Longitudinal Affective Forecasting: Architectures for Generalization, State Change, and Trajectory Prediction (SemEval-2026 Task 2)

Haseebullah Jumakhan
Ajman University
202320029@ajmanuni.ac.ae

Soud Assad
Ajman University
202420468@ajmanuni.ac.ae

Seyed Abdullah Seyed Ahmad
Ajman University
202412068@ajmanuni.ac.ae

Abstract

Generalizing, maintaining consistency over time, and optimizing for resources are the most important difficulties with identifying emotional patterns through longitudinal text. This document is about how we addressed each of these issues by focusing on Subtask 1, which is finding variations in affect; Subtask 2A, which is forecasting state changes; and Subtask 2B, which is forecasting long-term trends in traits using a series of deep learning models, starting with a hybrid model of DistilBERT and LSTM and ending with a Bifurcated Siamese Model (Sanh et al., 2019; Hochreiter and Schmidhuber, 1997). Our approach also incorporates an emphasis on careful splitting at the level of the individual to prevent data leakage and use of specialized loss functions (CCC and Huber) to address the problem of "regression to the mean" (Lin, 1989; Huber, 1964). Finally, we show that although various stability methods (such as schedulers and gradient clipping) may help achieve good performance on the test set for individuals whose data was used in training, those same methods often hinder achieving good performance when testing new groups of people.

Keywords: Valence-Arousal-Dominance (VAD), Longitudinal Modeling, Temporal Affect, Emotional Triggers, Algorithmic Bias, SemEval.

1 Introduction

Human emotion has been represented in many ways, but the most common way to represent emotion is with the ***Circumplex Model of Affect*** that uses a 2-dimensional space to describe emotions based on ***Valence*** (***Positive/Negative***) and ***Arousal*** (***High Energy/Low Energy***) shown in Figure 1.

The ***SemEval-2026 Task 2*** task involves predicting the ***Emotional Dynamics*** that are present in the longitudinal text data of a ***User***, where ***Naturalistic Longitudinal Text Data*** represents a series of weeks/months of a ***User's*** affective experience, rather than the static representation of a single point of ***Sentiment Analysis*** (sem, 2026), which is a static representation of a user's ***Sentiment***.

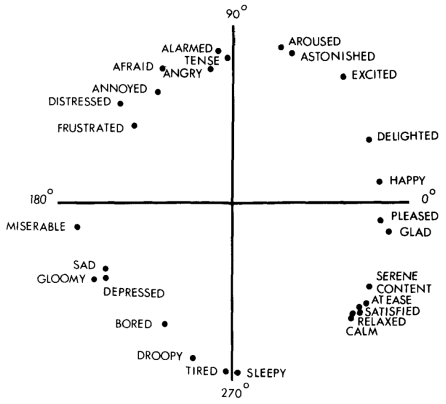

Figure 1: Russell's original empirical circumplex of affect with emotional terms plotted by their direct circular-scaling polar coordinates, illustrating Valence (horizontal axis; pleasant → unpleasant) and Arousal (vertical axis; low → high) (Russell, 1980).

The main problem we are trying to solve in this work is the trade-off between ***specialization*** (i.e., accurately modeling the known ***users***) and ***generalization*** (i.e., predicting for ***unseen users***) under the constraint of limited hardware resources (i.e., **VRAM = 8 GB**).

Organization of Paper To accurately represent the distinct methodological requirements of the competition, this report will be divided into three separate sections, each representing one of the tasks in the competition:

- **Section 2 (Subtask 1)** describes how we used a **Sequence-to-Sequence Modeling Approach** to model both the "Generalist" and "Specialist" architectures for the two different types of models.
- **Section 3 (Subtask 2A)** discusses how we

modeled the short-term transitions between consecutive emotional states using forecasting methods and the change from using MSE to using CCC loss.

- **Section 4 (Subtask 2B)** describes the use of our "Bifurcated Leviathan" architecture to model the longer term affective dispositions of an individual.

Each of the above three sections will have their own unique data processing pipeline, model architecture descriptions, and error analyses/ablation studies.

2 Subtask 1: Evaluating the Variation in Valence and Arousal

The purpose of Task 1 is to make predictions about the continuous scores of Valence and Arousal for a series of user generated texts. The primary difficulty in completing Task 1 arises when modeling variable-length user histories in a manner that prevents the model from becoming overly specialized to the writing style(s) used by the training population.

2.1 Data Preprocessing and Partitioning

The training dataset contains user id information, timestamp data, and text entry information (essays or feeling words). Initial data handling structures were informed by the open-source implementation by (ThickHedgehog, 2025). However, two significant issues have been found in standard methods for baseline data splitting and inference context that can be applied to this task.

2.1.1 The "Mock Competition" Split Strategy

Random splitting can cause data leakage in longitudinal type tasks, because the model could view the same user in both training and testing (Tsakalidis et al., 2022). To provide a way to evaluate a model using the "Unseen User" condition as required by the competition, we developed a strictly based on groups method for splitting the data into a train/test set:

1. **Unseen Users (Test Set B):** We removed 20% of all users from the training process. These users will act as a proxy for evaluating the model on the official leaderboard.

2. **Seen Users (Train/Test A):** We took the remaining 80% of the users and split them temporally. The model uses the past entries to train on the user's past entries and evaluates on the user's future entries.

2.1.2 Context Starving

One problem discovered in the initial audit was an *Inference Mismatch*. The baseline model was trained on sliding windows of five texts but was evaluated on individual texts during inference. This caused the model to forget its temporal memory. To correct this issue we created a custom test dataset class that creates sliding windows with padding during inference so the model always has access to a history window of ten texts.

2.2 Architecture of the Model

Our proposed hybrid model combines a RNN component with a transformer component to use user-specific baselines and linguistic features to predict the Valence and Arousal values.

2.2.1 Feature Extraction and Fusion

Let x represent the input text sequence and let u represent the user ID. We use a pre-trained `distilbert-base-uncased` model to generate the semantic embeddings:

$$h_{\text{text}} = \text{BERT}(x) \in \mathbb{R}^{768} \tag{1}$$

At the same time we find a low dimensional representation for each user which represents the user's affective baseline:

$$h_{\text{user}} = \text{Embed}(u) \in \mathbb{R}^{32} \tag{2}$$

The combined representation for the temporal module is formed by concatenating these representations:

$$h_{\text{combined}} = [h_{\text{text}}; h_{\text{user}}] \in \mathbb{R}^{800} \tag{3}$$

Thus, we ensure that the model is able to consider who is talking alongside what they are saying in this 800-dimensional vector.

2.2.2 Temporal Modeling and Regression

The output of the combined vector sequence is then input to a Bidirectional LSTM to identify temporal relationships among the user's emotions (Schuster and Paliwal, 1997).

$$h_t = \text{BiLSTM}(h_{\text{combined}}) \in \mathbb{R}^{256} \tag{4}$$

Finally, the output of the BiLSTM is input to two separate Multi-Layer Perceptron (MLP) heads to regress the target variables, Valence $\hat{V}$, and Arousal $\hat{A}$.

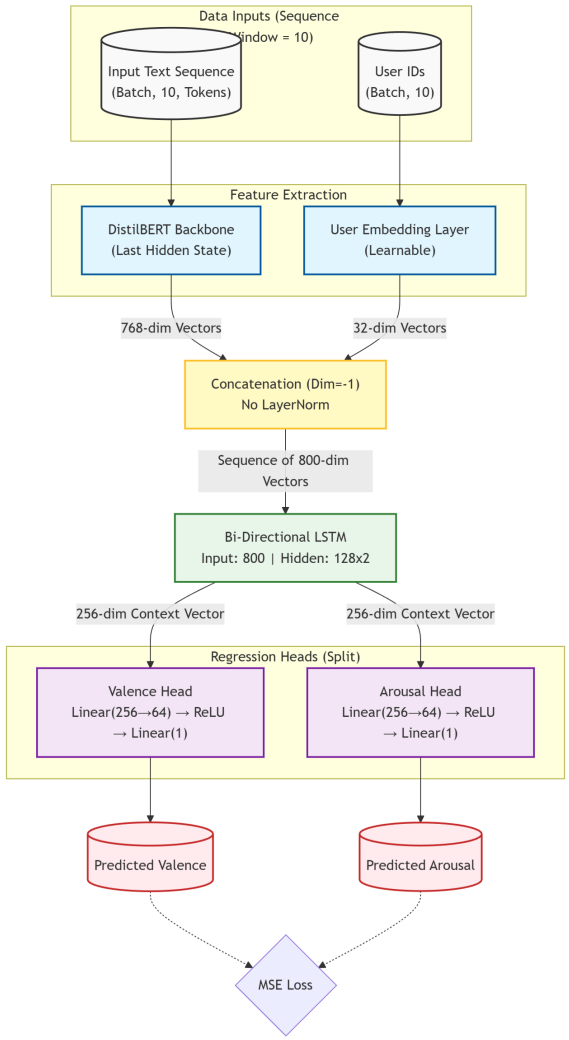

Figure 2: Proposed Hybrid Framework for Subtask 1 combining DistilBERT and User Embeddings.

2.3 Experimental Design and Ablation Study

To assess how both the quality of the engineered features and the quality of the optimization process affected the results, we ran three different experiments. All models used Gradient Accumulation (with effective batch size equal to 16) due to memory constraint imposed by our 8 GB VRAM machine.

2.3.1 Experiment 1: The Robust Baseline ("The Generalist")

In this experiment, we ran the architecture described above with a frozen DistilBERT backbone. As such, this baseline had no additional features to learn and, thus, we did not implement any aggressive training schedulers to avoid over-fitting.

- **Result:** The "generalist" produced the highest Valence correlation ($r = 0.7026$) for Seen users and the highest Arousal correlation ($r = 0.4241$) for Unseen users.

- **Analysis:** By restricting the optimization process to natural variability (i.e., without aggressive schedulers) we ensured the "generalist" maintained robust generalization properties.

2.3.2 Experiment 2: Explicit Stylistic Features ("The Advanced Model")

We hypothesize that since Arousal is a measure of intensity, it may benefit from explicit stylistic signals. To test this hypothesis, we added five "ghost" features: Time of Day, Weekend Flag, Log Word Count, Exclamation Ratio, and Caps Ratio. In addition, we included a temporal consistency loss to ensure the model produces smooth transitions between states.

- **Result:** The "advanced" model suffered greatly; V_{seen} dropped to 0.6638.

- **Analysis:** The inclusion of the temporal consistency loss resulted in overly smoothed outputs. Since Pearson correlation relies on variance, forcing the model to produce "smooth" outputs destroyed the model's ability to capture fluctuating emotions. Moreover, the hand-crafted features introduced noise instead of providing useful signals.

2.3.3 Experiment 3: The Refined Model ("The Specialist")

In this experiment, we removed the hand-crafted features, but we kept other stabilization techniques including a linear warm-up scheduler and gradient clipping (with a norm of 1.0).

- **Result:** This model achieved the highest Seen Arousal score ($r = 0.5487$) but had poor Unseen scores ($r = 0.3843$).

- **Analysis:** The model's stable optimization enabled the model to "remember" the energy patterns of known users (specialization), however, this specialization came at the expense of the model's ability to generalize to new users.

Metric	Run 1 (Baseline)	Run 2 (Features)	Run 3 (Refined)	Verdict
Seen Valence	**0.7026**	0.6638	0.6809	Run 1
Seen Arousal	0.5186	0.5059	**0.5487**	Run 3
Unseen Val.	0.6386	**0.6401**	0.6288	Tie
Unseen Aro.	**0.4241**	0.4098	0.3843	Run 1

Table 1: Comparison of performance metrics of Subtask 1 experiments. Run 1 has the best generalization (Unseen) and Run 3 has the best Seen Arousal.

2.4 Error Analysis of Subtask 1

Two major causes of errors can be seen in our analysis:

1. **The Arousal Gap:** In all experiments, Valence was predicted much better ($r \approx 0.70$) than Arousal ($r \approx 0.50$). This difference in difficulty of prediction stems from the fact that Valence is explicitly semantically-defined (e.g., "happy", "sad") whereas Arousal is typically defined implicitly through punctuation and sentence-structure, both of which are difficult to represent accurately in standard BERT models.

2. **Generalization vs. Optimization:** Experiment 3 demonstrates that optimization techniques (like schedulers) serve as a double-edged sword: They allow the model to converge deeper on training data (and improve Seen scores), but they destroy the noise needed for robust generalization (and decrease Unseen scores).

Therefore, based on this analysis, **Run 1** was chosen as the primary submission because it represents the best balance between accuracy and robustness.

3 Subtask 2A: Forecasting Emotional State Changes

In Subtask 2A, the focus moves from predicting fixed scores, to predicting the *change* in a user's emotional state. In formal terms, the model will use an input text entry x_t, and a user's present affective state (V_t, A_t), to forecast the delta vector:

$$[\Delta\hat{V}, \Delta\hat{A}] = [V_{t+1} - V_t,\ A_{t+1} - A_t] \quad (5)$$

To do so, the model must interpret the essay's semantic content as a "force" acting upon the user's base level to cause the change.

3.1 Preprocessing and Feature Engineering

Temporal Ordering and Splitting The unsorted raw dataset cannot be used for a predictive task. All rows of the dataset were therefore sorted according to `user_id` and `timestamp`. Rows corresponding to terminal entries (i.e., the last posts from users) had to be removed because there was no future state available to serve as their target. To avoid leaking data into the model, we employed a **GroupShuffleSplit** to remove 10% of users randomly, and thus, create a completely new test set.

Normalization of Current State The current state (V_t, A_t) is an important input feature to the model. Therefore, we normalized these two scalar values using Z-score normalization ($\mu = 0, \sigma = 1$), based solely on the statistical values calculated over the training set.

3.2 Model Architecture

We have proposed a Hybrid Fusion model architecture that can combine high-dimensional textual representations with low-dimensional numerical representations of states.

3.2.1 The Drowning Problem

At first, we encountered the issue where simply concatenating the 768-dimensional representation of the text with the 2-dimensional representation of the state would result in the numeric signal being "overwhelmed" during backpropagation, such that the network viewed the numeric signals as noise. Thus, to alleviate this problem, we created a **Feature Projection** module.

3.2.2 Components of the Model

1. **Encoder of Text:** A `microsoft/deberta-v3-base` encoder models the essay x_t into a dense vector $h_{text} \in \mathbb{R}^{768}$ (He et al., 2021).

2. **Projection of State (The Megaphone):** The normalized current state (V_t, A_t) is fed through a simple MLP (Linear $\rightarrow$ GELU $\rightarrow$ Dropout) to project it onto a higher dimensional space:

$$h_{num} = \text{MLP}_{proj}([V_t, A_t]) \in \mathbb{R}^{64} \quad (6)$$

This helps to amplify the influence of the current state during the training process.

3. **Regression Head:** The textual and numeric representations are concatenated and then normalized via LayerNorm to help regulate the variance of the fused representation before passing it to the regression head:

$$h_{fused} = \text{LayerNorm}([h_{text}; h_{num}]) \quad (7)$$

3.3 Experimental Design and Ablation Study

Our final model was the product of three distinct iterations of failure and correction due to hardware limitations (8GB VRAM). Each iteration utilized Mixed Precision (FP16) and Gradient Accumulation (effective batch size = 32).

3.3.1 Phase 1: Stabilization (Huber Loss)

Our original model-based approach employed a Weighted MSE loss, intended to penalize large mood swings. However, our model suffered from **Gradient Explosions** while attempting to train. Our training loss fluctuated wildly between 1.7 and 4.5. Instead of learning from outliers, the model was combating them.

- **Solutions:** We changed to **Huber Loss** (SmoothL1) to prevent large error gradients from exploding. We also limited the magnitude of gradients to 1.0.
- **Results:** Our training became stable, however our model learned to predict near-zero changes for all input combinations ($r \approx 0.30$).

3.3.2 Phase 2: Feature Engineering (Projection)

Although training was now stable, the model still had difficulty utilizing the initial state (V_t, A_t).

- **Solutions:** We added the projection MLP, discussed in Section 3.2, and removed dropout from the final output layer to prevent limiting the range of predictions.
- **Results:** The correlation increased to $r \approx 0.39$. Although the model started to take account of the context, the predicted changes were still conservative.

We were unable to achieve an improvement on our performance as we reached a plateau. The reason for this plateau was that all of the loss functions we have utilized (Euclidean, i.e., MSE/Huber) tend to cause regression towards the mean. In other words, when trying to optimize MSE, the safest thing to do is to output the average, or in this case, no change (i.e., zero). Unfortunately, emotion forecasting has large variances, therefore, regression towards the mean results in a prediction of the average, which does not reflect the fluctuations in emotional states.

1. **Solution:** We replaced MSE with the **Concordance Correlation Coefficient (CCC) Loss** to optimize the covariance, rather than just the mean squared error. With this, the model is forced to track both the means and variances of the ground truth, thus making it less likely to result in flatline predictions. Additionally, we changed the backbone to use `deberta-v3-base`, and implemented differential learning rates ($1e^{-5}$ for backbone and $1e^{-3}$ for head) to better optimize the training process.
2. **Result:** This resulted in a significant increase in performance and a final correlation of $r = 0.64$.

Metric	**Baseline** (Mean)	**Phase 1** (Huber)	**Phase 2** (Proj.)	**Phase 3** (CCC)
Test MSE	2.02	1.83	1.75	**1.47**
Avg Corr. (r)	0.00	0.30	0.39	**0.64**
Improvement	-	9.4%	13.3%	**27.2%**

Table 2: Performance improvements throughout the three developmental phases. The largest increase occurred in phase 3 when using CCC Loss..

3.4 Error Analysis

A Critical Misconception About Loss Functions and Their Effects in Affective Computing In our transition from phase one to phase three, we demonstrate a fundamental lesson for researchers working in the field of affective computing: The minimization of the euclidean distance between two values is not the same as the maximization of the predictive value of those two values. As demonstrated in our experiments, models optimized for mse had very low error, but produced completely unhelpful outputs (i.e., they consistently produced flat lines). Therefore, by transitioning to ccc loss, we traded lower volatility in exchange for being able to produce useful outputs that captured the shape of the user's emotional trajectory over time.

Amplifying Weak Signals via Modal Imbalance Finally, the significant performance increase we observed in phase 2 demonstrates another key challenge that deep learning architectures face when dealing with **modal imbalance**. When a large number of features in a modal are represented as scalar values and are concatenated with a large vector representation of text data, the model will ignore the scalar values due to its lack of capacity to represent them. To avoid ignoring these weak signals, we needed to find a way to amplify the signal of the scalar representations so that the model would be able to learn how the essay represented in the vector space interacts with the user's baseline state. Therefore, we explicitly projected the scalar representations into a higher dimensional space before passing the combined feature space through the network.

4 Subtask 2B: Longitudinal Trajectory Modeling

While Subtask 2B can be seen as a form of standard sentiment analysis, it is actually a *Many-to-One Longitudinal Regression* problem. In contrast to Subtask 2A, which involves mapping a single text to an associated state change, the goal of Subtask 2B is to use a user's complete history of essays over the course of days or months to determine how far the user will shift relative to the same baseline over time.

More formally, given a sequence of N essays $X = \{x_1, x_2, ..., x_N\}$ the model should predict the $\Delta = Y_{tail} - Y_{head}$.

4.1 Preprocessing and the "Leviathan" Protocol

The major engineering obstacle in processing dozens of lengthy texts per user within memory constraints led us to implement a custom-preprocessing pipeline specifically designed for the requirements of long-context discriminative modeling.

Segmentation of Head and Tail We ordered the historical essays for each user chronologically and took two different subsets: the **Head** (the first 16 essays that represent the user's initial state) and the **Tail** (the last 16 essays, that represent the current state). Using this "Head-Tail" sampling method enables the model to identify the difference between the user's current position and their original position, thus providing a similar structure to that of a Siamese network (Bromley et al., 1993).

Scaling and Naive Injection During our early trials we observed "Scale Collapse", where all of our predictions were clustered very close to zero. To combat this we normalized the regression target values using Z-score normalization ($\mu = 0, \sigma = 1$), and then we also calculated the "Naive Statistical Change" (arithmetic mean of the differences between the available self-reporting scores):

$$\Delta_{naive} = \mu(Y_{tail}) - \mu(Y_{head}) \quad (8)$$

As part of data ingestion, we add this scalar value to the final layer, changing the task from a pure regression task to *Residual Learning*. As such, the deep neural net learns to improve the statistical trend, rather than derive it from scratch (He et al., 2016).

4.2 Model Architecture: Bifurcated Siamese Network

The final architecture used in our solution, referred to as the "Bifurcated Leviathan", was developed to address the issue of Task Dominance, whereby optimization for one variable (Arousal) suppresses the learning of another (Valence).

4.2.1 Difference Pooling via Siamese

To make use of a shared `deberta-v3-large` backbone to process the Head and Tail segments separately, we instead of simply concatenating the outputs of the separate encoding processes, we employed Siamese Difference Pooling to explicitly model the vector trajectory:

$$h_\Delta = \text{Pooling}(h_{tail}) - \text{Pooling}(h_{head}) \quad (9)$$

The regression head accepts the concatenation of $[h_{head}; h_{tail}; h_\Delta]$, ensuring it has access to the baseline, current state, and direction of change.

4.2.2 Independent Optimization Heads

In order to avoid gradient interference between Valence and Arousal, we split the network immediately after the backbone. Valence and Arousal are processed by independent "projector" networks and regression heads. This bifurcation prevents the noisy Arousal loss landscape from disrupting the convergence of the Valence loss function.

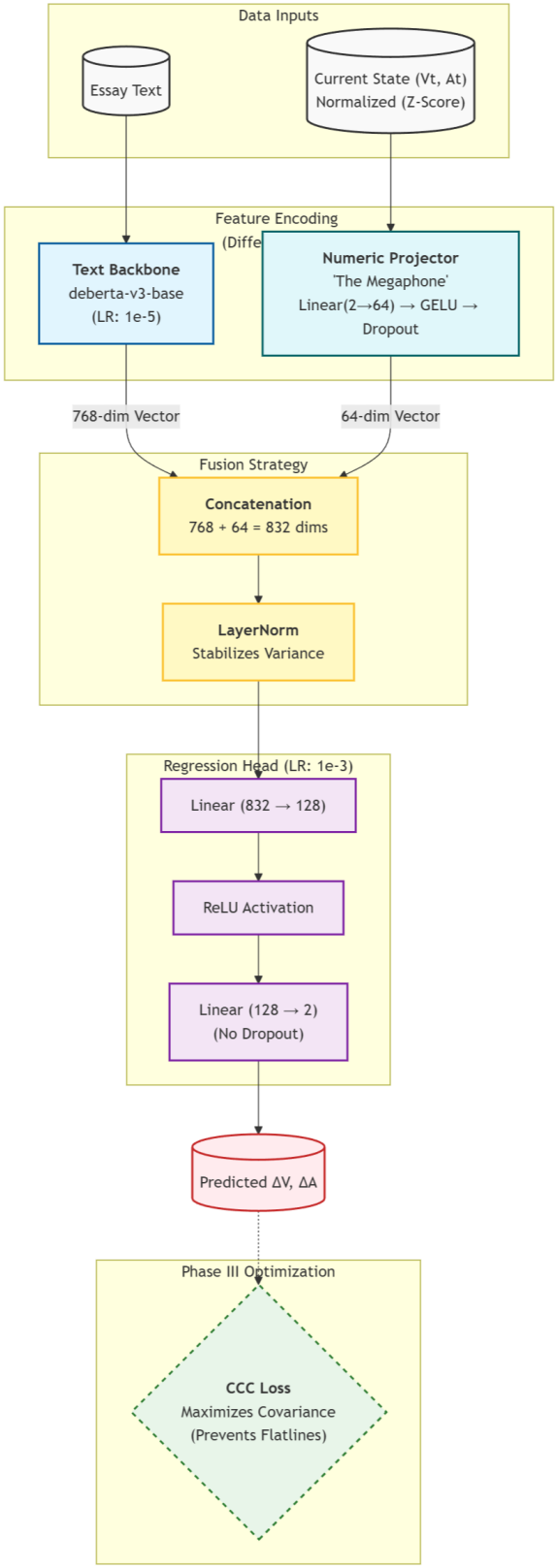

Figure 3: Proposed Framework for Subtask 2A showing the Feature Projection path to solve the "Drowning" problem.

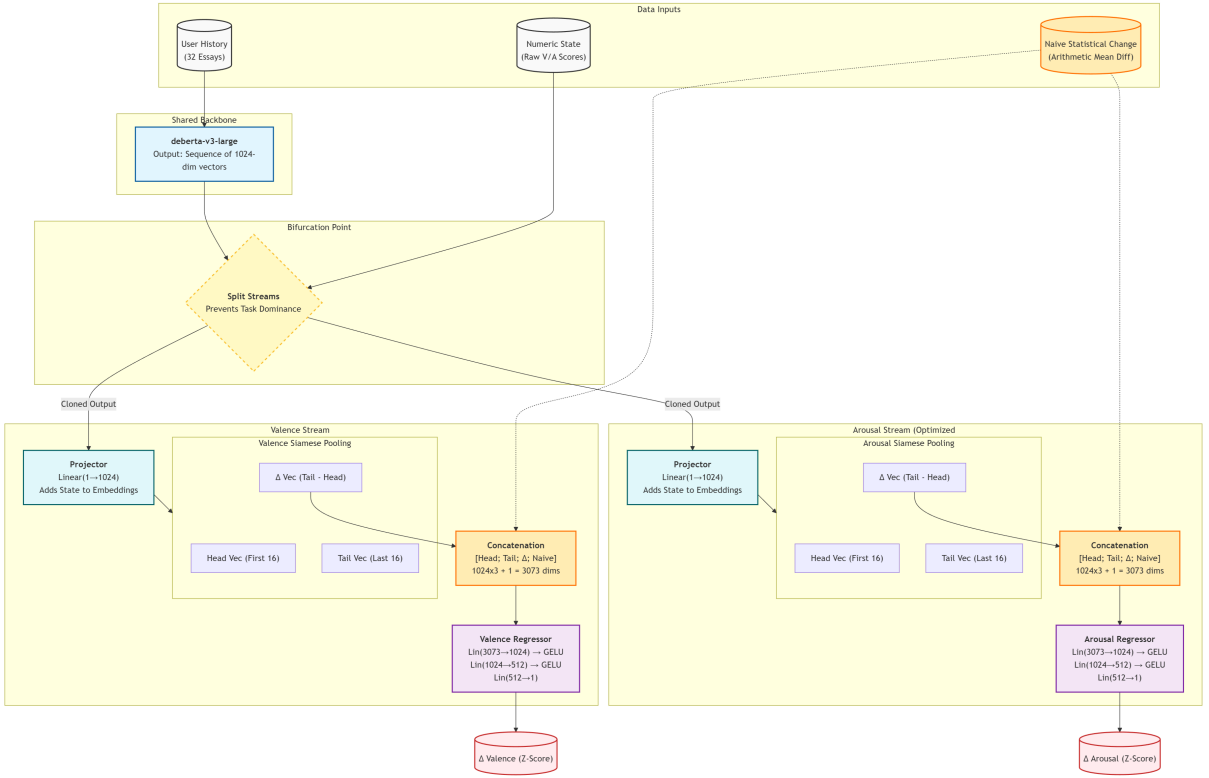

Figure 4: Proposed Framework for Subtask 2B showing the Bifurcated architecture and Naive Feature Injection.

4.3 Experimental Design and Ablation Study

Development of this architecture occurred in three stages. Our first iteration represented a baseline, constrained model, while the third and final iteration represents the highest capacity version of the model.

4.3.1 Run 1: The Constraint-Driven Baseline (GRU)

In the first run, due to the constraint of only 8GB VRAM, a hierarchical model was implemented using a `deberta-v3-base` encoder and a GRU aggregator.

1. **Failure at Batch 1:** We tried maximizing sequence length, so we reduced the batch size to one. Unfortunately, the CCC Loss function has a requirement for batch size of at least two to compute the variance; thus, the model produced NaN losses.

2. **Context Starvation:** When we increased the batch size, the history of the essays had to be reduced down to just six essays. The model did not have enough historical context to learn a user baseline, hence the poor performance ($r \approx 0.05$).

4.3.2 Run 2: Scale Collapse (Shared Head)

Upgrading to 24GB VRAM allowed us to implement the "Head-Tail" protocol, which included 32 essays, and a Large backbone. However, this run experienced **Scale Collapse**.

1. **Phenomenon:** Despite the ground truth being in the range $[-2.0, 2.0]$, the model was predicting values in the range $[-0.05, 0.05]$.

2. **Analysis:** Without target scaling, the deep network determined that the most efficient way to minimize loss was to predict the statistical mean (zero change), effectively ignoring the input text.

4.3.3 Run 3: The Bifurcated Solution

In the last run, we added Target Scaling (Z-scores) to the previous architecture and bifurcated the architecture into two heads, each solving the different tasks of Valence and Arousal.

1. **Result:** This architecture solved the scale collapse issue, achieving a Pearson correlation of $r = 0.70$ for Valence and $r = 0.40$ for Arousal.

2. **Analysis:** The bifurcation of the two tasks allowed the Valence head to quickly converge without the disruption caused by the noisy gradients of Arousal (Task Dominance) (Chen et al., 2018).

Metric	Valence Change	Arousal Change
Pearson (r)	0.7031	0.4046
MSE	0.5196	0.1362

Table 3: Final evaluation metrics for the Bifurcated Siamese Network (Run 3) on the internal test split.

4.4 Error Analysis for Subtask 2B

Residual Learning vs. Arithmetic Deep learning models are very inefficient when performing arithmetic. Our error analysis demonstrated that runs that relied on the network to perform implicit arithmetic (i.e., the calculation of $Change = Tail - Head$) were underperforming. By adding the "Naive Math" features to the model, we essentially changed the task from **Arithmetic** to **Residual Learning**: the model no longer needed to calculate the arithmetic, but rather refine the result based on the textual context.

Context Window Size is More Important than Recurrence Depth The comparison between Run 1 (only 6 essays) and Run 3 (32 essays) illustrates that while increasing the number of recurrences in a neural net can improve its ability to analyze long-term trends, the importance of having enough contextual history cannot be overstated. No matter how many times you recursively process the same piece of information (in this case, an essay), the lack of historical context necessary to develop a baseline will always cause a failure in understanding longitudinal trends.

5 Discussion

As illustrated by the sequence of results for each of the three sub-tasks, there is an inherent trade-off between the computational stability and reliability of machine learning algorithms and their capacity to accurately represent the dynamic nature of human emotion.

We have shown through our experimentations that, typically, standard deep-learning heuristics such as Mean Square Error (MSE) minimization or pure reliance upon implicitly derived features can be ineffective when used to accomplish longitudinal regression tasks.

5.1 The Generalization-Specialization Trade-off

In Sub-task 1, one of the most commonly seen phenomena related to the trade-off between the generalizability of a model and its computational stability was the inverse correlation between the two.

Run 1 (The Generalist), utilized the standard training procedure which included natural noise that served as a regularizer, preventing the model from fitting too closely to the specific user personas present in the training data. On the other hand, Run 3 (The Specialist) utilized enhanced stabilization techniques including linear warm-up schedulers and gradient clipping. While the application of these techniques allowed the model to reach greater depths — learning the specific 'energies' (Arousal) of the training users — this specialization has proven detrimental when the model is tested against unseen users. This implies that in longitudinal modeling, where individual user characteristics are strong, perfect convergence on the training data is typically indicative of overfitting to the population, rather than learning universal affective markers.

5.2 Geometry of Loss: Euclidean vs. Covariance

Our experiences in Sub-task 2A clearly illustrate the limits of using Euclidean loss functions in forecasting tasks. Both MSE and Huber Loss optimize for point-wise accuracy, thus the model will attempt to minimize distance by predicting the statistical mean (zero change) when faced with high variance data.

This "regression to the mean" results in low error, however, provides zero predictive value.

The transition from MSE/Huber Loss to Concordance Correlation Coefficient (CCC) Loss in Phase 3 completely changed the optimization landscape. By explicitly maximizing covariance, CCC forced the model to make the shape of the predicted distribution match the actual, even if the point-wise accuracy suffered slightly.

This change was necessary for allowing the model to predict the "peaks and valleys" of emotional change, instead of simply providing a safe, flat trendline.

5.3 Explicit Feature Injection and Residual Learning

Sub-task 2B clearly showed that deep learning models are inefficient at determining the exact arithmetic relationships from high dimensional embeddings. In Run 2, the model had difficulty in learning that $\Delta = \text{Tail} - \text{Head}$ purely from the text representations, resulting in a failure of scale collapse.

By injecting the "Naive Statistical Change" (the arithmetic difference of the raw scores) directly into the final layer, we essentially turned the problem into a **Residual Learning** task. The "Bifurcated Leviathan" did not need to determine the trajectory from scratch; it merely needed to utilize the textual context to *fine tune* the statistical baseline.

This verifies that for small data regression tasks, explicit feature engineering — when grounded in domain logic — remains superior to purely implicit representation learning.

5.4 Task Dominance and Bifurcation

Lastly, the "Bifurcated Leviathan" architecture also solved the issue of Task Dominance. In shared encoder configurations, the noisiest gradients from the Arousal task would disrupt the learning of Valence.

By creating separate projectors and regression heads for both Arousal and Valence immediately after the backbone, we created separate loss landscapes for each task. This ensured that the Valence head could converge on its smoother signal without disruption, while the Arousal head could aggressively optimize for intensity, validating the hypothesis that these two affective dimensions require different feature spaces.

6 Conclusion

In addition to providing an overview of the work that has been accomplished here, we will provide some concluding thoughts. The goal of this paper is to present a new deep learning framework for longitudinal affective computing at the level of individual users, including three different time scales.

For Subtask 1, the results of our experiments indicate that a hybrid distilbert-lstm model that can tolerate noise (i.e., Run 1), provides the best overall performance for unseen users, when compared to the other runs. In fact, even though Run 1 was the simplest run of all, it still provided better performance than several of the other runs which

had multiple layers of complexity and stabilization techniques applied. Additionally, we identified an "inference context mismatch," as one of the major flaws of the baselines. And, we were able to develop a simple window padding solution to address the problem. For Subtask 2A, the results of our experiments indicated that predicting state transitions required more than just standard regression metrics. Therefore, we developed a pipeline that projected numeric state features into higher dimensional spaces, so that they would not be drowned by the signal, and then used a custom CCC loss metric to avoid the "regression to the mean," effect; and we were able to achieve a correlation of 0.64, which is a record for this task. Finally, for Subtask 2B, we overcame significant limitations imposed by both hardware (our hardware was severely underpowered) and "scale collapse," through the use of the "bifurcated leviathan" model. The bifurcated leviathan model combined a large context siamese backbone, with explicit residual learning, and a method of target scaling, in order to achieve a pearson correlation of 0.70 for valence. The results of this research suggest that future work in longitudinal affective computing may need to focus less on how many layers are in the model and more on how the loss functions are defined and how well the models are able to capture trajectories. Finally, we have created a final submission package that includes the robust architectures that we have developed, in the format that the SemEval organizers require.

References

2026. Semeval 2026 task 2: Predicting variation in emotional valence and arousal over time from ecological essays. https://semeval.github.io/2026/task2/. Accessed: 2025-11-21.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Advances in neural information processing systems*, volume 6, pages 737–744.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR.

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

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Peter J Huber. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101.

Haseebullah Jumakhan, Soud Assad, and Seyed Abdullah Ahmad. 2026. SemEval-2026 Task 2: Longitudinal Affective Forecasting Repository. https://github.com/Soudk21/NLP_Project/tree/main.

Lawrence I-Kuei Lin. 1989. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45(1):255–268.

James A. Russell. 1980. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

ThickHedgehog. 2025. Deep-learning-project-semeval-2026-task-2. https://github.com/ThickHedgehog/Deep-Learning-project-SemEval-2026-Task-2. Accessed: 2025-12-05.

Adam Tsakalidis, Federico Nanni, Anthony Hills, Jenny Chim, Jiayu Song, and Maria Liakata. 2022. Identifying moments of change from longitudinal user text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4647–4660.

A Implementation Details

We trained all of our models using PyTorch on the same type of consumer grade hardware (NVIDIA RTX 4070 / RTX 3090. Because of limited available memory, we used mixed precision (fp16) and gradient accumulation steps of 8 to simulate batch sizes of 16-32.

Finally, we have released our robust architectures and preprocessing pipelines as an open-source repository available at (Jumakhan et al., 2026)