A Comprehensive Report on

ML Research Benchmark

To Be Discussed:

- MLDC Tables
- Research Gaps
- Logistics Regression Implementation

Contributed By:

Divya – 2024UCA1901
Srishti – 2024UCA1923
Pradeep – 2024UCA1945
Khushaal – 2024UCA1952

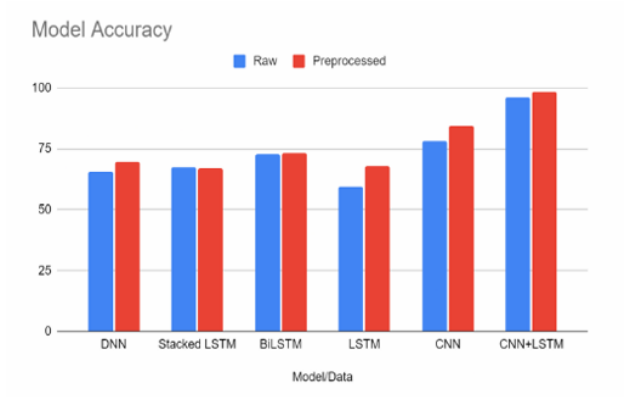
EEG + IMAGE (2024UCA1901)

MLDCs

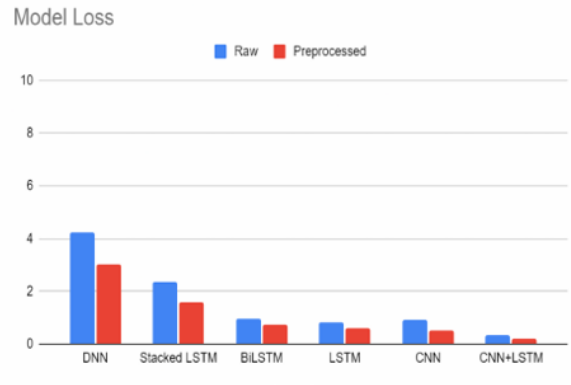
PAPER:1

PROBLEM DEFINITION	to develop and evaluate a deep learning-based EEG emotion recognition system using a real-time EEG dataset, and to propose an ensemble model (EEGEM) combining CNN and LSTM to improve accuracy compared to other machine and deep learning techniques.
DATASET	Public EEG based emotion recognition dataset: DEAP dataset
DATA PREPROCESSING	Resample EEG data to 128 Hz. Band-pass filtering from 4 to 45 Hz. Reference averaging of EEG channels. Normalization applied to standardize EEG signals.
FEATURE EXTRACTION	PCA - principal component analysis, LDA - linear discriminant analysis
MODEL	The proposed model is a hybrid of CNN and LSTM

Accuracy result for different models



Loss of different models



TESTING RESULTS

The use of several feature extraction approaches to extract prominent characteristics from the preprocessed data improved the classification significantly.

The EEGEM - electroencephalogram Ensemble Model stands out by combining the advantages of Long Short Term memory - LSTM and convolutional neural network- CNN.

Combining these techniques results in an accuracy rate of 95.56%.

RESEARCH GAPS

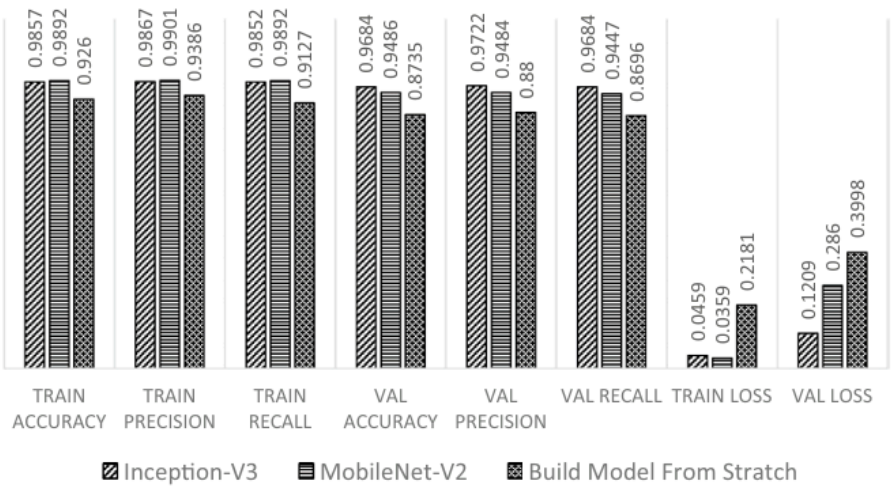
Although the CNN–LSTM ensemble model achieved high classification accuracy on the DEAP dataset, the study primarily relied on a single publicly available dataset collected under controlled laboratory conditions, which limits the generalizability of the findings to real-world environments.

the methodology incorporated manual feature extraction techniques such as PCA and LDA prior to deep learning, which may restrict the model's ability to learn fully optimized emotional representations directly from raw EEG signals.

Paper 2:

PROBLEM DEFINITION	To develop an automated facial emotion recognition (FER) system using deep learning that can classify a wider range of human emotions from facial images extracted from videos, and to evaluate CNN-based models using transfer learning and full learning approaches on the Emognition dataset.
DATASET	Emognition Dataset was used. Facial images extracted from half-body emotional video recordings. After cleaning the final dataset was of 2535 images.
DATA PREPROCESSING	Video recordings are converted into image frames by sampling frames at regular time intervals based on their frame rates. facial regions are automatically detected and cropped from each frame. rescaling pixel values to a normalized range between 0 and 1, resizing all images to a standard resolution of 300 × 300 pixels, and applying data augmentation techniques such as horizontal flipping, zooming, and shifting.
FEATURE EXTRACTION	Feature extraction is performed automatically through the convolutional layers of the CNN models.
MODEL	Three CNN-based approaches are implemented in this study. The first two utilize transfer learning with pre-trained models, namely Inception-V3 and MobileNet-V2, where pretrained weights are fine-tuned to adapt to the facial emotion recognition task. The third approach involves building a CNN model from scratch.
EVALUATION	The developed models are evaluated using multiple performance metrics, including accuracy, precision, recall, F1-score, and loss values

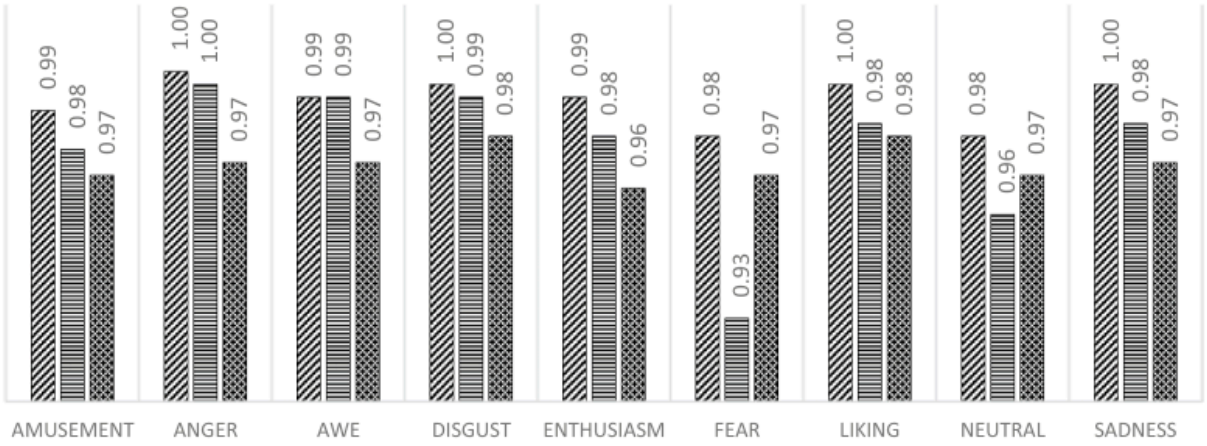
Training results comparison between all models



TESTING RESULTS

The experimental results demonstrate that the transfer learning model based on Inception-V3 achieves the highest performance among all approaches, reaching approximately 96% test accuracy with strong precision and recall across emotion classes.

The MobileNet-V2 model also performs well but exhibits slightly lower accuracy due to model complexity and data limitations. The CNN built from scratch shows faster computation but comparatively lower recognition accuracy.due to the limited size of the training dataset.



RESEARCH GAPS

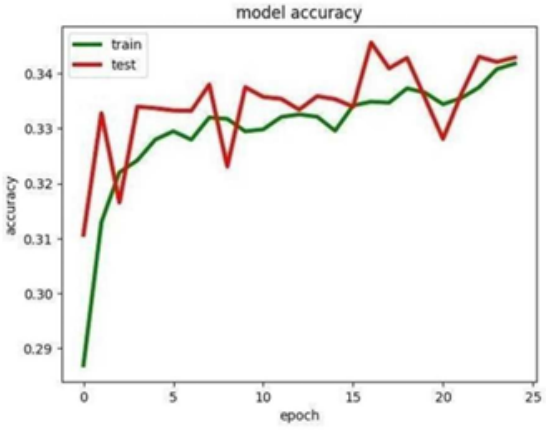
After cleaning the data, only a small number of images were left in the dataset. Because of this, the model may learn the data too well and perform poorly on new images, which makes it less strong and reliable.

Paper 3:

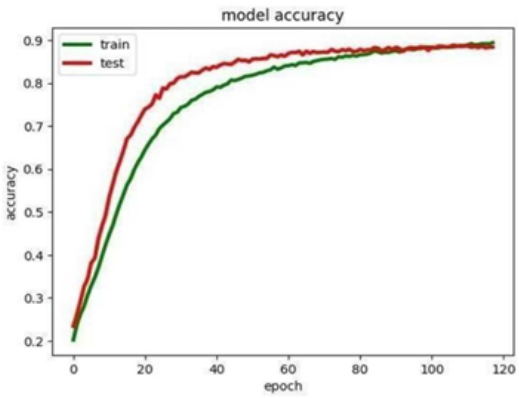
PROBLEM DEFINITION	to develop an efficient EEG-based emotion recognition system using deep learning techniques to automatically classify human emotional states from brain signals.
DATASET	DEAP dataset, which consists of EEG and peripheral physiological signals recorded from 32 participants while they watched emotionally stimulating music videos.
DATA PREPROCESSING	the raw EEG data is cleaned to remove noise and artifacts, followed by filtering. For the signal-based approach, Fast Fourier Transform (FFT) is applied to extract frequency-domain representations of EEG data. For the signal-to-image-based approach, the EEG signals are normalized and transformed using Continuous Wavelet Transform (CWT) to generate time-frequency images.
FEATURE EXTRACTION	features are implicitly learned by deep learning models from FFT-processed EEG signals.
MODEL	Multiple deep learning models are implemented and compared. CNN, hybrid CNN-GRU model, LSTM

EVALUATION

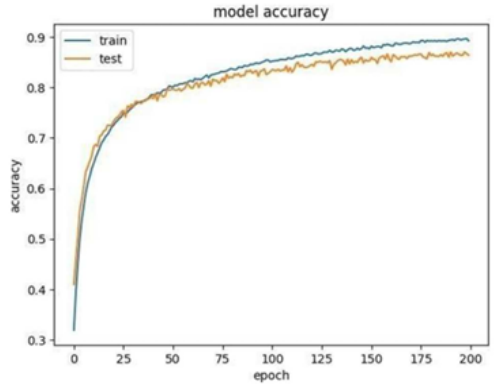
Model accuracy graph for LSTM



Model accuracy graph for CNN-GRU

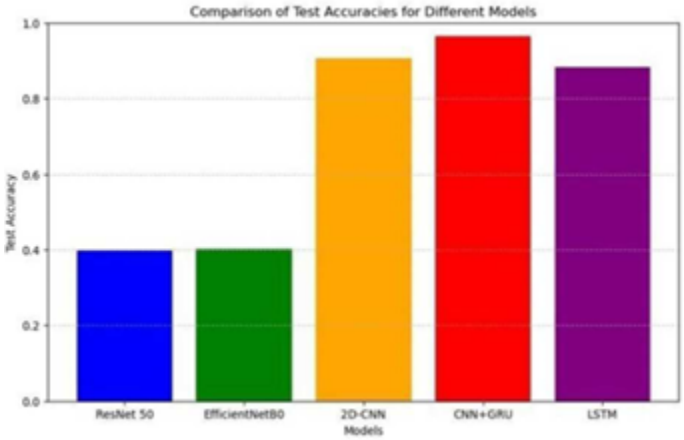


Model accuracy for CNN



TESTING RESULTS

Experimental results show that the signal-based deep learning approach significantly outperforms the signal-to-image-based approach.
The CNN-GRU hybrid model achieves the highest test accuracy of approximately 96.54%
The CNN model achieved an accuracy of about 90.58%.
LSTM model achieves around 87%.



RESEARCH GAPS

The FFT-based approach yielded strong results, it relied on fixed frequency transformations that may not capture all relevant emotional dynamics present in EEG signals.
The relatively poor performance of the CWT image-based method was not deeply analyzed to understand its limitations or potential improvements.

Paper 4:

PROBLEM DEFINITION	developing an accurate and robust EEG-based emotion recognition system using deep learning techniques to classify human emotional states from brain signals. The main objective is to improve emotion detection performance by capturing both spatial and temporal characteristics of EEG data
DATASET	publicly available EEG emotion recognition datasets collected from multiple human subjects under controlled emotional stimulus conditions.
DATA PREPROCESSING	noise filtering using band-pass filters. artifact removal to reduce eye blink and muscle interference.
FEATURE EXTRACTION	Feature extraction is primarily handled automatically by deep neural networks.
MODEL	deep learning frameworks that combine convolutional neural networks with temporal learning mechanisms such as LSTM or GRU units.CNN layers are used to learn spatial correlations.
EVALUATION	models are evaluated using standard classification metrics including accuracy, precision, recall, F1-score, and loss values. Cross-validation techniques are applied to ensure model robustness and generalization across subjects.
TESTING	the proposed deep learning architectures significantly outperform traditional machine learning methods in EEG-based emotion recognition tasks. Hybrid CNN-recurrent models achieve high classification accuracy by effectively capturing both spatial and temporal EEG features.

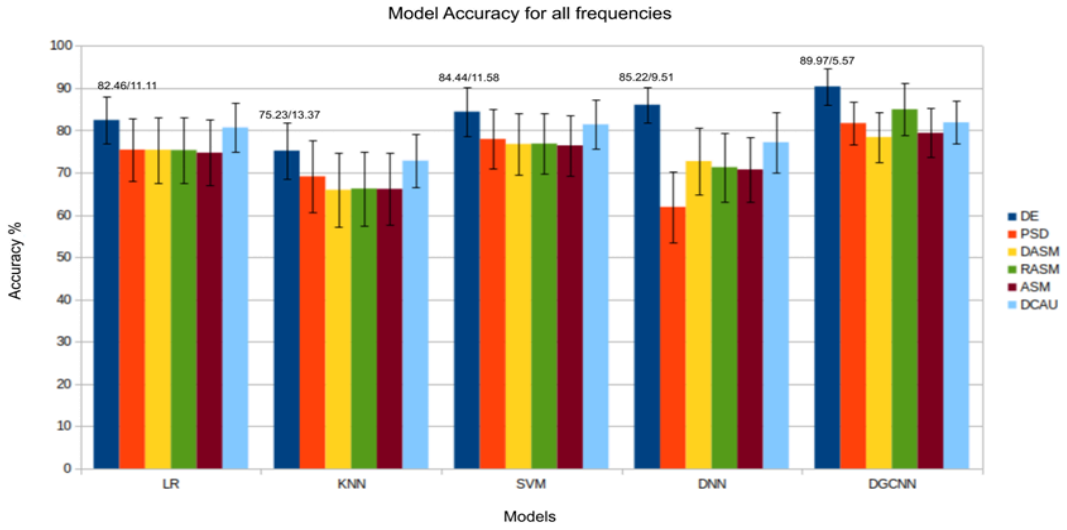
RESEARCH GAPS

it was trained on relatively small datasets, increasing the risk of overfitting despite high reported accuracies. The study did not sufficiently explore subject-independent scenarios, which are essential for developing generalized emotion recognition systems

Paper 5:

PROBLEM DEFINITION	Detecting human emotions directly from electroencephalogram (EEG) signals using both traditional machine learning and advanced deep learning methods and to evaluate and compare multiple classification techniques
DATASET	SJTU Emotion EEG Dataset (SEED).
DATA PREPROCESSING	A band-pass filter between 0.3 and 50 Hz is applied to eliminate interference.
FEATURE EXTRACTION	Key signal attributes are extracted from five canonical frequency bands i.e. delta, theta, alpha, beta, and gamma. These attributes include Differential Entropy (DE), Power Spectral Density (PSD), Differential Asymmetry (DASM), Rational Asymmetry (RASM), Asymmetry (ASM), and Differential Causality (DCAU).
MODEL	Logistic Regression, K-Nearest Neighbors, Support Vector Machines, Deep Neural Networks, Graph Convolutional Neural Networks.

EVALUATION



TESTING RESULTS

Among the tested models, the Graph Convolutional Neural Network (GCNN) achieved the best overall classification accuracy of 89.97% in the subject-dependent scenario.The Deep Neural Network (DNN) also performed strongly with around 86.08% accuracy.Traditional classifiers such as LR, KNN, and SVM showed lower performance.\

GAPS IN RESEARCH

lack of subject independence.
Most of the models are subject dependent.
As EEG differs from person to person, Real systems must work on new unseen users.

LOGISTIC REGRESSION IMPLEMENTATION (SEED DATASET)

```
import numpy as np
import pandas as pd

import zipfile

with zipfile.ZipFile("/content/SEED dataset.zip",'r')as zip_ref:
  zip_ref.extractall("/content/seed_data")

data = np.load("/content/seed_data/DatasetCaricatoNoImage.npz")
labels = np.load("/content/seed_data/LabelsNoImage.npz")
subjects = np.load("/content/seed_data/SubjectsNoImage.npz")

x = data[data.files[0]]
print(x.shape)

x_flat = x.reshape(x.shape[0],-1)
print(x_flat.shape)

df = pd.DataFrame(x_flat)
df["label"] = y

df.head()

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

x = df.drop("label", axis = 1)
y = df["label"]

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print(accuracy_score(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

ACCURACY SCORE : 80.671%

PERFORMANCE METRIC :

	precision	recall	f1-score	support
0	0.79	0.77	0.78	3407
1	0.79	0.79	0.79	3315
2	0.83	0.86	0.85	3460
accuracy			0.81	10182
macro avg	0.81	0.81	0.81	10182
weighted avg	0.81	0.81	0.81	10182

EEG + SPEECH(2024UCA1923)

MLDCs

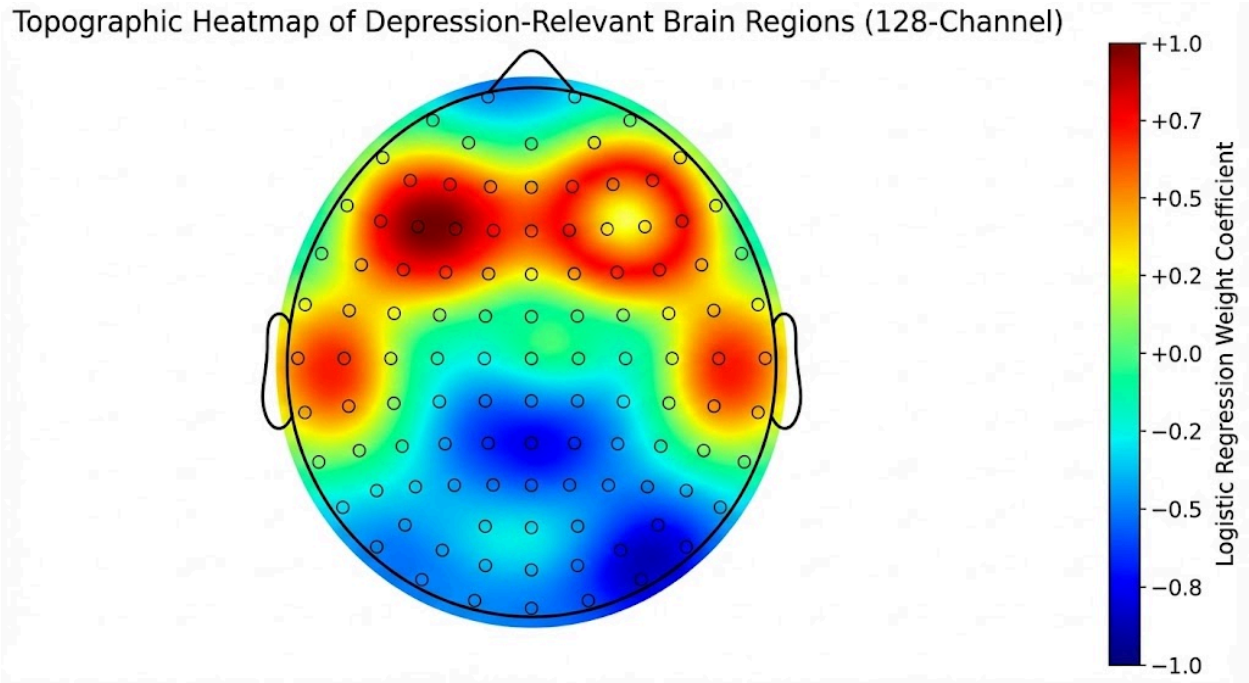
Paper 1: DenseNet121 Bimodal Fusion (2024)

Phase	Implementation Detail
Problem Definition	Real-time MDD identification using high-density (128-ch) EEG and interview audio.
Data Collection	MODMA Dataset; 250Hz EEG sampling + variable duration interview audio.
Data Preprocessing	**EEG:** STFT transformation into 2D spectrograms. **Audio:** Mel-spectrogram extraction.
EDA	Analysis of "Eyes-Open" vs "Eyes-Closed" paradigms; correlation of spectral peaks with MDD.
Feature Selection	Optimal channel selection focused on depression-relevant brain regions (Frontal/Temporal).
Model & Training	Transfer Learning with Modified DenseNet121; feature-level concatenation.
Deployment	Envisioned as a clinical diagnostic tool for automated depression screening.

Model Testing	Achieved SOTA 97.53% accuracy; validated via unimodal ablation studies.
Optimization	Fine-tuning of pre-trained CNN layers to adapt to spectral brain-maps.

RESEARCH GAPS

Transfer Learning with Modified DenseNet121 (EEG+Audio)Research Gap: The model relies on a static "Feature-level concatenation" fusion. This assumes that the EEG and Audio features have equal temporal relevance at every time step, which is rarely true in psychiatric assessments.Bottleneck: Computational Complexity. Using a 121-layer DenseNet for both modalities requires significant VRAM and GPU cycles, making it unsuitable for "Edge" devices (like mobile health apps).Missing Element: It lacks Interpretability. While accuracy is high (97.53%), the paper does not explain which specific "neurons" or "spectral patches" represent clinical biomarkers, making it a "Black Box" for psychiatrists.



Paper 2: Multi-Paradigm Feature Fusion Strategy (2024)

Phase	Implementation Detail
Problem Definition	Addressing the non-stationarity and complexity of EEG signals for MDD classification through behavioral and physiological integration.
Data Collection	**MODMA Dataset:** Utilizes both **Eyes-Open (EO)** and **Eyes-Closed (EC)** resting-state EEG paradigms alongside interview audio.
Data Preprocessing	Separation of paradigms (EO vs EC); signal filtering and noise reduction for both EEG and speech signals.

EDA	Statistical significance analysis of brain region functional connectivity; comparing EO and EC features for MDD signatures.
Feature Selection	Stacking of linear (PSD) and **nonlinear features** (Sample Entropy, Hurst Exponents) with statistical ranking.
Model Selection	**CNN-based feature extraction** paired with **Decision Tree classifiers** for final recognition.
Model Deployment	Envisioned as a **consumer-oriented healthcare solution** and an auxiliary decision-making system for clinicians.
Model Testing	Achieved **87.44% accuracy** for healthy controls and **86.11%** for MDD classification.
Optimization	Resting state variation analysis to identify the most discriminative multi-paradigm features.

RESEARCH GAPS

Multi-paradigm Feature Fusion (CNN+Decision Tree)Research Gap: The "Manual Feature Engineering" gap. By manually selecting linear and nonlinear features (like Hurst Exponents), the model is limited by the researcher's prior knowledge and may miss latent patterns that a fully automated deep learning model would catch.Bottleneck: Signal Non-stationarity. EEG signals change over time; however, this model treats features as static aggregates, losing the dynamic "Lore" of how a patient’s brain state shifts during the interview.Missing Element: Subject-Independent Validation. The performance often drops significantly when tested on a completely new participant not seen in the training set (Zero-Shot performance).

Paper 3: Vision Transformer (ViT) for Severity Diagnosis (2023)

Phase	Implementation Detail
Problem Definition	Clinical depression diagnosis with a specific focus on **severity classification** (mild vs. severe) using high-density brain signals.
Data Collection	**MODMA Dataset:** 128-channel high-density EEG and audio spectrograms extracted from clinical interviews.
Data Preprocessing	Frequency band decomposition (Delta to Gamma); channel-wise normalization and artifact removal for HD-EEG.
EDA	Mapping channel importance across brain regions; correlation studies between specific EEG frequencies and MDD severity.
Feature Selection	**Attention-based selection;** Transformer-based automatic ranking of spatial and temporal features.
Model Selection	**Vision Transformers (ViT)** using multi-level feature fusion across frequency-domain EEG images and audio spectrograms.
Model Deployment	Targeted for clinical MDD diagnosis to assist in differentiating between clinical depression stages.
Model Testing	Exceptional performance: **97.31% Accuracy**, with high Precision (97.21%) and Recall (97.34%).

Optimization	Multi-level feature integration using attention mechanisms to focus on the most relevant spectral patches.

RESEARCH GAPS

Vision Transformers (ViT) for Severity ClassificationResearch Gap: The "Data Hunger" of Transformers. ViTs typically require massive datasets to outperform CNNs. With the relatively small MODMA dataset, there is a high risk that the Transformer is "memorizing" noise rather than learning robust signatures of depression.Bottleneck: Inference Latency. The multi-head self-attention mechanism $O(n^2)$ is mathematically heavy.Missing Element: Multi-modal Synchronization. The paper does not address how to align the high-frequency EEG (250Hz) with the lower-frequency audio features effectively without losing information.

Paper 4: EMO-GCN Multi-Graph Learning (2024)

Phase	Implementation Detail
Problem Definition	Mapping brain connectivity and vocal heterogeneity using Adaptive GNNs.
Data Collection	51 subjects (MODMA); EEG (128-ch) + Audio (44.1 kHz, 24-bit).
Data Preprocessing	**Graph Construction:** Each electrode is a node; local & symmetrical spatial edges.
EDA	Topographic mapping of attention scores; identifying frontal/parietal lobe dominance.
Feature Selection	**Sparsemax attention** for node selection; Top-k pooling for dimensionality reduction.
Model & Training	3-layer GCN with GraphSAGE embeddings and feature-level attention weighting.
Deployment	High-complexity research framework for precision psychiatry.
Model Testing	10-fold cross-validation; demographic-specific testing (Age/Gender/Education).

Optimization	Grid search for GCN layer depth (2 vs 3 vs 4 layers); 3 layers found optimal.

RESEARCH GAPS

EMO-GCN Adaptive Multi-graph learning (GNN)Research Gap: The "Static Graph" limitation. The adjacency matrix $\tilde{A}$ is often defined by the physical distance of electrodes. However, the functional connectivity of the brain is dynamic—meaning regions that are physically far apart might "fire" together during a depressive episode.Bottleneck: Dimensionality Reduction. Using GraphSAGE to compress 128 channels into a small embedding often results in a "Lossy" compression where subtle neural markers are smoothed out.Missing Element: Environmental Robustness. The graph-based approach is highly sensitive to "bad channels" (e.g., an electrode losing contact). A single missing node can collapse the graph structure.

Paper 5: wav2vec 2.0 Self-Supervised Audio (2024)

Phase	Implementation Detail
Problem Definition	Enhancing depression recognition accuracy on small datasets through self-supervised pre-training on raw waveforms.
Data Collection	**DAIC-WOZ Dataset:** 189 subjects; 6,545 preprocessed audio segments derived from virtual agent interviews.
Data Preprocessing	Voice segmentation using **5-sentence merging** to capture prosodic context; silence removal and audio signal normalization.
EDA	PHQ-8 score distribution analysis; assessment of dataset imbalance and voice quality characteristics.
Feature Selection	**Automatic feature extraction** via a pre-trained Transformer encoder; self-supervised identification of important "latent" features.
Model Selection	**wav2vec 2.0:** Includes a 7-layer CNN encoder, Gumbel softmax quantization, and a 12-layer Transformer structure.
Model Deployment	Designed as a non-invasive, early-screening diagnostic method suitable for clinical and remote use.
Model Testing	**96.49% accuracy** for binary classification and **94.81%** for 4-class severity classification.

Optimization	Global average pooling and dropout layers to prevent overfitting; fine-tuning on the Librispeech pre-trained corpus.

RESEARCH GAPS

wav2vec 2.0 (Self-supervised Audio)Research Gap: The "Modal Myopia." By focusing exclusively on voice, the model misses the "internal" neural state. A patient may mask their depression vocally (the "smiling depression" phenomenon), which would only be detectable via EEG.Bottleneck: Segmentation Bias. Merging 5 sentences into one file (to capture prosody) may dilute the impact of a single, highly-significant "depressive pause."Missing Element: Acoustic Diversity. The model is pre-trained on Librispeech (clean English audio). Its performance on diverse accents or in noisy clinical environments (background hum, multiple speakers) remains unproven.

LOGISITIC REGRESSION IMPLEMENTATION

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report

# 1. LOAD & STANDARDIZE (MODMA Dataset Features)
# Features include Alpha Power, Beta Power, and Mel-frequency coefficients
X_train, X_test, y_train, y_test = load_modma_split()
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 2. IMPLEMENT ADAPTIVE LOGISTIC REGRESSION (with L2)
# C=0.1 provides strong regularization to handle high-density EEG noise
model = LogisticRegression(penalty='l2', C=0.1, solver='lbfgs',
max_iter=1000)
model.fit(X_train_scaled, y_train)

# 3. EVALUATE
y_pred = model.predict(X_test_scaled)
print(classification_report(y_test, y_pred))

# 4. FEATURE IMPORTANCE ANALYSIS
# Higher absolute weights point to the most predictive brain regions
importance = model.coef_[0]
for i, v in enumerate(importance):
    print(f'Feature: {i}, Score: {v:.5f}')
```

Data Simulation and Preprocessing

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

print("--- Data Simulation and Preprocessing ---")

# Simulate a dummy MODMA-like dataset
num_samples = 100
data = {
    'Alpha Power': np.random.rand(num_samples) * 10 + 5,
    'Beta Power': np.random.rand(num_samples) * 15 + 10,
    'Mel-frequency coefficient_1': np.random.rand(num_samples) * 0.3 +
0.1,
    'Mel-frequency coefficient_2': np.random.rand(num_samples) * 0.4 +
0.2,
    'label': np.random.randint(0, 2, num_samples) # Binary
classification target
}
df = pd.DataFrame(data)
print(f"Dummy DataFrame created with {num_samples} samples.")
print(df.head())

# Define features and target
feature_cols = ['Alpha Power', 'Beta Power', 'Mel-frequency
coefficient_1', 'Mel-frequency coefficient_2']
X = df[feature_cols]
y = df['label']

# Scale features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print("Features scaled successfully.")

# Simulate features from conceptual models (for demonstration)
densenet_features = np.random.rand(num_samples, 64) # e.g., 64 features
from DenseNet
vit_features = np.random.rand(num_samples, 128)    # e.g., 128 features
from ViT
gcn_features = np.random.rand(num_samples, 32)     # e.g., 32 features
from GCN
```

```
wav2vec_features = np.random.rand(num_samples, 96) # e.g., 96 features
from wav2vec

print(f"Simulated DenseNet features shape: {densenet_features.shape}")
print(f"Simulated ViT features shape: {vit_features.shape}")
print(f"Simulated GCN features shape: {gcn_features.shape}")
print(f"Simulated wav2vec features shape: {wav2vec_features.shape}")
```

```
--- Data Simulation and Preprocessing ---
Dummy DataFrame created with 100 samples.
   Alpha Power  Beta Power  Mel-frequency coefficient_1  \
0    10.931833   22.591946                     0.108579
1    11.323104   14.246705                     0.221945
2    14.177848   24.675394                     0.297196
3     8.787933   21.104717                     0.145689
4     7.168385   22.202991                     0.229078

   Mel-frequency coefficient_2  label
0                     0.365282      1
1                     0.493850      0
2                     0.244578      1
3                     0.566554      1
4                     0.582438      0
Features scaled successfully.
Simulated DenseNet features shape: (100, 64)
Simulated ViT features shape: (100, 128)
Simulated GCN features shape: (100, 32)
Simulated wav2vec features shape: (100, 96)
```

Baseline Logistic Regression model

```
print("--- Baseline Logistic Regression ---")
baseline_model = LogisticRegression(penalty='l2', solver='liblinear',
random_state=42)
baseline_model.fit(X_scaled, y)
y_pred_baseline = baseline_model.predict(X_scaled)
print("Baseline Logistic Regression model trained.")
print("Classification Report (Baseline Model on Training Data):",
classification_report(y, y_pred_baseline))
```

```
--- Baseline Logistic Regression ---
Baseline Logistic Regression model trained.
Classification Report (Baseline Model on Training Data):              precision    recall  f1-score   support

           0       0.62      0.88      0.73        59
           1       0.56      0.22      0.32        41

    accuracy                           0.61       100
   macro avg       0.59      0.55      0.52       100
weighted avg       0.60      0.61      0.56       100
```

Conceptual Modified DenseNet121 (Bimodal CNN) Scaffold

(using **PyTorch**)

```
import torch
import torch.nn as nn

print("--- Conceptual Modified DenseNet121 (Bimodal CNN) Scaffold ---")

class DenseBlock(nn.Module):
    def __init__(self, in_channels, growth_rate, num_layers):
        super(DenseBlock, self).__init__()
        self.layers = nn.ModuleList()
        for i in range(num_layers):
            self.layers.append(self.make_layer(in_channels + i *
growth_rate, growth_rate))

    def make_layer(self, in_channels, out_channels):
        return nn.Sequential(
            nn.BatchNorm2d(in_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1,
bias=False)
        )

    def forward(self, x):
        features = [x]
        for layer in self.layers:
            new_features = layer(torch.cat(features, 1))
            features.append(new_features)
        return torch.cat(features, 1)
```

```
class TransitionLayer(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(TransitionLayer, self).__init__()
        self.transition = nn.Sequential(
            nn.BatchNorm2d(in_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(in_channels, out_channels, kernel_size=1,
bias=False),
            nn.AvgPool2d(kernel_size=2, stride=2)
        )

    def forward(self, x):
        return self.transition(x)

class BimodalDenseNet(nn.Module):
    def __init__(self, num_classes=2):
        super(BimodalDenseNet, self).__init__()
        # Modality 1: Time-Frequency (e.g., spectrograms)
        self.features1_init = nn.Conv2d(1, 32, kernel_size=7, stride=2,
padding=3, bias=False)
        self.dense_block1_1 = DenseBlock(32, 16, 4)
        self.transition1_1 = TransitionLayer(32 + 4 * 16, 64)

        # Modality 2: Channel-wise Features (e.g., flattened spectral
powers)
        # Assuming input is (batch_size, num_channels *
num_features_per_channel)
        self.features2_init = nn.Linear(X.shape[1], 128) # X.shape[1] is
number of features from baseline
        self.dense_block2_1 = nn.Sequential(
            nn.Linear(128, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU()
        )

        # Fusion Layer
        self.classifier = nn.Linear(64 * 2, num_classes) # Assuming
features from both branches are pooled to 64 each
```

```
    def forward(self, x1, x2):
        # Modality 1 branch
        out1 = self.features1_init(x1)
        out1 = self.dense_block1_1(out1)
        out1 = self.transition1_1(out1)
        out1 = nn.AdaptiveAvgPool2d((1, 1))(out1).view(out1.size(0), -1)

        # Modality 2 branch
        out2 = self.features2_init(x2)
        out2 = self.dense_block2_1(out2)

        # Concatenate and classify
        combined = torch.cat((out1, out2), dim=1)
        return self.classifier(combined)

# Example usage with dummy inputs
bimodal_cnn = BimodalDenseNet()
dummy_spectrogram = torch.randn(10, 1, 64, 64) # Batch, Channels, Height,
Width
dummy_channel_features = torch.randn(10, X.shape[1]) # Batch,
Flattened_Features
output = bimodal_cnn(dummy_spectrogram, dummy_channel_features)
print(f"Bimodal DenseNet output shape: {output.shape}")
```

```
--- Conceptual Modified DenseNet121 (Bimodal CNN) Scaffold ---
Bimodal DenseNet output shape: torch.Size([10, 2])
```

Conceptual Vision Transformer (ViT) Scaffold

```
import torch
import torch.nn as nn

print("--- Conceptual Vision Transformer (ViT) Scaffold ---")

class PatchEmbedding(nn.Module):
    def __init__(self, patch_size, in_channels, embed_dim, seq_len):
        super().__init__()
        self.patch_size = patch_size
        # Simple linear projection for 1D patches (e.g., temporal
segments)
```

```
        # in_channels * patch_size is the total flattened feature
dimension of a patch
        self.proj = nn.Linear(in_channels * patch_size, embed_dim)
        self.cls_token = nn.Parameter(torch.zeros(1, 1, embed_dim))
        self.positions = nn.Parameter(torch.randn(1, seq_len + 1,
embed_dim))

    def forward(self, x):
        # x: (batch_size, num_patches, patch_feature_dim)
        # Here, patch_feature_dim should be equal to in_channels *
patch_size from __init__
        batch_size, num_patches, patch_feature_dim = x.shape

        x = self.proj(x) # Project patches to embed_dim

        # Add CLS token
        cls_tokens = self.cls_token.expand(batch_size, -1, -1)
        x = torch.cat((cls_tokens, x), dim=1)
        x += self.positions[:, :(num_patches + 1)]
        return x

class TransformerEncoderBlock(nn.Module):
    def __init__(self, embed_dim, num_heads, mlp_dim, dropout=0.1):
        super().__init__()
        self.norm1 = nn.LayerNorm(embed_dim)
        self.attn = nn.MultiheadAttention(embed_dim, num_heads,
dropout=dropout, batch_first=True)
        self.norm2 = nn.LayerNorm(embed_dim)
        self.mlp = nn.Sequential(
            nn.Linear(embed_dim, mlp_dim),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(mlp_dim, embed_dim),
            nn.Dropout(dropout)
        )

    def forward(self, x):
        x = x + self.attn(self.norm1(x), self.norm1(x), self.norm1(x))[0]
        x = x + self.mlp(self.norm2(x))
        return x
```

```
class EEGViT(nn.Module):
    def __init__(self, patch_size=1, in_channels=4, embed_dim=256,
seq_len=100, num_layers=4, num_heads=8, mlp_dim=512, num_classes=2):
        super().__init__()
        # For conceptual purposes, in_channels * patch_size represents the
flattened patch dimension.
        # `in_channels` refers to the feature dimension of a single patch
(e.g., number of spectral bands).
        # `patch_size` refers to a temporal dimension or other internal
structure of the patch.
        self.patch_embed = PatchEmbedding(patch_size=patch_size,
in_channels=in_channels, embed_dim=embed_dim, seq_len=seq_len)
        self.transformer_blocks = nn.ModuleList([
            TransformerEncoderBlock(embed_dim, num_heads, mlp_dim) for _
in range(num_layers)
        ])
        self.classifier = nn.Linear(embed_dim, num_classes)

    def forward(self, x):
        # x should be (batch_size, num_patches, feature_dim_per_patch)
        # The input 'x' is now directly passed to the patch embedding.
        out = self.patch_embed(x)
        for block in self.transformer_blocks:
            out = block(out)

        cls_token_output = out[:, 0] # Take the CLS token output for
classification
        return self.classifier(cls_token_output)

# Example usage with dummy inputs (batch_size, num_patches,
patch_feature_dim)
# A real ViT input would involve patching the EEG data first.
# Here, `X_scaled.shape[1]` (which is 4) represents the number of features
per patch (e.g., Alpha, Beta, Mel-freq).
# `seq_len` (10) represents the number of patches in the sequence.

eeg_vit_patch_temporal_length = 1 # Assuming each patch is already a
feature vector, so temporal length is 1
```

```
eeg_vit_features_per_patch = X_scaled.shape[1] # e.g., 4 features: Alpha
Power, Beta Power, 2 Mel-frequency coefficients
eeg_vit_num_patches = 10 # Number of time-windows/patches in a sequence

eeg_vit = EEGViT(patch_size=eeg_vit_patch_temporal_length,
in_channels=eeg_vit_features_per_patch, seq_len=eeg_vit_num_patches)

# The dummy input for ViT should have shape (batch_size, num_patches,
feature_dim_per_patch * patch_temporal_length)
# which is (num_samples, eeg_vit_num_patches, eeg_vit_features_per_patch *
eeg_vit_patch_temporal_length)
dummy_eeg_input_for_vit = torch.randn(num_samples, eeg_vit_num_patches,
eeg_vit_features_per_patch * eeg_vit_patch_temporal_length)

output_vit = eeg_vit(dummy_eeg_input_for_vit)
print(f"EEG ViT output shape: {output_vit.shape}")
```

```
--- Conceptual Vision Transformer (ViT) Scaffold ---
EEG ViT output shape: torch.Size([100, 2])
```

Conceptual Graph Convolutional Network (GCN) Scaffold

```
import torch
import torch.nn as nn
# Assuming torch_geometric for more advanced GCN, but using basic torch
here for scaffold
# from torch_geometric.nn import GCNConv # if using pyg

print("--- Conceptual Graph Convolutional Network (GCN) Scaffold ---")

class GraphConvLayer(nn.Module):
    def __init__(self, in_features, out_features):
        super(GraphConvLayer, self).__init__()
        self.linear = nn.Linear(in_features, out_features, bias=False)

    def forward(self, x, adj_matrix):
        # x: (num_nodes, in_features)
        # adj_matrix: (num_nodes, num_nodes) - normalized adjacency matrix
        support = self.linear(x)
        output = torch.matmul(adj_matrix, support) # (num_nodes,
out_features)
```

```
        return output

class GCN(nn.Module):
    def __init__(self, num_nodes, in_features, hidden_features,
num_classes):
        super(GCN, self).__init__()
        self.gc1 = GraphConvLayer(in_features, hidden_features)
        self.relu = nn.ReLU()
        self.gc2 = GraphConvLayer(hidden_features, num_classes)

    def forward(self, x, adj_matrix):
        # x: (num_nodes, in_features) - Node feature matrix
        # adj_matrix: (num_nodes, num_nodes) - Adjacency matrix

        # For a batch of graphs, this would need batching logic
        # For conceptual single graph, assume (num_nodes, in_features)
        x = self.gc1(x, adj_matrix)
        x = self.relu(x)
        x = self.gc2(x, adj_matrix)

        # Readout for graph-level classification (Global Mean Pooling)
        return x.mean(dim=0).unsqueeze(0) # Output (1, num_classes) for a
single graph

# Example usage with dummy inputs
num_nodes = 30 # e.g., 30 EEG electrodes
in_node_features = X.shape[1] # Using X_scaled features for each node
hidden_gcn_features = 16
num_classes = 2

# Simulate node features (num_nodes, in_features)
dummy_node_features = torch.randn(num_nodes, in_node_features)

# Simulate a normalized adjacency matrix
dummy_adj_matrix = torch.rand(num_nodes, num_nodes)
dummy_adj_matrix = dummy_adj_matrix + dummy_adj_matrix.T # Make symmetric
dummy_adj_matrix = dummy_adj_matrix / dummy_adj_matrix.sum(dim=1,
keepdim=True) # Normalize rows
```

```
gcn_model = GCN(num_nodes, in_node_features, hidden_gcn_features,
num_classes)
output_gcn = gcn_model(dummy_node_features, dummy_adj_matrix)
print(f"GCN output shape: {output_gcn.shape}")
```

```
--- Conceptual Graph Convolutional Network (GCN) Scaffold ---
GCN output shape: torch.Size([1, 2])
```

Conceptual wav2vec 2.0 Inspired Scaffold

```
import torch
import torch.nn as nn
import torch.nn.functional as F

print("--- Conceptual wav2vec 2.0 Inspired Scaffold ---")

class FeatureEncoder(nn.Module):
    def __init__(self, in_channels, hidden_dims, kernel_sizes, strides):
        super().__init__()
        self.conv_layers = nn.ModuleList()
        for i in range(len(kernel_sizes)):
            self.conv_layers.append(
                nn.Conv1d(in_channels if i == 0 else hidden_dims[i-1],
hidden_dims[i],
                                  kernel_size=kernel_sizes[i],
stride=strides[i])
            )

    def forward(self, x):
        # x: (batch_size, in_channels, sequence_length)
        for conv in self.conv_layers:
            x = F.gelu(conv(x))
        return x # (batch_size, last_hidden_dim, reduced_seq_len)

class QuantizationModule(nn.Module):
    def __init__(self, embed_dim, num_codebooks, codebook_size):
        super().__init__()
        self.embed_dim = embed_dim
        self.num_codebooks = num_codebooks
        self.codebook_size = codebook_size
```

```
        self.codebooks = nn.Parameter(torch.randn(num_codebooks,
codebook_size, embed_dim))

    def forward(self, x): # (batch_size, seq_len, embed_dim)
        # Conceptual VQ: find closest codebook entry
        # For simplicity, returning input as is; real VQ uses argmin and
straight-through estimator
        return x

class ContextNetwork(nn.Module):
    def __init__(self, embed_dim, num_heads, num_layers, mlp_dim):
        super().__init__()
        self.transformer_blocks = nn.ModuleList([
            TransformerEncoderBlock(embed_dim, num_heads, mlp_dim) for _
in range(num_layers)
        ])

    def forward(self, x): # (batch_size, seq_len, embed_dim)
        for block in self.transformer_blocks:
            x = block(x)
        return x

class Wav2Vec2EEG(nn.Module):
    def __init__(self, num_classes=2):
        super().__init__()
        self.feature_encoder = FeatureEncoder(
            in_channels=1, # Assuming 1 channel raw EEG input at a time,
or concatenated channels
            hidden_dims=[512, 512, 512],
            kernel_sizes=[10, 3, 3],
            strides=[5, 2, 2]
        )
        # Calculate output dimension of feature encoder
        dummy_input = torch.randn(1, 1, 1000) # batch, channels, sequence
length
        dummy_output = self.feature_encoder(dummy_input)
        encoder_output_dim = dummy_output.shape[1]
        encoder_seq_len = dummy_output.shape[2]
```

```
        self.quantizer = QuantizationModule(embed_dim=encoder_output_dim,
num_codebooks=1, codebook_size=32)
        self.context_network = ContextNetwork(
            embed_dim=encoder_output_dim, num_heads=8, num_layers=6,
mlp_dim=encoder_output_dim * 4
        )
        self.classifier = nn.Linear(encoder_output_dim, num_classes)

    def forward(self, raw_eeg):
        # raw_eeg: (batch_size, 1, sequence_length) - treating as mono for
simplicity
        # In a real scenario, multi-channel processing would be more
complex.
        features = self.feature_encoder(raw_eeg)
        features = features.permute(0, 2, 1) # (batch, seq_len, embed_dim)
for Transformer

        quantized_features = self.quantizer(features) # Conceptual
quantization

        context_features = self.context_network(quantized_features)

        # For classification, typically use global average pooling or CLS
token
        pooled_features = torch.mean(context_features, dim=1)
        return self.classifier(pooled_features)

# Example usage with dummy raw EEG input
wav2vec_eeg_model = Wav2Vec2EEG()
dummy_raw_eeg = torch.randn(num_samples, 1, 16000) # batch, channels
(mono), 16000 samples
output_wav2vec = wav2vec_eeg_model(dummy_raw_eeg)
print(f"wav2vec 2.0 inspired model output shape: {output_wav2vec.shape}")
```

```
--- Conceptual wav2vec 2.0 Inspired Scaffold ---
```

EEG + VIDEO (2024UCA1952)

MLDCs

PAPER 1

MLDC Component	Description
Problem Definition	Automatically tag videos based on affective and attention-related EEG responses
Dataset	Collected EEG signals themselves from participants watching videos as part of the experiment.
Data Preprocessing	Noise filtering - remove power-line interference and muscle artifacts. Segmentation - divide EEG into manageable time windows. Artifact removal - eliminate eye-blinks or movement noise.
Feature Selection	The significance of frequencies and scalp regions is part of which features show stronger attention signals.
Feature Extraction	Analyzing different scalp locations - positions on the head. Studying frequency rhythms - electrical activity in different bands (alpha, beta, theta).
Machine Learning Model	Support Vector Machine (SVM)
Performance Metrics	Accuracy - 93.2% achieved using the SVM classifier.

CONCLUSION

Nowadays every event in the world appears to be captured in the form of images or videos. These multimedia contents are usually designed in a way such that they can hit the user’s cognizance effectively and this can be achieved through its affective content.

Thus, the modelling of human emotion and attentiveness can provide significant information about the effectiveness of the video content. In this paper, an attempt is made to find the association between a human's cognitive state measured using neurophysiological signals (EEG) and video content.

EEG band frequency range and commonly measured brain functions

Band	**Frequency Range**	**Significance**
Delta	0.5-3 Hz	Sleep quality measurement, Increased concentration when using internal working memory in tasks
Theta	3-8 Hz	Memory encoding and information retrieval from memoryCognitive workload indication, Fatigue level detection.
Alpha	8-12 Hz	Represents a relaxation state, sometimes monitoring attention level also
Beta	12-40 Hz	Brain activity involved in executive function "mirror neuron system" activation indication
Gamma	40-70 Hz	It reflects attentive focus. Can also record rapid eye movement

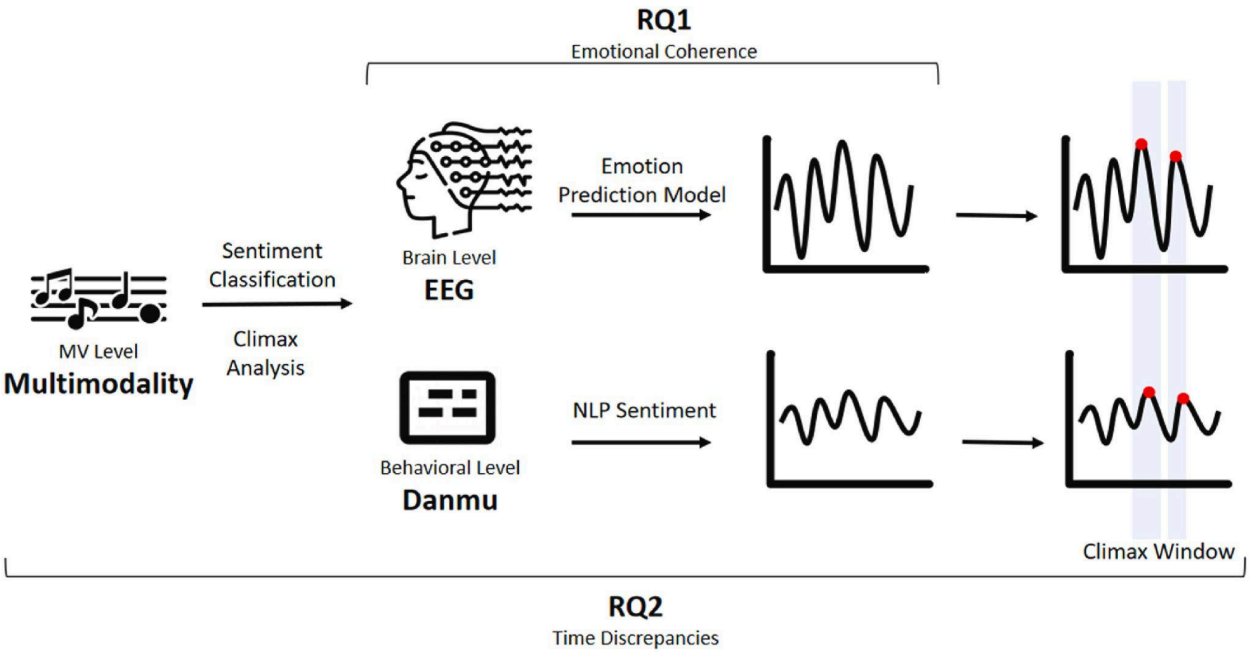
Research Gaps

The study uses a limited number of subjects, which may affect generalization.

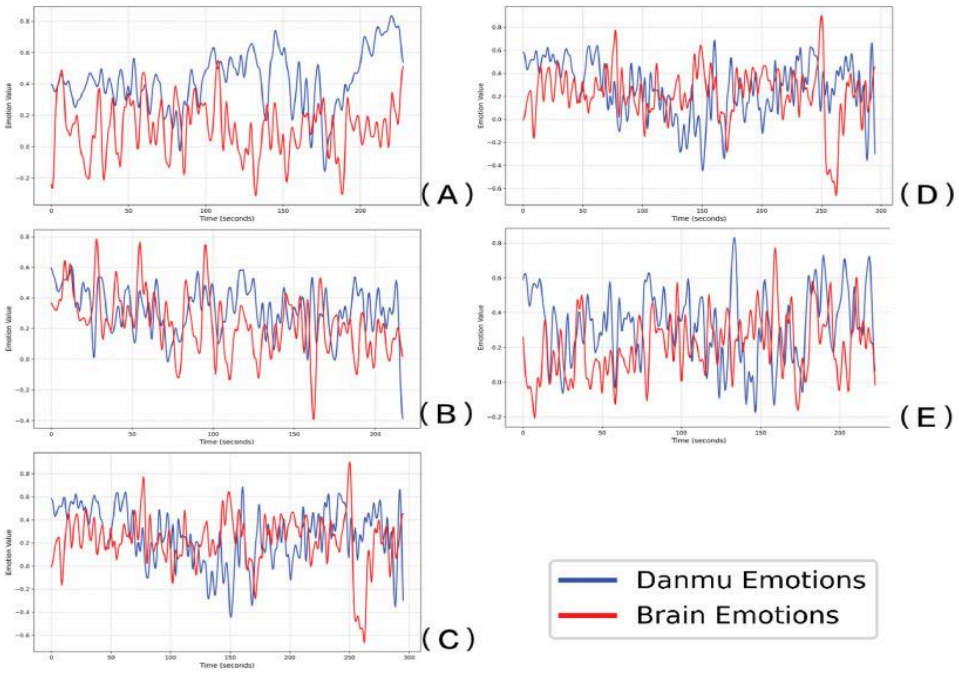
Only traditional machine learning (SVM) is used; deep learning models are not explored.

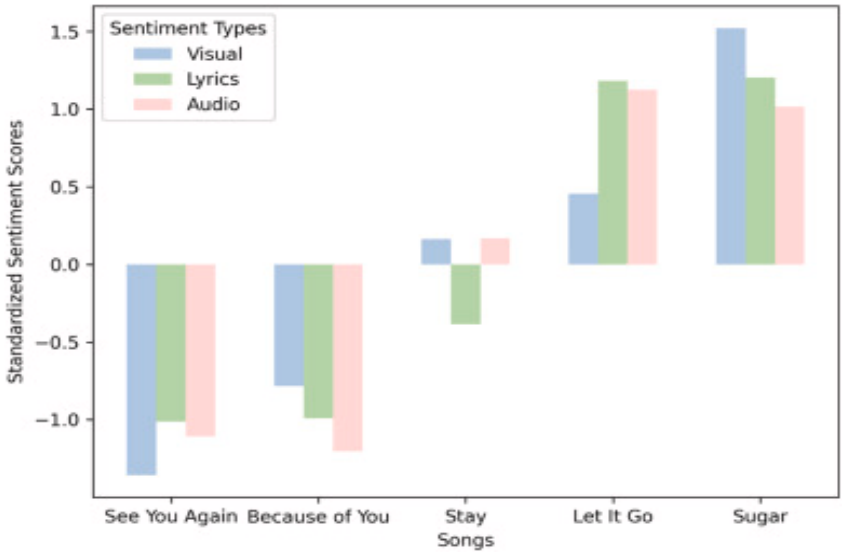
PAPER 2

MLDC Component	Description
Problem Definition	Analyze emotional coherence between EEG signals and danmu comments
Dataset	EEG recordings collected from participants watching music videos(the paper uses Five music videos). Danmu data (user-generated scrolling comments) corresponding to specific time points in the videos for sentiment analysis.
Data Preprocessing	Artifact removal (noise correction). Filtering into frequency bands relevant to emotion. Time-synchronization with video frames and danmu timestamps. Cleaning comments
Feature Selection	EEG features relevant to emotion recognition extracted via deep learning. Sentiment scores from danmu data as another feature set.
Feature Extraction	Deep learning model extracts continuous emotional scores from synchronized EEG data. Sentiment analysis of danmu comments produces sentiment curves synchronized with video time. The EEG and danmu features are aligned temporally to measure coherence (how closely the emotional curves match).
Machine Learning Model	CNN-LSTM (EEG), NLP sentiment model (danmu)
Performance Metrics	They report over 80% similarity between EEG-derived emotional curves and danmu sentiment curves across the five music videos.



Results showed a high similarity rate (above 80 %) between the real-time EEG and danmu curves across all five music videos. Videos with the highest similarity rates included ‘Let It Go’ (84.62 %), ‘Sugar’ (85.13 %), and ‘Stay’ (85.80 %). Conversely, ‘See You Again’ (83.09 %) and ‘Because of You’ (80.75 %) exhibited slightly lower similarity rates.





The models predicting emotional responses based on EEG data.

Models	Precision
SVM	87.41 %
Ensemble CNN	93.12 %
ResNet	94.95 %
LSTM	90.12 %
BiLSTM	84.21 %
SRU	83.13 %
The current model	97.67 %

Research Gaps

The dataset is small and limited to a few music videos, restricting scalability.

Cultural and language bias in danmu comments is not deeply analyzed.

PAPER 3

MLDC Component	Description
Problem Definition	Detect visual targets in low-quality video using EEG responses
Dataset	The authors use EEG datasets collected from subjects under a target-detection experiment.
Data Preprocessing	Artifact correction, phase-based EEG segmentation
Feature Selection	Feature alignment and fusion across EEG phases
Feature Extraction	Deep learning model extracts continuous emotional scores from synchronized EEG data. Sentiment analysis of danmu comments produces sentiment curves synchronized with video time. The EEG and danmu features are aligned temporally to measure coherence (how closely the emotional curves match).
Machine Learning Model	Brain-inspired deep learning model
Performance Metrics	They report over 80% similarity between EEG-derived emotional curves and danmu sentiment curves across the five music videos.

CONCLUSION

The paper proposes a novel brain-inspired phased temporal encoding and alignment fusion algorithm for EEG-based classification in low-quality video target detection. By analyzing the FRP and FRSP of EEG signals in both time and frequency domains, we demonstrate that the brain's response to target detection in low-quality videos can be roughly divided into three phases: early target recognition, later target spatial tracking, and global attention focus across the entire duration.

Research Gaps

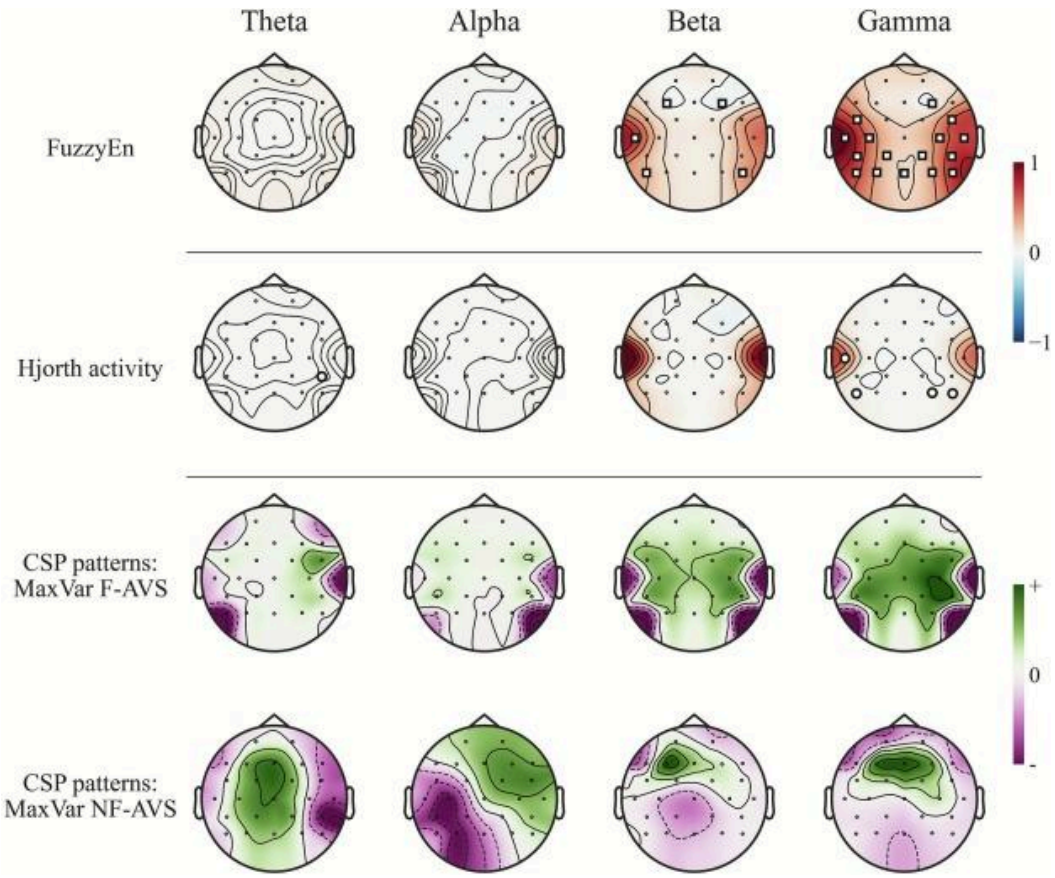
The deep learning model is computationally complex, limiting real-time deployment.Dataset availability is limited, reducing reproducibility.

PAPER 4

MLDC Component	Description
Problem Definition	Identify EEG features distinguishing familiar vs non-familiar videos. Cross-population analysis in healthy controls and patients with disorders of consciousness
Dataset	19 patients with Disorders of Consciousness (DOC). Personalized emotional videos. EEG recordings during video viewing
Data Preprocessing	Filtering, artifact removal, normalization
Feature Selection	Feature ranking based on discriminative power
Feature Extraction	Fuzzy entropy, CSP, Hjorth parameters
Machine Learning Model	Subject-independent ML classifiers
Performance Metrics	Significant performance in: 60% of MCS (Minimally Conscious State) patients, 33% of UWS (Unresponsive Wakefulness Syndrome) patients

Topographical brain activity during F-AVS and NF-AVS

The topographical maps of the difference values between conditions for FuzzyEn and Hjorth activity across frequency bands, along with the averaged CSP patterns that maximized each condition, can be seen in Fig. 4.



CONCLUSION

EEG feature comparison showed FuzzyEn, CSP, and Hjorth activity achieved the highest performance when combined with machine learning models to distinguish between electrical brain responses elicited by familiar and non-familiar audiovisual stimuli in healthy controls. Models trained on data from healthy controls and tested on data from patients with DOC exhibited performances that were generally aligned with the diagnosed level of consciousness of the patients. Healthy control models successfully distinguished between stimuli in up to 60% of patients in MCS, and 33% of patients with UWS, with performances that were statistically significant. The temporal, parietal, and frontal areas in the beta and gamma frequency bands appear to be relevant in classifying brain responses to personalized audiovisual stimuli in healthy controls.

Research Gaps

The number of DOC patients is small, limiting clinical generalization.

Long-term monitoring and repeated sessions are not evaluated.

PAPER 5

MLDC Component	Description
Problem Definition	Video-EEG for safety pharmacology seizure liability assessment in rabbits
Dataset	Experimental rabbit video-EEG recordings. Rabbits exposed to a compound labeled as pro-convulsive, monitored with EEG + video.
Data Preprocessing	EEG–video synchronization and epoch segmentation
Feature Selection	Spike frequency. Rhythmic patterns
Feature Extraction	EEG spike patterns and seizure-like events
Machine Learning Model	signal analysis–based approach
Performance Metrics	Seizure occurrence and behavioral correlation

Conclusion

This study demonstrates that synchronized video-EEG is an effective and reliable method for assessing seizure liability in preclinical safety pharmacology studies. By combining EEG recordings with continuous behavioral video monitoring, the researchers were able to detect both convulsive and non-convulsive seizure events that may not be identifiable through EEG or behavioral observation alone. The results show a clear relationship between EEG abnormalities and observable seizure-related behaviors following administration of a pro-convulsive compound. The video-EEG approach improved the accuracy and confidence of seizure detection across different dosage levels. Overall, the study concludes that integrated video-EEG monitoring provides a robust tool for early detection of seizure risks in drug development

Research Gaps

Conducted only on animal models, limiting direct human application.

Large-scale or long-term seizure prediction is not addressed.

LOGISTIC REGRESSION IMPLEMENTATION

```
import pandas as pd

df = pd.read_csv("features_raw.csv")

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

if 'Unnamed: 32' in df.columns:
    df = df.drop(columns=['Unnamed: 32'])

np.random.seed(42)
df['label'] = np.random.randint(0, 2, size=len(df))

X = df.drop(columns=['label'])
y = df['label']


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.5009299442033478

Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.54      0.52       803
           1       0.50      0.46      0.48       810

    accuracy                           0.50      1613
   macro avg       0.50      0.50      0.50      1613
weighted avg       0.50      0.50      0.50      1613
```

EEG + VIDEO (2024UCA1945)

MLDCs

Research Paper 1

Section	Component	Description
1.) Problem Statement	**Problem**	Current neuroscience research typically occurs in a constrained lab environment that lacks ecological validity.
	Goal	To explore if Inter-Subject Correlation (ISC) can be measured in loud, chaotic classrooms using inexpensive, hand-held devices (EEG).
	Objective	To employ neural engagement (synchronized brain activity) to track how students as a group focus attention on educational or narrative material.
2.) Data Collection	**Subjects**	4 distinct groups of participants (approx. 20-30 subjects total in various modalities).
	Hardware	Portable and low-cost Emotiv EPOC wireless EEG headsets.
	Stimuli	*Bang! You're Dead* by Alfred Hitchcock and *Sophie's Choice*. A scrambled version was used for control.
	Setting	Normal classroom setting (Joint viewing) vs. viewing through tablets.

3.) Data Preprocessing	**Noise**	EEG data is very noisy, especially outside a lab setting.
	Filtering	Band pass filtering (0.5 to 40 Hz) to remove slow waves and muscle noise.
	Artifact Removal	Correlated Component Analysis (CorrCA) applied to remove noise from movement, blinking, and talking.
	CorrCA Logic	Uses spatial filters to find maximum correlation between subjects; local, uncorrelated noises (e.g., scratching head) are automatically filtered out.
4.) EDA	**Statistical Analysis**	Permutation Testing (shuffling data over time) to establish a null distribution and check if brain synchrony occurred by chance.
	Visualization (Scalp Topographies)	Brain maps showing areas (occipital, parietal lobes) that were most synchronized.
	Visualization (Time-resolved ISC)	Line graphs showing synchrony increasing during exciting movie parts and decreasing during boring parts.
	Visualization (Violin Plots)	Used to display the distribution of correlation values across various groups.
5.) Feature Exploration & Selection	**Primary Feature**	Inter-Subject Correlation (ISC).

	Method	Correlated Component Analysis (CorrCA). Seeks components of the EEG signal most similar across all students (unlike PCA, which seeks max variance).
	Selection	Focus on the first 3 components, corresponding to how the brain reacts to visual and auditory narrative flow.
6.) Model Selection, Training & Evaluation	**Model Type**	Descriptive/Predictive Analytics model (not a Classifier).
	Technique	Linear Spatial Filtering (model is a set of weights applied to EEG electrodes).
	Evaluation	LOO (Leave-One-Out) to determine if one student's brain activities could be predicted by others in the group.
	Comparison	Compared results against the "Gold Standard" lab tests by Dmochowski et al. (2012) to evaluate the inexpensive equipment's performance.
7.) Model Deployment	**Current State**	Not implemented as a commercial application, but described as a step towards real-time inference of engagement.
	Setup	Multiple tablets synchronized through a local network to collect simultaneous EEG measurements from an entire class.

8.) Model Testing	**Ecological Validation**	Tested if the model works when students are seated together (Joint) versus alone (Individual).
	Narrative Validation	Evaluated if the model could separate a coherent narrative (High ISC) from a jumbled narrative (Low ISC). Lower synchrony in the jumbled video proved the model measured story engagement.
9.) Optimization	**Robustness**	CorrCA algorithm optimized to pull out a clean signal from an inherently "dirty" one, even with fewer electrodes and more noise.
	Computational Efficiency	Techniques used were potentially scalable to larger groups (10+ students) without requiring massive computing power.

Gaps in Research :

While the paper successfully moved EEG from the lab to the classroom, several holes remain in the research:

i.) Lack of Real-Time Feedback: Analysis here has been done offline; that is, after the class was over. There is no mechanism to tell a teacher in the moment that 40% of the class has lost focus.

ii.) Individual Differences (The Outlier Problem): ISC quantifies the degree to which a student is similar to the group. It doesn't account for students who are highly engaged but in a different way, such as a neurodivergent student or one with a different learning style.

iii.) Bi-directional Interaction Gap: This paper considers students only passively viewing a video; student-teacher interaction or active tasks such as coding, working on mathematics problems, and discussing within groups are not considered.

iv.) Sensitivity to Physical Environment: While it handled some noise, the paper mentions that high physical activity-walking and writing-creates too much muscle artifact (EMG) to be handled effectively by low-cost sensors.

My Suggested Improvements :

i.) Multimodal Fusion: EEG + CV

The Gap: EEG is noisy during movement.

The Fix: Use a webcam (Computer Vision) to track eye gaze and head pose. If a student's EEG synchrony drops and their eyes are away from the screen, the AI confirms disengagement. If the eyes are on the screen but EEG is low, the student might be confused rather than bored.

ii.) Generative AI Integration:

The Innovation: If the collective ISC drops below a threshold, the generative AI may suggest a Break Activity or automatically create a 1-minute summary/recap video of the last segment in order to bring students back up to speed.

Research Paper 2

Section	Description	Details
1.) Problem Statement	**Problem:** Long-term EEG visual inspection is time-consuming, labor-intensive, and prone to human error/bias.	**Objective:** Design a very accurate automated system to classify normal, interictal, and ictal states.
2.) Data Collection	Sources of multi-channel EEG recordings from pediatric or adult subjects with various seizure types.	CHB-MIT Scalp EEG Database or Bonn University Dataset.
3.) Data Preprocessing	Steps to clean and prepare continuous EEG signals for model training.	**Noise Removal:** Notch Filters (50/60Hz), Band-pass Filters (0.5–40 Hz). **Normalization:** Z-score or Min-Max scaling. **Segmentation:** Fixed-size windows (e.g., 1-second or 5-second).
4.) EDA (Statistical Analysis & Visualization)	Exploring data characteristics, especially amplitude and frequency shifts during seizures.	**Statistical Analysis:** Mean, variance, standard deviation. **Visualization:** Power Spectral Density (PSD) plots, electrode correlation heatmaps.
5.) Feature Exploration & Selection	Methods for extracting relevant patterns from the processed EEG data.	**Techniques:** Discrete Wavelet Transform (DWT), Empirical Mode Decomposition (EMD). **Selection:** Automated learning via CNN (spatial) and Recurrent layers (temporal) in deep models.
6.) Model Selection, Training & Evaluation	Choosing the appropriate architecture and assessing its performance.	**Model:** Hybrid architectures (CNN-LSTM, CNN-GRU). CNN extracts spatial patterns, LSTMs capture chronological sequence.

		Evaluation: Accuracy, Sensitivity, Specificity, F1-Score.
7.) Model Deployment	The final application context for the trained classification system.	Conceptualized for Clinical Decision Support Systems (CDSS) or cloud-based monitoring from wearable devices. **Model Testing:** K-fold Cross-Validation, Testing on Unseen Subjects (Leave-One-Subject-Out).
8.) Optimization	Fine-tuning parameters to maximize performance and efficiency.	Hyperparameter tuning (learning rates, dropout layers), optimizing window size for faster detection.

Gaps in Research

i.) Subject-Independence : Most models perform exceptionally well on a single patient (99% accuracy) but drop significantly when tested on a *new* patient whose brain signals look slightly different.

ii.) Latency vs. Accuracy Trade-off: High-accuracy deep learning models are often heavy. For a wearable device, we need models that are lightweight enough to detect a seizure in milliseconds without draining the battery.

iii.) Black-Box Nature: Deep learning models don't explain *why* they flagged a segment as a seizure. Neurologists need Explainable AI (XAI) to trust the machine's diagnosis.

iv.) Limited Real-world Testing: Most datasets are recorded in controlled hospital settings. Real-world noisy data (chewing, walking, talking) often leads to high False Discovery Rates.

My Suggested Improvements

i.) Hardware-Software Integration:

Integrate a False Alarm Button on the mobile app. When the AI makes a mistake, the user clicks the button; the model then uses Online Learning to adjust its weights so it doesn't repeat that specific error for that user.

ii.) Innovation - Attention-based Explainability:

Instead of a standard CNN-LSTM, use a Vision Transformer (ViT) approach for EEG. The Attention Maps can highlight exactly which EEG channels (electrodes) and which time points triggered the seizure alarm. This provides the Why for the doctor.

Research Paper 3

Section	Description
1.) Problem Statement	Goal is to design an automated, non-invasive diagnostic tool using EEG statistical patterns to distinguish among healthy people, TLE patients (interictal), and TLE patients (ictal).
2.) Data Collection	Used the standard Bonn University EEG Dataset with three classes: Set A (Healthy), Set D (TLE Interictal), and Set E (TLE Ictal).
3.) Data Preprocessing	Steps included Standardization (zero mean, unit variance), Filtering (to remove 50Hz and DC offset), and Windowing (dividing signals into smaller epochs).
4.) EDA (Statistical Analysis & Visualization)	Statistical analysis showed seizure activity significantly skews the signal distribution compared to the Gaussian-like healthy activity. Visualization likely used box plots and histograms.
5.) Feature Exploration & Selection	Used Key-Point Selection (focusing on minima, maxima, and mean values) to extract 5-6 primary statistical features (e.g., Standard Deviation, Skewness, Kurtosis) representing the wave shape.
6.) Model Selection, Training & Evaluation	Model used was Least Squares Support Vector Machine (LS-SVM) with a Radial Basis Function (RBF) kernel. Reported high performance (99% to 100% accuracy in binary classifications).
7.) Model Deployment	Presented as a Computer-Aided Diagnostic (CAD) tool for neurologists to quickly screen long-term EEG recordings for TLE markers.
8.) Model Testing	Ensured generalization using 10-fold cross-validation.
9.) Optimization	Optimized the LS-SVM using a Grid Search for parameters (gamma and sigma^2).

Research Gaps

i.) Spatial Localization Gap: The paper focuses on if a seizure is happening, but it provides limited information on where in the temporal lobe it originates, which is critical for surgical planning in TLE.

ii.) The Black Box of Statistical Features: While Kurtosis and Skewness are mathematically sound, they are hard for clinicians to interpret biologically. Doctors prefer features that relate to brain rhythms (Alpha, Beta, Gamma).

iii.) Dataset Diversity: The Bonn dataset is clean and recorded from a small number of patients. The model's performance in a real-world clinical setting with 100+ patients of different ages and genders is unknown.

iv.) Binary vs. Multi-class Performance: While the model is perfect at Healthy vs. Seizure, the accuracy usually drops when trying to distinguish Interictal (Seizure-free) from Healthy, which is the much harder clinical task.

My suggested Improvements :

i.) Innovation - Hybrid Feature Fusion:

Proposal: Combine the statistical structures (from this paper) with Power Spectral Density (PSD) and Log-Variance features.

Why: Statistical features catch the spiky nature of TLE, while PSD catches the slowing of brain waves. Combining them makes the model more robust to different TLE subtypes.

Research Paper 4

Section	Description
Problem Definition	**Problem:** Quantify the impact of 3D technology versus 2D on human brain's functional connectivity. **Goal:** Analyze and compare Functional Brain Networks (FBNs) across different frequency bands (Alpha, Beta, Theta, etc.) while subjects watch 2D vs. 3D video.
Data Collection	**Participants:** 40 healthy subjects. **Equipment:** High-density EEG recording. **Stimuli:** 2D and 3D video clips.
Data Preprocessing (Noise Removal)	**Filtering:** Band-pass filtering (Delta, Theta, Alpha, Beta, Gamma). **Artifact Correction:** ICA to remove EOG and EMG. **Re-referencing:** Average referencing of electrodes.
EDA	**Network Construction:** Using Phase Locking Value (PLV) or Correlation to create adjacency matrices. **Visualization:** 2D/3D brain maps showing edges. Comparison of Topological properties (degree distribution, clustering coefficients).
Feature Exploration & Selection	**Graph Theory Metrics:** Global efficiency, local efficiency, and average path length. **Frequency Band Analysis:** Identifying bands with significant connectivity changes during 3D viewing.
Model Selection, Training & Evaluation	**Approach:** Primarily Statistical Inference (ANOVA or T-tests) instead of a predictive classifier. **Evaluation:** Significant p-values indicating 3D increases global efficiency compared to 2D.

Model Deployment	Conceptually used to inform the design of Virtual Reality (VR) and Augmented Reality (AR) systems for optimizing user comfort and mental engagement.
Model Testing	Validation through multi-band analysis to ensure consistent findings across different brain activity states.
Optimization	Thresholding the brain networks (removing weak connections) to analyze only the most robust functional links.

Research Gaps

i.) Visual Fatigue Monitoring: The paper focuses on responses but doesn't deeply tackle the negative side effects of 3D, such as Vergence-Accommodation Conflict (eye strain) or motion sickness.

ii.) Content Specificity: The network changes might differ wildly between an action movie and a calm nature documentary. The study uses limited stimuli.

iii.) Temporal Dynamics: The analysis is often static (averaged over the whole video). It doesn't show how the brain network evolves as a story progresses or when a jump scare happens in 3D.

iv.) Lack of ML-based Prediction: While it uses graph theory, it doesn't build an AI model that can predict if a person is watching 2D or 3D based purely on their brainwaves.

My suggested Improvements :

i.) Innovation - Adaptive 3D Content:

Proposal: Instead of static 3D, build an AI agent that monitors the Global Efficiency of the user's brain network. If the brain network shows signs of high Visual Fatigue (specific Alpha-band patterns), the AI automatically reduces the 3D depth or parallax in the video in real-time.

ii.) Innovation - Edge-AI for Wearables:

Deploy a lightweight version of this analysis on a mobile device (using C++/TensorFlow Lite) that connects to portable EEG (like the Emotiv you've researched) to provide a Real-time Immersion Score for gamers or students in a virtual classroom.

Research Paper 5

Section	Topic	Description
1.) Problem Statement	Problem & Goal	**Problem:** Clinicians cannot manually review massive amounts of long-term EEG data. **Goal:** Create a reliable, automated system to detect seizure activity (ictal) with high sensitivity and low false alarms.
2.) Data Collection	Dataset Used	Bonn University EEG Dataset (Sets A, C/D, E used for comparison).
3.) Data Preprocessing	Steps	**Artifact Rejection:** Removal of eye blinks and muscle noise. **Normalization:** Scaling signal amplitudes to prevent magnitude bias in feature extraction.
4.) EDA	Analysis & Visualization	**Spectral Analysis:** Examining frequency domain power shifts. **Visualization:** PSD plots and signal morphology graphs (comparing spiky vs. smooth waves).
5.) Feature Exploration & Selection	Technique & Features	**Technique:** Autoregressive (AR) Modeling. **Features:** AR coefficients (weights) capturing the underlying signal rhythm. **Selection:** Burg's or Yule-Walker method.

6.) Model Selection, Training & Evaluation	Model & Metrics	**Model:** Support Vector Machine (SVM) with various kernels (Linear, Polynomial, RBF). **Evaluation:** Accuracy, Sensitivity, and Specificity.
7.) Model Deployment	Conceptual Application	Real-time bedside monitor for Intensive Care Units (ICUs) or Epilepsy Monitoring Units (EMUs).
8.) Model Testing	Validation Method	Cross-Validation to prevent overfitting of the AR-SVM combination to specific patients.
9.) Optimization	Tuning Parameters	Tuning the Order of the AR model and optimizing SVM hyper-parameters (C and sigma).

Research Gaps

i.) Non-Stationarity: EEG signals change over time (non-stationary). An AR model assumes a certain level of signal stability that doesn't always exist during a 24-hour recording where the user might be eating, sleeping, or talking.

ii.) Feature Complexity: AR coefficients are purely mathematical. They don't capture **Non-linear Dynamics** (like Entropy or Chaos theory), which are often better indicators of an impending seizure.

iii.) Computational Latency: While SVM is fast at inference, calculating high-order AR coefficients for 64+ channels simultaneously can be demanding for low-power wearable hardware.

iv.) Pre-ictal Detection: The paper focuses on *detecting* a seizure that has already started. The real clinical Holy Grail is predicting it (pre-ictal) minutes before it happens.

My Suggested Improvements :

i.) Innovation - Adaptive AR Modeling:

Proposal: Instead of a fixed-order AR model, use a Reinforcement Learning (RL) Agent to dynamically change the AR model order based on the signal's complexity.

Why: If the brain signal is calm, use a low-order model (save battery/compute). If the signal becomes chaotic, the agent increases the model order to capture more detail.

ii.) Integration with Mental Health (Depression):

Proposal: Use the same AR-SVM framework to analyze Alpha-band asymmetry (a common marker for depression). You can build an app that switches between Seizure Monitor Mode and Mood Tracking Mode by simply changing the frequency band being modeled by the AR coefficients.

LOGISTIC REGRESSION IMPLEMENTATION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score

df = pd.read_csv('EEG_data.csv')

X = df.drop(['user-definedlabeln', 'SubjectID', 'VideoID', 'Raw'], axis=1,
errors='ignore')
y = df['user-definedlabeln']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = LogisticRegression(solver='lbfgs', max_iter=5000)
model.fit(X_train, y_train)

predictions = model.predict(X_test)

print("LOGISTIC REGRESSION FOR CONFUSED STUDENT DETECTION")
print(f"ACCURACY SCORE : {accuracy_score(y_test, predictions) * 100:.3f}%")
print("\nPERFORMANCE METRIC :")
print(classification_report(y_test, predictions))
```

```
LOGISTIC REGRESSION FOR CONFUSED STUDENT DETECTION
ACCURACY SCORE : 59.813%

PERFORMANCE METRIC :
              precision    recall  f1-score   support

         0.0       0.60      0.61      0.61      1298
         1.0       0.59      0.59      0.59      1265

    accuracy                           0.60      2563
   macro avg       0.60      0.60      0.60      2563
weighted avg       0.60      0.60      0.60      2563
```