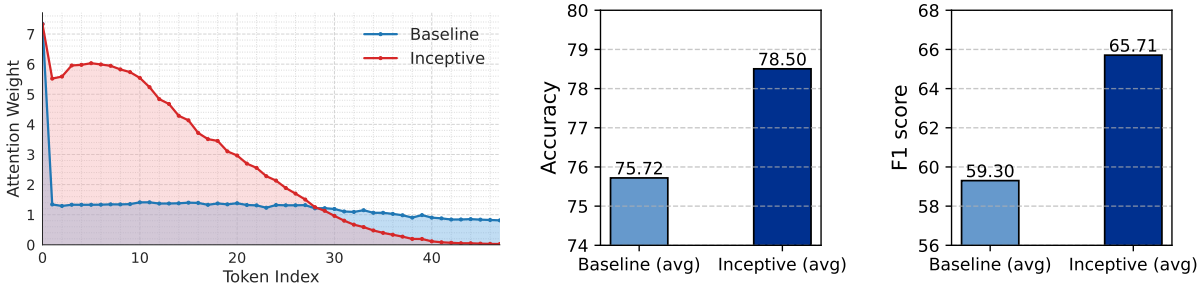
1 Introduction

Since its introduction, the transformer architecture (Vaswani et al., 2017) has revolutionized the field of natural language processing (NLP). Encoder-based transformer models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), XLNet (Yang et al., 2019), Electra (Clark et al., 2020), DeBERTa v3 (He et al., 2023), and ModernBERT (Warner et al., 2024) have demonstrated impressive performance across a wide range of NLP tasks. In addition to these general-purpose models, a number of domain-specific BERT-based models like BioBERT (Lee et al., 2019), SciBERT (Beltagy et al., 2019), LegalBERT (Chalkidis et al., 2020), BERTweet (Nguyen et al., 2020) have emerged, which are further pre-trained on domain-specific corpora to capture the unique language, terminology, and stylistic features of various specialized fields. In parallel, cross-lingual models like XLM-R (Conneau et al., 2020) and language-specific models such as BanglaBERT (Bhattacharjee et al., 2022) have extended this architecture to support diverse linguistic settings, including low-resource languages like Bangla. Despite the emergence of LLMs, fine-tuned encoder models remain state-of-the-art in non-generative tasks like classification and NER (Edwards and Camacho-Collados, 2024; Sun et al., 2023; Chen et al., 2025).

All these encoder models are designed to aggregate all token embeddings into a single representation, called the `[CLS]` token, which is later used for downstream tasks like classification. Although convenient, this approach of collapsing an entire text sequence with multiple aspects into one single embedding can cause information loss (Chang et al., 2023), particularly when there are short-range dependencies between tokens (Guo et al., 2019; Li et al., 2021). During experiments, we observed that the over-reliance on `[CLS]` token makes the encoder models insufficient in capturing fine-grained contextual nuances or localized cues critical for tasks like emotion recognition or irony detection (Fig. 1a). This issue is even more pronounced in multi-label tasks, which require token-level attention rather than a single sequence-level summary.

To address these limitations, we propose *Inceptive Transformers* – a lightweight and modular architecture that augments a transformer baseline by stacking an inception-style 1-D convolution module on top. Instead of using `[CLS]`-based pooling, we feed the final hidden states from the baseline transformer (e.g. RoBERTa or BioBERT) to a multi-scale feature extraction module inspired by inception networks. This module employs parallel 1-D convolutional filters with varying kernel sizes that are designed to recognize local features, such as key phrases or word combinations that are indicative of specific classifications. The goal of the inception module is to incorporate local features without sacrificing global context, which is

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 25845–25860



(a) Baseline vs Inceptive DeBERTa v3 (Irony) (b) Irony Detection (c) OHSUMED

Figure 1: Encoder models like DeBERTa over-rely on the [CLS] token, whereas inceptive DeBERTa redistributes attention dynamically based on task contribution (1a). Experiments show the superior performance of inceptive models (1b and 1c).

achieved by using a residual connection to concatenate the original transformer's hidden states with the multi-scale features. These enriched features are then processed by a self-attention mechanism, which dynamically assigns weights to tokens based on their task-specific contribution, thus allowing the model to effectively prioritize relevant tokens.

As an illustration, let us consider the ironic text: *"work Wednesday–Sunday... #yay #not #moneytho"*. BERTweet fails to identify it as ironic because the dominant neutral phrase (*"work Wednesday–Sunday"*) outweighs the local sarcastic hashtags when collapsed into a single `[CLS]` representation. In contrast, Inceptive BERTweet correctly predicts irony because its multi-scale convolution and post-hoc attention redistribute focus toward the localized cues (*'#yay', '#not'*), capturing the contradiction that signals sarcasm.

Our experiments demonstrate that Inceptive Transformers consistently outperform both general-purpose (RoBERTa, DeBERTa v3, ModernBERT, XLM-R) and domain-specific (BERTweet, BioBERT, CT-BERT, BanglaERT) baselines. On five different tasks (Bangla and English emotion recognition, irony detection, disease identification, and anti-vaccine concern classification), we observed performance gains from **1%** to as high as **14%**, with less than **10%** inference-time overhead. Attention maps show that inceptive models redistribute attention from [CLS] toward task-relevant spans, increasing token coverage and mitigating over-reliance on any single token.

Major contributions of our work are as follows.

- **A Novel, Modular Architecture.** We introduce a lightweight, modular, plug-and-play architecture for enriching the contextual representations of transformer models through a post-hoc multi-scale feature extraction module that can be integrated into any pre-trained encoder model without costly re-training. To the best of our knowledge, this has not been explored before.
- **A Generalizable Framework.** We propose a domain- and backbone-agnostic framework that can be applied on top of any encoder, including general-purpose, domain-specific, and cross-lingual models, without any model-specific adjustments. Furthermore, through comprehensive evaluations, we show that our inceptive models perform strongly in four different datasets from diverse domains and languages – highlighting its general applicability.
- **Rigorous Validation.** We validate the effectiveness of our method through a rigorous empirical evaluation that includes statistical significance testing across multiple runs and extensive ablation studies for isolating the impact of each architectural component.

2 Related Work

Text classification models range from traditional machine learning approaches like decision trees (Law and Ghosh, 2022), support vector machines (SVM), k-nearest neighbors (KNN) (Hanifelou et al., 2018), and ensemble learning (Zhu et al., 2023; Wu et al., 2016), to more advanced deep learning techniques like RNN and LSTM (Lai et al., 2015; Onan, 2022). Convolution networks have also been used (Conneau et al., 2017; Choi et al., 2019; Yao et al., 2019; Soni et al., 2022), but they often struggle to capture long-range dependencies in text.

Combining convolution with transformers. After the transformer architecture (Vaswani et al.,

2017) was introduced, many works have combined convolution with transformers, but these works mostly focus on vision related tasks (Fang et al., 2022; Si et al., 2022; Yuan et al., 2023). Application on NLP domain remains limited to a few works (Zheng and Yang, 2019; Wan and Li, 2022; Chen et al., 2022; Wu et al., 2024) — which mostly focus on improving a particular transformer model, like BERT or XLNet. In comparison, we provide a general architecture capable of improving different types of transformer models, both domain-specific and general-purpose.

Modifications of BERT-like models. Several works modify BERT through architectural or pre-training adaptations to better suit specific tasks or domains, including SpanBERT (Joshi et al., 2020), StructBERT (Wang et al., 2019), and CodeBERT (Feng et al., 2020). Other works such as MT-DNN (Liu et al., 2019a) introduce multi-task learning objectives on top of BERT, while KnowBERT (Peters et al., 2019) integrates external knowledge bases into BERT's architecture. Our work is orthogonal to these efforts: instead of modifying the pre-training strategy, we propose an architectural enhancement that can be directly plugged into existing BERT-like models.

Encoder vs LLMs. While LLMs have shown impressive generative capabilities in recent years, a number of studies demonstrate that encoder models still perform better on supervised non-generative tasks like classification and NER. Edwards and Camacho-Collados (2024) compared in-context prompting with LLaMA/Flan-T5 against fine-tuning encoder models like RoBERTa on 16 datasets and found that fine-tuned encoder models performed better. Sun et al. (2023) showed that even with few-shot prompting and chain-of-thought reasoning, LLMs underperformed compared to fine-tuned models like RoBERTa and XLNet in classification tasks. Chen et al. (2025) found that encoder models like BioBERT and PubMedBERT remain state-of-the-art in biomedical domain tasks like document classification, NER, and relation extraction — significantly outperforming LLMs like GPT-4 and LLaMA 2 13B.

3 Inceptive Transformer

3.1 Motivation

Transformer-based models rely on token-level embeddings derived primarily from self-attention layers to capture global dependencies and context within text sequences. In our experiment, we visualized the attention maps of these models in Section 5.4, which show a strong bias in attention towards the `[CLS]` token, while intermediate tokens often receive comparatively lower attention. The `[CLS]` token is a weighted aggregation of all token embeddings in the sequence, which the model relies on to represent the entire sequence for classification tasks. This bias suggests an underutilization of contextual and local dependencies, potentially limiting the model's ability to effectively capture fine-grained patterns and hierarchical structures present in textual data.

Our model is designed to address this gap by incorporating convolutional operations, which excel at capturing local patterns and hierarchical structures in data (Gu et al., 2018; Li et al., 2022). CNNs are typically not used on textual data due to their inability to capture long-range dependencies. However, using convolution makes sense in our model because it operates on embeddings generated by a transformer— not on raw text. This allows the convolutional operations to refine the already globally contextualized embeddings by emphasizing fine-grained, local features that might otherwise be overlooked. Furthermore, instead of using a single convolution layer with a fixed kernel size, we use an inception module (Szegedy et al., 2015) to apply convolutions with multiple kernel sizes to learn features at different levels of granularity– capturing both token-level patterns and phrase-level dependencies.

The applicability of our model is not limited to general-purpose transformers like RoBERTa. Domain-specific pre-trained models such as BioBERT, CT-BERT, or BERTweet show similar attention biases as BERT and RoBERTa, leading to challenges in capturing local and hierarchical dependencies. By integrating our model's multi-scale feature extraction approach, these domain-specific variants can also be enhanced, improving their ability to represent diverse patterns within specialized input data. This versatility makes our model a robust addition to any transformer-based architecture.

3.2 Model Architecture

The full workflow of our inceptive models is illustrated in Fig.2. The input to our model is pre-processed text data, which need to be tokenized using an appropriate pre-trained tokenizer corresponding to the chosen transformer model. Mathematically, the input can be represented as $X =$

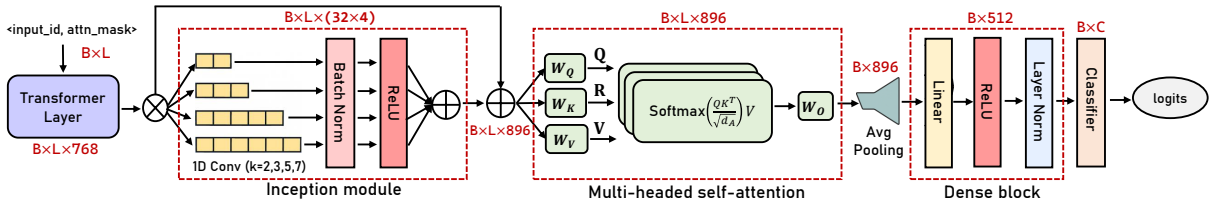


Figure 2: End-to-end architecture of the Inceptive Transformer framework. Output shapes are annotated for each main component. B : batch size, L : input sequence length, C : number of classes. In this figure we have used 768 as hidden state dimension of BERT-like transformer models, 32 as output channels for each convolution branch, and 512 as the target dimensionality of the linear layer in dense block. W_Q, W_K, W_V, W_O are learnable projection matrices, and d_A is the dimension of each attention head.

$[x_1, x_2, \ldots, x_L]$ where L is the sequence length, and each x_i corresponds to a token from the text. X is passed to the transformer layer.

3.2.1 Transformer Layer

The first layer of our architecture is a transformer-based model, such as RoBERTa or BioBERT. Given input X, the transformer layer generates a tensor of hidden states $H \in \mathbb{R}^{B\times L\times d}$ where B is the batch size, L is the sequence length, and d is the hidden state dimension. We denote $H[b, i, :] = h_i^{(b)} \in \mathbb{R}^d$ as the contextual embedding for the i-th token in the b-th input. A dropout layer is applied to H to prevent overfitting.

3.2.2 Inception Module

The primary task of this layer is to extract multi-scale local features. The inception module receives contextual embeddings H generated by the transformer and applies parallel convolutional layers with small kernel sizes k (e.g., $k = 2, 3, 5, 7$) to learn features at different granularities. Smaller kernels ($k = 2$ or 3) capture fine-grained token-level relationships, such as modifiers or word pair dependencies, whereas larger kernels ($k = 5$ or 7) capture slightly broader local patterns, such as syntactic or semantic relationships over small phrases.

Each branch of the inception module applies a 1D convolution over the sequence of contextual embeddings generated by the transformer. Let the input be $H \in \mathbb{R}^{L\times d}$, where L is the sequence length and d is the hidden size. For a convolution with kernel size k, each filter has weights $W \in \mathbb{R}^{k\times d}$ and a bias term $b \in \mathbb{R}$. The output at position i is computed as:

$$Y_i = \sum_{j=0}^{k-1} W_j \cdot H_{i+j} + b$$

where $H_{i+j} \in \mathbb{R}^d$ is the embedding of the $(i + j)$-th token, and $W_j \in \mathbb{R}^d$ is the j-th row of the filter. This operation slides across the sequence to produce a feature map $Y \in \mathbb{R}^{L\times c}$, where c is the number of convolutional filters (i.e. output channels) used in the branch. To preserve the original sequence length, we apply appropriate padding: for kernel size 2, we use right padding of 1; for kernel sizes 3, 5, and 7, we apply symmetric padding.

After the convolution, each branch further processes its output using batch normalization to stabilize and accelerate the training process, followed by a ReLU activation to introduce non-linearity. Finally, the outputs of all four branches are concatenated along the channel dimension to form a combined feature map $C \in \mathbb{R}^{B\times L\times(4\cdot c)}$. To preserve information from the original transformer output, we concatenate H and C along the feature dimension to form $R \in \mathbb{R}^{B\times L\times(d+4c)}$. This residual connection ensures that the original features are retained alongside the multi-scale features. This combined representation, enriched with both global and multi-scale local features, is then passed to the self-attention layer for further processing.

3.2.3 Self-Attention

After the inception module extracts multi-scale features, an additional self-attention mechanism is necessary to capture dependencies and relationships across the enriched feature space R. This ensures that tokens that contribute the most to the task are effectively prioritized and selected, allowing the model to focus on the most relevant features.

Given $R \in \mathbb{R}^{B\times L\times d_R}$, the attention mechanism maps it to query Q, key K, and value V:

$$Q = RW_Q, \quad K = RW_K, \quad V = RW_V$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d_R\times d_A}$, d_R is the enriched feature space dimension, and d_A is the attention head dimension. The attention scores are

computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_A}}\right)V$$

Since we use multi-headed attention, the outputs of multiple attention heads are concatenated and projected back to the original feature space:

$$A = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$$

where $W_O \in \mathbb{R}^{(h \cdot d_A) \times d_R}$ is a learnable projection matrix and h is the number of attention heads, another tunable hyperparameter. The attention output $A \in \mathbb{R}^{B \times L \times d_R}$ captures refined dependencies across both token positions and feature scales.

3.2.4 Adaptive Average Pooling

To reduce the sequence-level representation A to a fixed-size vector suitable for classification, global average pooling is applied across the sequence length. Given the attention output $A \in \mathbb{R}^{B \times L \times d_R}$, we first permute it to $\mathbb{R}^{B \times d_R \times L}$. Afterwards, adaptive average pooling computes the average over the entire sequence for each feature channel, regardless of the input length, by dynamically adjusting the pooling region. Mathematically:

$$P_{b,i} = \frac{1}{L}\sum_{j=1}^{L} a_{b,i,j}$$

where $a_{b,i,j}$ is the value of the ith feature channel at the jth position for the bth example. This produces a tensor $P \in \mathbb{R}^{B \times d_R \times 1}$, which is then squeezed to yield a final pooled representation $P \in \mathbb{R}^{B \times d_R}$.

3.2.5 Dense Block

For further refinement, the pooled representation P is passed through a dense block consisting of three sublayers. First, a fully connected layer is used to reduce the dimensionality by $D = PW_d + b_d$ where $W_d \in \mathbb{R}^{d_R \times d_D}$, $b_d \in \mathbb{R}^{d_D}$, and d_D is the target dimensionality (e.g., 512). ReLU activation is used to introduce non-linearity, and layer normalization is used to stabilize the output. The output of the dense block $D \in \mathbb{R}^{B \times d_D}$ represents a compact and refined feature set ready for classification.

3.2.6 Final Classification

The output D is passed to a linear classifier, which computes logits for each class as $O = DW_f + b_f$; where $W_f \in \mathbb{R}^{d_D \times C}$ and $b_f \in \mathbb{R}^C$. The logits $O \in \mathbb{R}^{B \times C}$ are interpreted based on the task.

Notation. For brevity, we will refer to our inceptive models using the convention $i-\text{Baseline}-n$, where i signifies 'Inceptive', 'Baseline' refers to the Pretrained Model (PLM) (e.g. RoBERTa, BioBERT), and n denotes the number of output channels in each branch of the inception module. For instance, *i*BioBERT-128 is the Inceptive BioBERT model with 128 output channels per convolution.

4 Experimental Setup

4.1 Datasets

For a robust evaluation, we test our framework on five tasks from four datasets spanning diverse domains, text lengths, and both multi-class and multi-label settings. Multi-class tasks include emotion recognition (Mohammad et al., 2018) and irony detection (Van Hee et al., 2018) from the TweetEval benchmark (Barbieri et al., 2020), alongside a large-scale Bengali emotion detection dataset (Faisal et al., 2024) to evaluate performance on a morphologically rich, low-resource language. Multi-label tasks include OHSUMED [1], a collection of biomedical journal abstracts, and CAVES (Poddar et al., 2022), a dataset of anti-COVID vaccine concerns such as side-effects, ingredients, corporate greed, political motivations, etc. More details on the datasets can be found in Appendix B.

4.2 Model Training and Evaluation

Each input sequence is tokenized using a model-specific tokenizer and then passed through the model to generate logits. For multi-class classification, the model predicts mutually exclusive class probabilities using softmax activation and cross-entropy loss, whereas for binary and multi-label tasks, it outputs non-exclusive probabilities with sigmoid activation and binary cross entropy with logits loss. During backpropagation, gradients were clipped to a maximum norm of 1.0 to ensure numerical stability. The AdamW optimizer (Kingma and Ba, 2014) with weight decay was used to update the model weights.

The training process was conducted iteratively over multiple epochs, with a Cosine Annealing learning rate scheduler. At the end of each epoch, the model was evaluated on the validation dataset to monitor key metrics, including accuracy, F1-score, AUC-ROC (multi-class), AUPR (multi-label), and inference time. The best model was selected based

[1] OHSUMED-link

on accuracy for binary and multi-class classification tasks, and F1-score for multi-label tasks. Each model was run 10 times on each dataset. The models were trained and evaluated using 40GB A100 GPU. However, all of our models can be run on 16 GB GPUs (e.g. P100). We used the `transformer` version 4.48.3.

4.3 Hyperparameters

Table 1: Hyperparameters.

Hyperparameter	Value
Sequence Length	128, 512 (ohsumed)
Batch Size	32
Epochs	12
Learning Rate	1e-5
Weight Decay	1e-3, 1e-4 (ohsumed, caves)
Sigmoid threshold	0.5

The hyperparameters used in this experiment are shown in Table 1. All hyperparameter values were selected empirically based on validation set performance.

5 Results

5.1 Comparative Performance

In this section, we compare the performance of the inception-enhanced models against the baselines. Multi-class performance comparison (in terms of accuracy) is shown in Table 2, while multi-label comparison (F1-score) is shown in Table 3. A detailed comparison can be found in appendix D, where we also report precision, recall, AUC-ROC and AUPR, that also account for class imbalance. To account for stochasticity, we performed 10 independent training and evaluation runs for each model on each dataset, and reported the average performance on the test set. Performance comparison across all runs can be found in Appendix E.

In the task of detecting irony in social media posts, the domain-specific BERTweet model saw a **2.20%** improvement through our inceptive framework, while the inference time increased 1.89% – highlighting a clear positive trade-off. Similarly, general-purpose encoder models like RoBERTa, DeBERTa and ModernBERT also improved by margins of **2.57%**, **6.15%**, and **3.90%** respectively – all of which were greater than the inference overhead incurred. Specifically, Inceptive DeBERTa performed at a level similar to that of BERTweet, despite the latter being domain-pretrained.

Table 2: Multi-class performance comparison in test set

Model	Accuracy	Inference Time (s)
Irony Detection		
BERTweet	82.69	1.59
iBERTweet-16	**84.51**	1.62
RoBERTa	75.15	1.60
iRoBERTa-32	**77.08**	1.68
ModernBERT	67.77	1.85
iMB-32	**70.41**	1.93
DeBERTa v3	77.27	1.89
iDB v3-16	**82.02**	1.96
Emotion Recognition		
BERTweet	83.29	2.83
iBERTweet-64	**84.11**	2.93
RoBERTa	81.69	2.88
iRoBERTa-16	**82.42**	3.00
ModernBERT	76.10	3.41
iMB-16	**78.70**	3.47
DeBERTa v3	83.93	3.40
iDB v3-16	**84.16**	3.55
Bangla Emotion Recognition		
BanglaBERT	69.98	15.65
iBanglaBERT-16	**70.74**	16.62
XLM-RoBERTa	65.91	15.42
iXLMR-16	**66.53**	15.77

In comparison, the improvements were modest in emotion recognition, where both Inceptive BERTweet-64 and Inceptive RoBERTa-16 achieved approximately **1%** improvement over baselines. A more substantial gain was observed for ModernBERT, where the inceptive variant improved accuracy by **3.42%**. In Bangla emotion recognition, Inceptive BanglaBERT-16 improved on its baseline by **1.08%**, while Inceptive XLM-RoBERTa-16 achieved a **0.94%** increase in accuracy over vanilla XLM-RoBERTa .

On the long-form, multi-label OHSUMED dataset, our framework delivered its most significant improvements. Inceptive BioBERT achieved an average F1-score of 72.34, which is a **13.92%** improvement over BioBERT that came at a cost of 9.02% increase in runtime. Other models also benefited substantially: Inceptive ModernBERT showed a **12.98%** F1-score increase, while Inceptive DeBERTa improved by **9.93%**. Notably, Inceptive RoBERTa (65.44) performed better than BioBERT (63.50), which is specifically pre-trained on biomedical corpora; thus highlighting the gener-

Table 3: Multi-label performance comparison in test set

Model	F1-score	Inference Time (s)
OHSUMED		
BioBERT	63.50	53.88
iBioBERT-128	**72.34**	58.74
BioBERT-Large	73.12	154.00
iBioBERT-Large-128	**74.77**	158.01
RoBERTa	61.53	67.42
iRoBERTa-128	**65.44**	74.44
ModernBERT	56.71	72.95
iMB-128	**64.07**	77.17
DeBERTa v3	55.47	83.27
iDB v3-128	**60.98**	87.57
CAVES		
CT-BERT	74.24	10.27
iCTBERT-16	**74.86**	10.56
BioBERT-Large	71.00	10.75
iBioBERT-Large-16	**72.32**	11.02
RoBERTa	71.11	4.67
iRoBERTa-32	**72.11**	4.78
ModernBERT	63.31	5.01
iMB-32	**66.63**	5.09
DeBERTa v3	69.05	5.05
iDB v3-32	**71.66**	5.12

alization capability of our plug-and-play architecture. Furthermore, Inceptive BioBERT performed at a similar level as BioBERT-large, despite the latter taking almost 3x as much time to run and requiring significantly more compute power. Nonetheless, Inceptive BioBERT also improved by **2.26%**, confirming the framework's applicability to larger models as well.

Finally, our inceptive module demonstrated consistent gains in the noisy and complex CAVES dataset as well. BioBERT-large improved by **1.86%** whereas CT-BERT-large saw a gain of **0.84%**. The performance uplift was more pronounced among general-purpose models: RoBERTa improved by **1.41%**, DeBERTa by **3.78%**, and ModernBERT by **5.24%**.

Overall, our lightweight inceptive framework has improved every PLM we tested while maintaining reasonable efficiency. The performance delta varies depending on the baseline capability. General-purpose models like DeBERTa v3 and ModernBERT achieved the most significant improvements, whereas more capable domain-specific pre-trained models like CT-BERT and BERTweet saw modest but consistent gains.

Cross Validation Results

Table 4: 10-fold cross validation results comparison

Dataset	Baseline		Inceptive	
	Mean	Std Dev	Mean	Std Dev
Emotion	80.80	1.27	81.38	1.19
Irony	77.49	1.20	78.10	1.27
OHSUMED	65.06	1.35	72.57	0.62
CAVES	71.88	0.94	72.86	0.88

We conducted 10-fold cross-validation for both the baseline and inceptive models across all datasets except the large-scale Bangla dataset (resource constraints). For OHSUMED, we used the training set; for the other datasets, we combined the training and validation sets. The mean and standard deviation of the evaluation scores are reported in Table 4. Across all datasets, the inceptive models consistently achieved higher mean accuracy or F1-scores compared to the baselines. Additionally, in all but one case (irony detection), the inceptive models had a lower variance, indicating more stable performance. These results highlight the robustness and generalizability of our proposed architecture.

5.2 Performance vs Complexity Trade-off

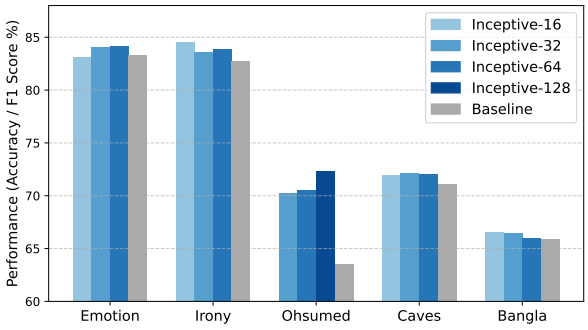


Figure 3: Performance comparison of all tested inceptive configurations and baseline models

A key hyperparameter of our inceptive models is the number of output channels in convolution branches, which we tuned to determine the ideal inception module configuration in each dataset. To account for this added architectural complexity, we have compared the performance of all inception configurations against the baseline models. The results presented in Fig. 3 show that even the lowest performing configuration outperforms the baseline in all but one dataset, and the average performance is always higher. This suggests that extensive tuning is not strictly necessary — any selected config-

uration is likely to yield gain over baseline. This comparison is a post-hoc analysis performed on the test set – these results were not used for the best configuration selection.

A heuristic. For shorter text sequences (emotion recognition, irony, CAVES) 16 or 32 channels perform best, while 128 channels performed best in OHSUMED only – where the text sequences are considerably larger. As a general heuristic, we recommend starting with 16 channels and scaling up based on input length or label complexity. It also aligns with the intuition that longer texts benefit from richer multi-scale features.

5.3 Statistical Significance Testing

Table 5: Wilcoxon Signed-Rank Test Results. BT: BERTweet, BB: BioBERT, RoB: RoBERTa, MB: ModernBERT, DB: DeBERTa v3, ***i***: inceptive model.

Dataset	Models	Gain	p-value
Emotion	BT, *i*BT-64	**+0.98%**	0.00195
	MB, *i*MB-16	**+3.42%**	0.00195
Irony	BT, *i*BT-16	**+2.20%**	0.00585
	MB, *i*MB-16	**+4.33%**	0.00195
	DB, *i*DB-16	**+6.14%**	0.00195
OHSUMED	BB, *i*BB-128	**+13.92%**	0.00195
	MB, *i*MB-128	**+12.98%**	0.00195
	DB, *i*DB-128	**+9.93%**	0.00195
CAVES	RoB, *i*RoB-32	**+1.41%**	0.00195
	MB, *i*MB-32	**+5.24%**	0.00195
	DB, *i*DB-32	**+3.78%**	0.00195
Bangla	XLM, *i*XLM-16	**+0.94%**	0.00195

For statistical significance testing, we performed the Wilcoxon signed-rank test, which is a non-parametric test and suitable for paired comparison on the same test set. Each model was run 10 times for statistical analysis. As shown in Table 5, the p-value in each test is below the 0.05 significance threshold. Therefore, we conclude that the gain achieved are statistically significant.

5.4 Performance Interpretation

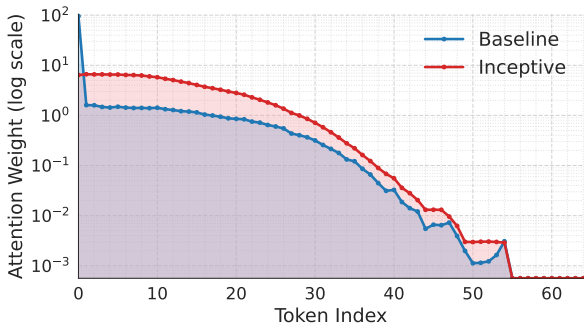
The attention maps for both baseline and inceptive transformers are presented in Fig. 4. Attention weights in the baselines are heavily skewed toward the initial `[CLS]` token, while the rest of the tokens receive comparatively negligible attention. In contrast, the inceptive attention maps highlight a more balanced distribution of attention weights across the sequence. Tokens that were overlooked by transformer-based models, particularly those in the middle of the sequence, now receive more attention, reflecting their contextual importance. This is a direct result of our architectural improvements, which not only enhance contextual representations, but also enable the model to weight tokens dynamically based on their contribution to the task.

Disease identification. The OHSUMED dataset involves long, complex sequences of medical abstracts, where relevant information is dispersed throughout the text. Mentions of symptoms, treatments, or diagnoses appear in different parts of the text, each contributing to the prediction of a specific disease label. As such, it is difficult for the [CLS] token to capture all these into a single token. In contrast, multi-scale convolutional branches of our inception module extract local patterns at varying granularities across the entire sequence, allowing the model to capture dependencies between neighboring tokens that may correspond to biomedical expressions, compound noun phrases, or domain-specific terminology spread throughout the abstract. As a result, our inceptive models achieve a significant **13%** improvement in this dataset.

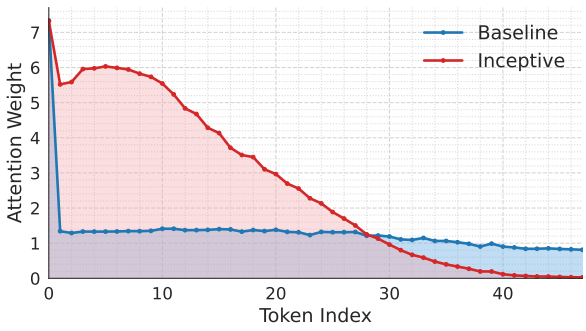
Irony and emotion detection. Irony is usually conveyed through specific linguistic patterns or subtle contextual cues like exaggerated praise or contradiction between words and context. These markers hide in nuanced phrasing rather than being explicitly stated in words – which risks being diluted or averaged out when compressed into a single vector. In comparison, keywords like "happy" or "angry" strongly signify a particular emotion and are more likely to surface in [CLS] even with suboptimal pooling. This is why the [CLS] token proved to be "more capable" in emotion recognition (where the best-performing baseline was 0.98% worse than inceptive model) compared to irony detection (+2.20% gain using inception).

5.5 Ablation Study

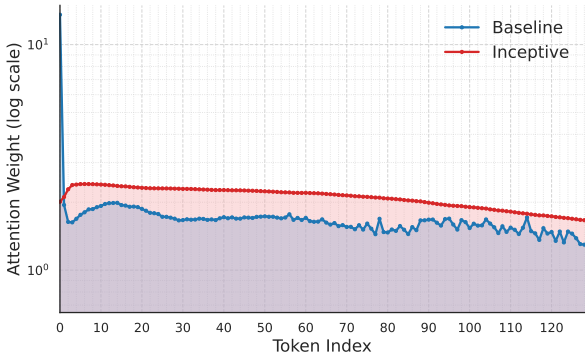
We perform an ablation study to understand the contribution of each main component of our framework. The results in Table 6 show that multi-scale convolution always performs better than a single convolution branch, which justifies our use of the inception module. Moreover, there is no single 'best' kernel size — different kernel sizes capture complementary linguistic patterns, and their combination is what enables the model to adapt effectively across diverse domains and tasks. The ablation study also shows that both self-attention and



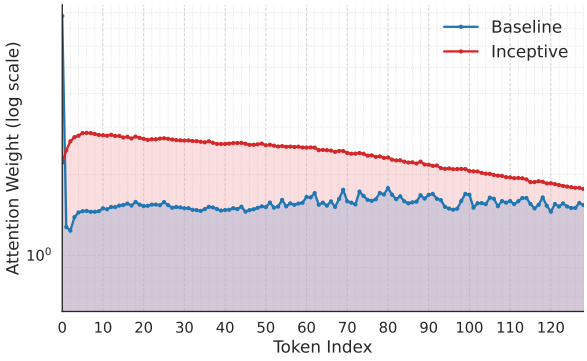
(a) Baseline vs Inceptive BERTweet (Irony detection)



(b) Baseline vs Inceptive DeBERTa v3 (Irony detection)



(c) Baseline vs Inceptive BioBERT (OHSUMED)



(d) Baseline vs Inceptive ModernBERT (OHSUMED)

Figure 4: Attention received by each token in baseline and inceptive models. Baseline attention maps are extracted from the self-attention weights of the transformer's final layer. Inceptive attention maps come from the custom self-attention layer applied after fusion of transformers' hidden states with multi-scale inception features. In both cases they represent token-to-token interaction weights — i.e., how much one token attends to or is influenced by others when forming the final sequence representation. Attention is visibly skewed towards the initial [CLS] token in baseline models, whereas inceptive models balance token importance dynamically based on task relevance.

Table 6: Ablation study results for inception module, each convolution branch, self-attention, and dense block

Model	**Full**	**Single convolution branch**				**No conv**	**No attn**	**No dense**
		$k = 2$	$k = 3$	$k = 5$	$k = 7$			
***i*BT (Emotion)**	84.11	83.82	83.63	83.06	83.79	83.27	83.63	83.51
***i*BT (Irony)**	84.51	83.80	83.42	83.24	83.85	82.61	82.61	82.48
***i*BB (OHSUMED)**	72.34	70.09	69.93	71.11	69.73	67.27	71.54	69.00
***i*RoB (CAVES)**	72.11	71.98	71.68	71.97	71.49	71.50	71.31	71.38

dense block add value, as removing either worsens performance. Finally, if we remove convolution entirely, the PLM + self-attention + dense block combination performs only marginally better than the baseline PLM, proving that multi-scale feature extraction is essential.

6 Conclusion

In this paper we presented *Inceptive Transformer*, a general convolution-based framework that enhances the performance of both general-purpose transformer models like RoBERTa and domain-specific pre-trained language models such as BERTweet, BioBERT, and CT-BERT. Our experiments show statistically significant performance gains ranging from 1% to 14% while inference overhead remained less than 10%. Moreover, our approach consistently delivers strong results across diverse domains and languages while maintaining computational efficiency. In future work, we plan to adapt our model to other tasks (e.g., NER, Q/A) and architectures (e.g., encoder-decoder models) and investigate the impact on tasks involving long-range dependencies such as code classification.

7 Limitations

Dependency on output channels. A limitation of our architecture is that it requires tuning the number of output channels in the inception module to achieve optimal performance in different datasets. For example, while an inception module with 128 output channels works best for BioBERT, 16 (for irony detection) and 32 or 64 (for emotion recognition) output channels are more suitable for BERTweet. However, we empirically found that even the lowest performing inception configuration outperformed the baseline in all but one case.

Focus on encoder models. We applied our inceptive framework exclusively to bidirectional encoder-only transformer models; encoder-decoder models (e.g., T5 or BART) were not explored. Applying the inception module in such generative or sequence-to-sequence settings may require architectural adaptations.

Impact on long-range dependencies not tested. By design, inceptive models are most suitable for tasks that depend on local features as well as global features. We have not explored how incorporating inductive biases would impact classification on domains dominated by long-range dependencies, such as code-classification with CodeBERT style models. This can be an exciting future work.

Acknowledgment

We thank the anonymous reviewers and meta-reviewer for their feedback that has greatly helped in improving the quality of the paper. While writing the paper, we used AI assistance for polishing a few sentences and for some minor debugging of the code. The authors remain fully responsible for both the manuscript and the code. M Saifur Rahman is partially supported by Basic Research Grant from BUET, and CodeCrafters-Investortools Research Grant.

References

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa-Anke, and Leonardo Neves. 2020. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: Pretrained language model for scientific text. In *EMNLP*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Ahmad, Kazi Samin Mubasshir, Md Saiful Islam, Anindya Iqbal, M. Sohel Rahman, and Rifat Shahriyar. 2022. BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327, Seattle, United States. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Haw-Shiuan Chang, Ruei-Yao Sun, Kathryn Ricci, and Andrew McCallum. 2023. Multi-CLS BERT: An efficient alternative to traditional ensembling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–854, Toronto, Canada. Association for Computational Linguistics.

Qingyu Chen, Yan Hu, Xueqing Peng, et al. 2025. Benchmarking large language models for biomedical natural language processing applications and recommendations. *Nature Communications*, 16(1):3280.

Xinying Chen, Peimin Cong, and Shuo Lv. 2022. A long-text classification method of chinese news based on bert and cnn. *IEEE Access*, 10:34046–34057.

Byung-Ju Choi, Jun-Hyung Park, and SangKeun Lee. 2019. Adaptive convolution for text classification. 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)*, pages 2475–2485.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, pages 4171–4186.

Aleksandra Edwards and Jose Camacho-Collados. 2024. Language models for text classification: Is in-context learning enough? In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Torino, Italia. ELRA and ICCL.

Moshiur Rahman Faisal, Ashrin Mobashira Shifa, Md Hasibur Rahman, Mohammed Arif Uddin, and Rashedur M. Rahman. 2024. Bengali & banglish: A monolingual dataset for emotion detection in linguistically diverse contexts. *Data in Brief*, 55:110760.

Jinsheng Fang, Hanjiang Lin, Xinyu Chen, and Kun Zeng. 2022. A hybrid network of cnn and transformer for lightweight image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1103–1112.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. 2018. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377.

Maosheng Guo, Yu Zhang, and Ting Liu. 2019. Gaussian transformer: A lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6489–6496.

Zahra Hanifelou, Peyman Adibi, Sayyed Amirhassan Monadjemi, and Hossein Karshenas. 2018. Knn-based multi-label twin support vector machine with priority of labels. *Neurocomputing*, 322:177–186.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29.

Anwesha Law and Ashish Ghosh. 2022. Multi-label classification using binary tree of classifiers. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(3):677–689.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Pengfei Li, Peixiang Zhong, Kezhi Mao, Dongzhe Wang, Xuefeng Yang, Yunfeng Liu, Jianxiong Yin, and Simon See. 2021. Act: an attentive convolutional transformer for efficient text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13261–13269.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. 2022. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4221–4231. Association for Computational Linguistics.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14. Association for Computational Linguistics.

Aytuğ Onan. 2022. Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification. *J. King Saud Univ. Comput. Inf. Sci.*, 34:2098–2117.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Soham Poddar, Azlaan Mustafa Samad, Rajdeep Mukherjee, Niloy Ganguly, and Saptarshi Ghosh. 2022. Caves: A dataset to facilitate explainable classification and summarization of concerns towards covid vaccines. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Chenyang Si, Weihao Yu, Pan Zhou, Yichen Zhou, Xinchao Wang, and Shuicheng Yan. 2022. Inception transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22.

Sanskar Soni, Satyendra Singh Chouhan, and Santosh Singh Rathore. 2022. Textconvonet: a convolutional neural network based architecture for text classification. *Applied Intelligence (Dordrecht, Netherlands)*, 53:14249 – 14268.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore. Association for Computational Linguistics.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

C. X. Wan and B. Li. 2022. Financial causal sentence recognition based on bert-cnn text classification. *Journal of Supercomputing*, 78:6503–6527.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. *Preprint*, arXiv:1908.04577.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

D. Wu, Z. Wang, and W. Zhao. 2024. Xlnet-cnn-gru dual-channel aspect-level review text sentiment classification method. *Multimed Tools Appl*, 83:5871–5892.

Qingyao Wu, Mingkui Tan, Hengjie Song, Jian Chen, and Michael K. Ng. 2016. Ml-forest: A multi-label tree ensemble method for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2665–2680.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5754–5764.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:7370–7377.

Feiniu Yuan, Zhengxiao Zhang, and Zhijun Fang. 2023. An effective cnn and transformer complementary network for medical image segmentation. *Pattern Recognition*, 136.

Shaomin Zheng and Meng Yang. 2019. A new method of improving bert for text classification. In *Intelligence Science and Big Data Engineering. Big Data and Machine Learning*, pages 442–452. Springer International Publishing.

Xiaoyan Zhu, Jiaxuan Li, Jingtao Ren, Jiayin Wang, and Guangtao Wang. 2023. Dynamic ensemble learning for multi-label classification. *Information Sciences*, 623:94–111.

A Baseline Architecture

The architecture for the baseline models follows the standard fine-tuning approach for transformer-based classification tasks: the pre-trained transformer layer is followed by a dropout layer (p = 0.3) and a linear classification head. The hidden state of the `[CLS]` token from the final transformer layer is used as the sequence representation, which is passed through dropout to prevent overfitting and then mapped to the output classes via a fully connected layer.

B Dataset Class Distributions

Table 7: Dataset statistics. C : number of classes or labels; $\overline{C}$: average number of labels per instance (for multi-label); and $\overline{L}$: average token length of each text.

Dataset	#Texts	C	$\overline{C}$	$\overline{L}$
Emotion	5,052	4	–	24.35
Irony	4,601	2	–	21.54
Bangla	80,098	6	–	18.6
OHSUMED	13,929	23	1.66	289.51
CAVES	9,921	11	1.16	58.35

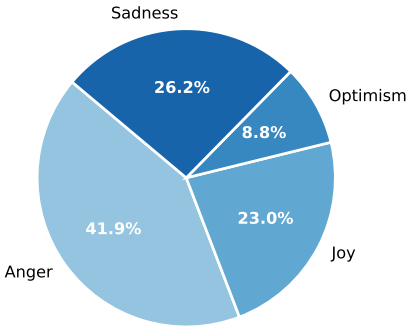


Figure 5: Class distribution in Emotion Recognition

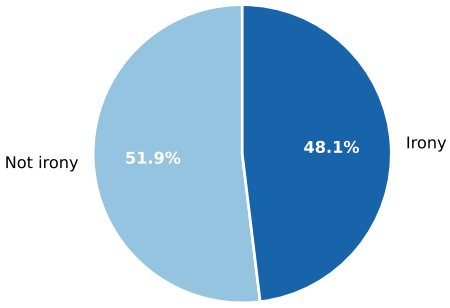


Figure 6: Class distribution in Irony Detection

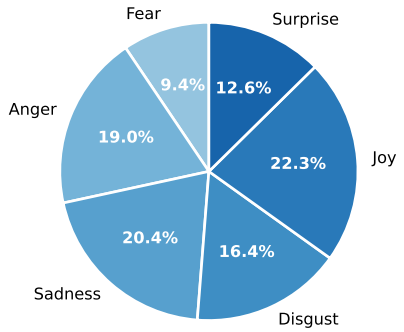


Figure 7: Class distribution in Bangla emotion detection

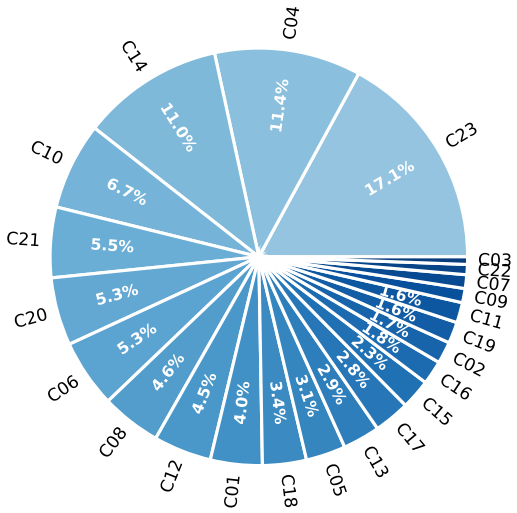


Figure 8: Class distribution in OHSUMED

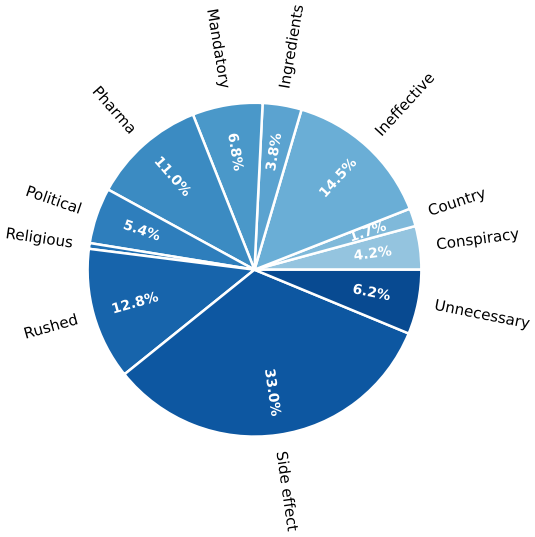


Figure 9: Class distribution in CAVES

C Word clouds

Figure 10: Word clouds of OHSUMED dataset

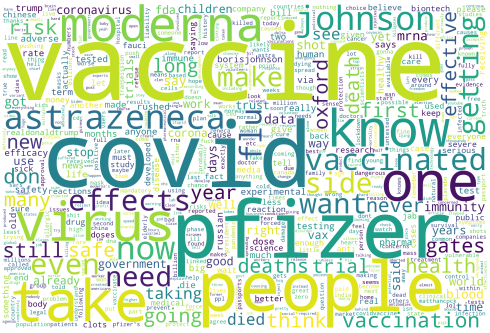


Figure 11: Word clouds of CAVES dataset

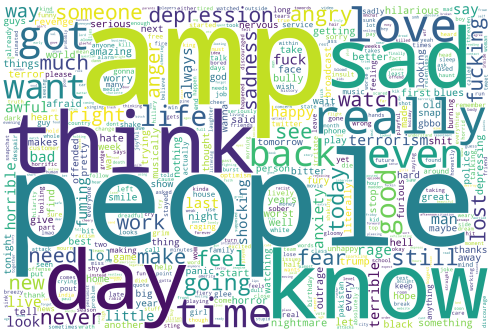


Figure 12: Word clouds of emotion recognition dataset

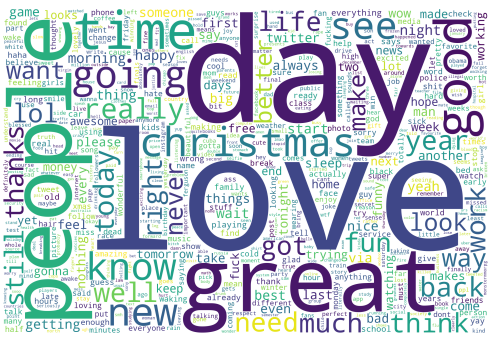


Figure 13: Word clouds of irony detection dataset

D Full Performance Comparison

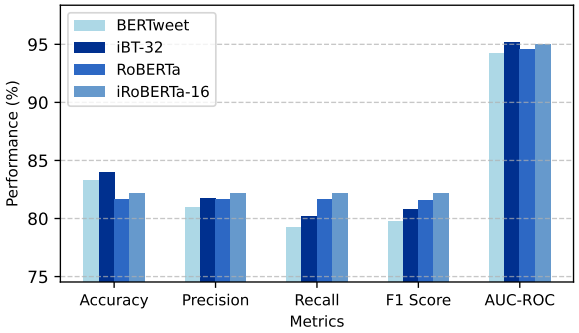


Figure 14: Performance comparison in Emotion Recognition

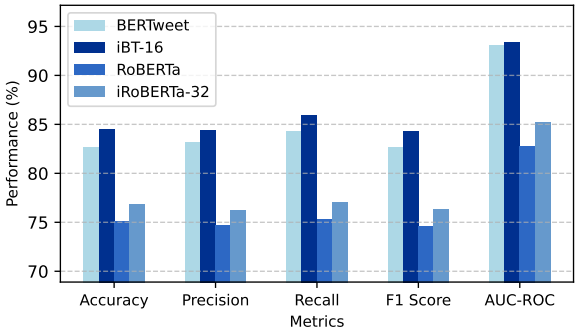


Figure 15: Performance comparison in Irony Detection

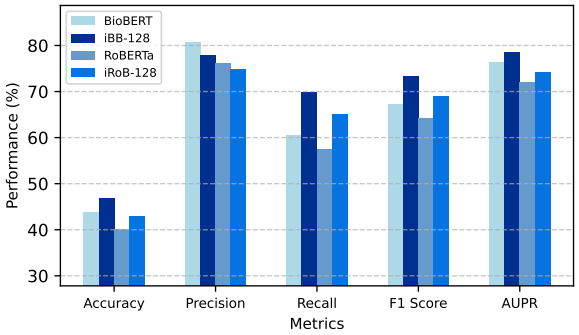


Figure 16: Performance comparison in OHSUMED

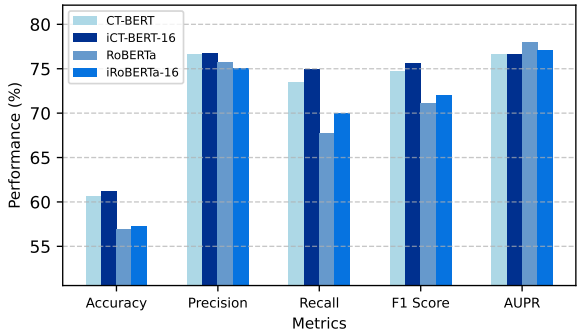


Figure 17: Performance comparison in CAVES

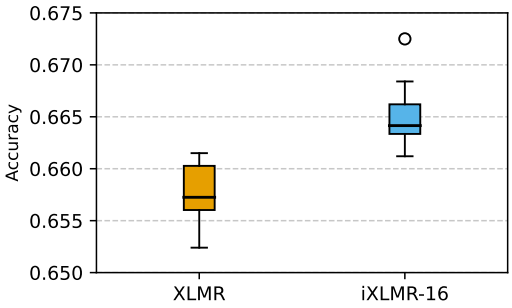


Figure 20: Accuracy distribution across 10 runs in Bangla emotion detection

E Comparison across All Runs

Fig. 18, 19, 20, 21, and 22 show the comparison of baseline pretrained models (BERTweet, XLMR, BioBERT, RoBERTa) against the inception models across all 10 runs.

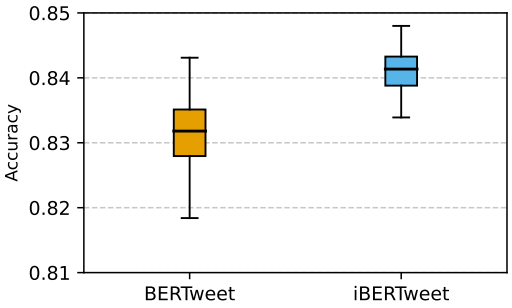


Figure 18: Accuracy distribution across 10 runs in Emotion Recognition

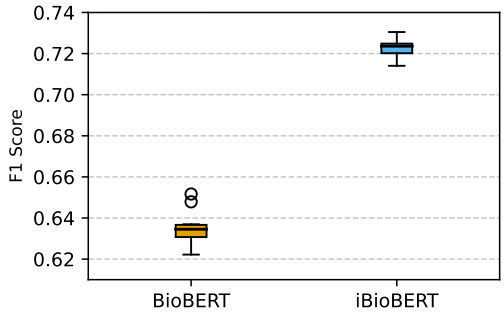


Figure 21: F1-score distribution across 10 runs in OHSUMED

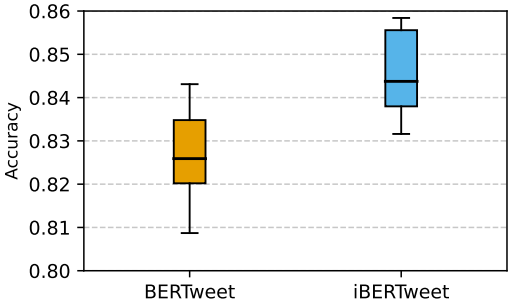


Figure 19: Accuracy distribution across 10 runs in Irony Detection

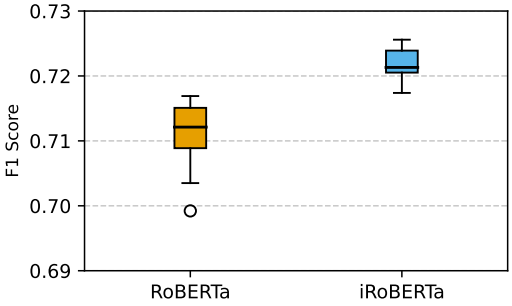


Figure 22: F1-score distribution across 10 runs in CAVES

F Attention Maps

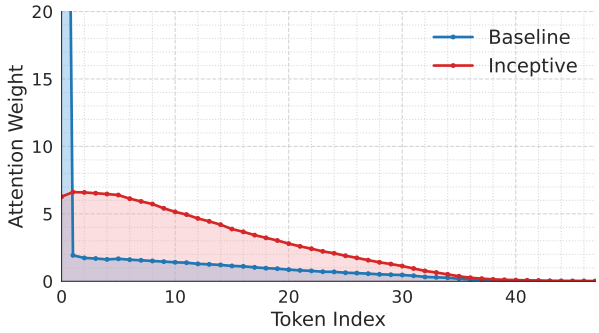


Figure 23: Baseline vs Inceptive BERTweet attention map (emotion)

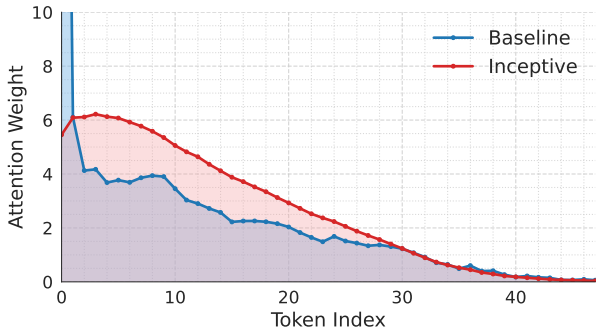


Figure 24: Baseline vs Inceptive RoBERTa attention map (emotion)

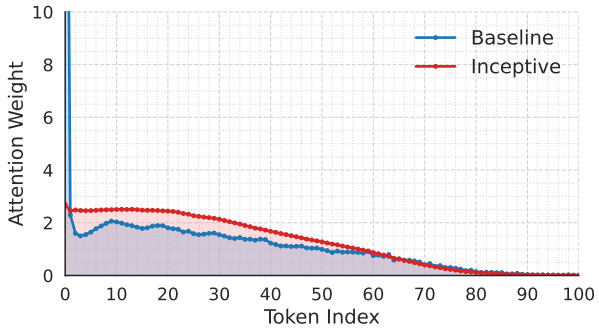


Figure 25: Baseline vs Inceptive RoBERTa attention map (CAVES)

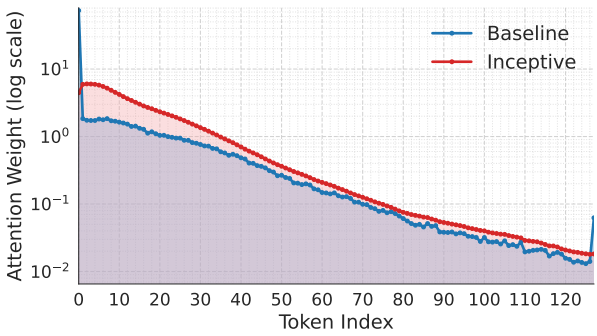


Figure 26: Baseline vs Inceptive XLM-R attention map (Bangla)

G Stacking Multiple Inception Modules

We implemented a simple variant of our architecture that stacks two inception modules sequentially – the output of the first inception module is fed into the second one for further multi-scale refinement. We tested it on the two datasets where we had most improvements – irony and OHSUMED – and found that it performs equally or slightly worse than our original architecture (but still better than the baselines). This is likely due to the additional module introducing redundant or over-smoothed representations, which may obscure the fine-grained features captured by the first inception layer.

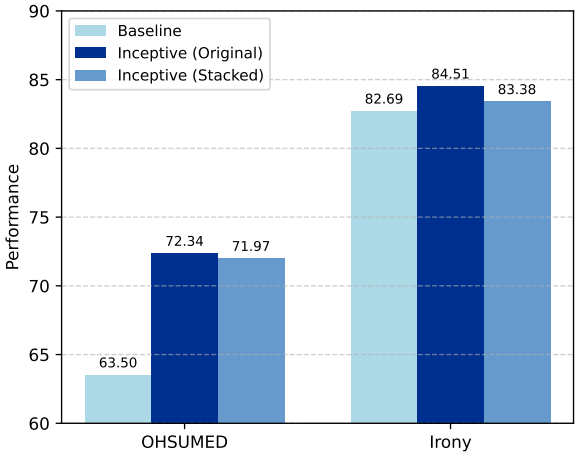


Figure 27: Single inception vs 2-stacked inception