

Predicting video game Stimuli Using EEG Scans - CHALINI, FERNANDEZ, LEE

Brief Description

EEG (electroencephalographic) devices are used to measure the electrical activity of the brain. They work by placing sensors around the head of a person and these sensors measure the electrical changes produced by the brain. The voltage changes are caused by ionic currents within and between some brain cells called neurons. EEG shows the brain activity that is happening as participants absorb and process information and the sensors placed around their head measure the changes in current at each specific location where a sensor is placed.

EEG is currently most commonly used for performance and wellness of humans, consumer research, healthcare, seizure diagnosis, sleep study for sleep disorders, quantitative neuroscience and academic research. There has been previous research trying to identify emotional recognition in brain activity using EEG. For example, the Emotion Recognition Using Electroencephalography Signals of Older People for Reminiscence Therapy research focused on the feasibility of using EEG signals for automatic emotional recognition during reminiscence therapy for older people rather than using more conventional methods such as surveys and interviews. This study found two models that scored higher than 90% in accuracy when determining whether a user's experience was positive or negative. However, using EEG data to predict emotions has not yet been extensively researched.

Team Members

Ronald Lee, Aris Chalini, Jesus Fernandez

Problem Statement

Can we infer something about the video game a person is playing based on an EEG reading of their brain?

Objective

The objective of this project is to build a machine learning model that can predict whether someone is playing a boring, calm, horror, or funny video game, given an EEG reading. This will provide an understanding of what areas of the brain are related to specific emotions that humans are feeling. We believe this could help us understand the emotions of disabled children and adults who have trouble communicating. Some industries that might benefit from this research include the healthcare industry and the social work industry.

Our secondary objective is to be able to apply our best performing model to new EEG data such as the DREAMER dataset described below in the Datasets section.

Approach / Methodology

We trained and tested our models on the GAMEEMO dataset. This was done with a 60/20/20 train/validate/test split. We trained multiple models, some were trained on the raw dataset and other on the features-extracted dataset. The raw dataset contains the EEG brainwave signals collected from experiments which monitor brain waves while participants are playing video games that trigger different emotions. The features-extracted dataset was created by a feature extraction script that extracts features from the raw dataset. Thus the raw data is time-domain data and the features-extracted dataset is raw data transformed to frequency-domain data, with the addition of statistical features used for EEG analysis in neuroscience. Since we are working with EEG data, our intuition is that the frequency-domain data will build a more effective model. However, we decided to train some models with the time-domain data in order to (1) see if we could create a neural network which could rival

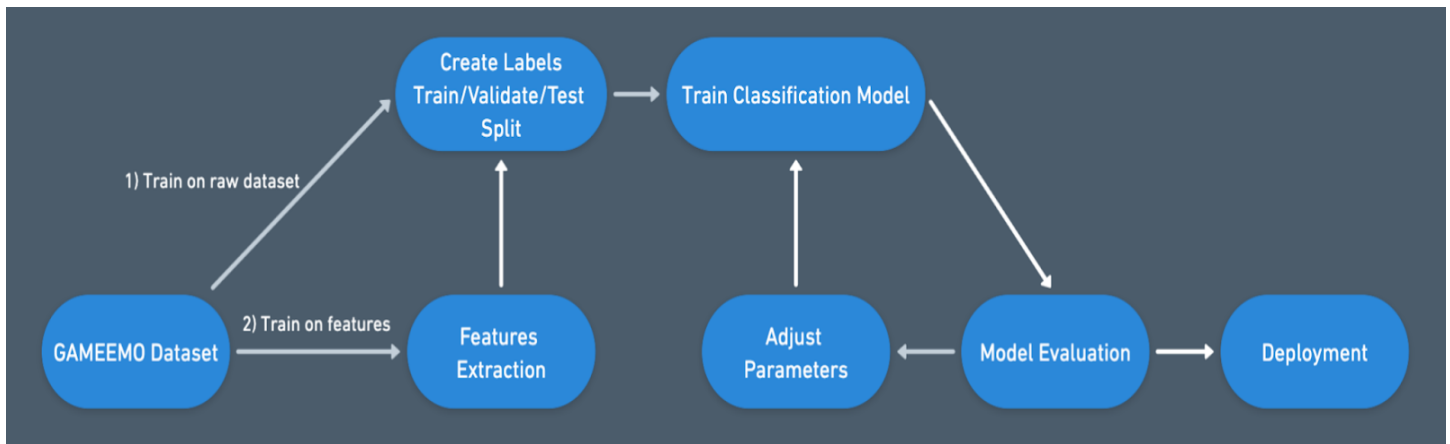
the complex feature engineering of the frequency-domain data and (2) see if any particular EEG sensor is significantly more useful in these predictions than the others.

We denoted the model which scored the highest accuracy on the test dataset as our best model. Then, we deployed it to make predictions on DREAMER - our secondary dataset. The DREAMER dataset contains EEG signals collected with the same device as the GAMEEMO dataset, however with slightly different labeling in regards to emotional state of the subject. The subjects in the DREAMER dataset were responding to a different stimuli; they were reacting to video clips instead of playing video games. Despite the differences in the two datasets, we thought it would be interesting to find out how transferable our emotional-state-predicting model would be to EEG datasets produced with other stimuli. We will use accuracy to assess this transferability.

Block Diagram

The model training pipeline consists of the following steps:

1. Extract features from the raw dataset to form the features dataset (frequency-domain data).
2. Label dataset. Split dataset in three. Training and validation datasets will be used for model creation. The test dataset will be used to assess the performance of the model.
3. Given initialized hyperparameters, train the classification model.
4. Evaluate the model with the validation dataset and examine success-statistics. If satisfied, skip to 6.
5. Adjust the hyperparameters and/or model structure.
6. Compare performance of all models created with steps 1-5. Choose the best model to deploy as a predictor of emotional-state for an alternative datasource: the DREAMER dataset.



Datasets

Datasource: GAMEEMO Study

Twenty-eight subjects had their EEG signals recorded while playing four video games from four distinct genres.

The GAMEEMO study's raw data was processed into the two datasets that we used for training our models. The raw data consists of EEG brain waves signals collected by an EEG device called the 14-channel EMOTIV EPOC+. This device has 14 sensors that are positioned to have contact at 14 locations on the user's head in order to record brain wave frequencies. This device also has a built-in sinc-filter which is used to remove higher frequency components of the recorded signals, components that likely result from physical movement.

Our sinc-filtered data consists of EEG readings for each subject/video-game combination (28 subjects and 4 different stimuli). The EEG reading yields 14 features corresponding to each of the 14 channels produced by the

Labels

GAMEEMO: Each entry in the dataset is labeled by the type of emotional response that was triggered.

Label	Conceptual Label	Video Game
1	Funny	Goat Simulator
2	Horror	Slender - The Arrival
3	Boring	Train Sim World
4	Calm	Unravel

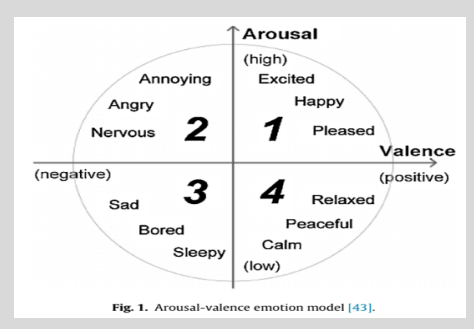


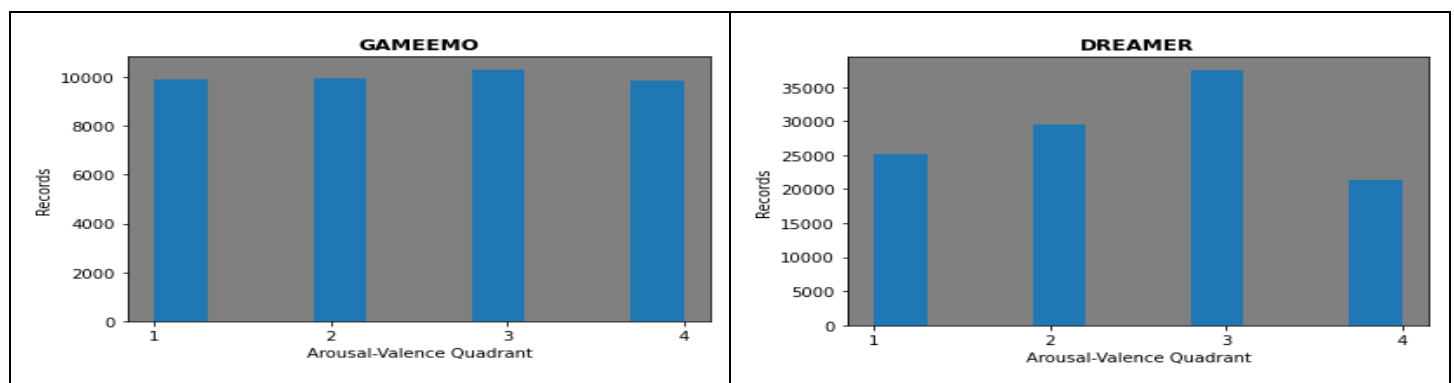
Fig. 1. Arousal-valence emotion model [43].

After playing each game, subjects were asked to rate the degree of funny, horror, boring, and calm experienced during the play session. These surveys confirmed that the selected games for the aural-visual stimuli were indeed good choices to represent the four quadrants of the Arousal-Valence emotional model.

DREAMER: Each entry in the dataset is labeled with a subject-reported value for Arousal, Valence, and Dominance. We used the mapping below in order to label this dataset with labels that matched those of the GAMEEMO dataset. These labels correspond to the same Arousal-Valence emotional model as GAMEEMO's.

Arousal	Valence	Dominance	Label	Arousal-Valence
3-5	3-5	ignored	1	HAPV
3-5	1-2	ignored	2	HANV
1-2	1-2	ignored	3	LANV
1-2	3-5	ignored	4	LAPV

Whereas the labeling for the GAMEEMO dataset is more uniformly distributed, the labeling for the DREAMER dataset has some variance that results naturally from our chosen label mappings.



Statistical Features Extraction

We used a features extraction script for brainwave data developed by Jordan J. Bird and was made public in his github repo (<https://github.com/jordan-bird/eeg-feature-generation>). The extraction techniques used in this script are backed by research in EEG signal analysis such that features extracted can be trained on models that

are not designed to handle time-series data. In this section we will explain the techniques used by this script to engineer the 3738 features included in our frequency-domain dataset.

Features are calculated using rolling sets of raw data that each represent 1 second (128 x 14). In addition, each row of this features-extracted dataset represents two sets of features. The first half (first set) of the features represent those calculated based on the previous 1 second of brainwave signal data and the second half (second set) is for the current 1 second of brainwave signal data. Here is a simple table to demonstrate the idea:

Row ID	Features 1 to 1869	Features 1870 to 3738
1	Features data for 1st second	Features data for 2nd second
2	Features data for 2nd second	Features data for 3rd second
3	Features data for 3rd second	Features data for 4th second
...

For each 1 second, the following features are computed:

Min/max/mean/std: For each second, the data is further broken down into half of a second and quarters of a second. For each 2 halves of a second (0.5 second each), it computes the signal data's min/max/mean/std. Same for the 4 quarters of a second (0.25 second each).

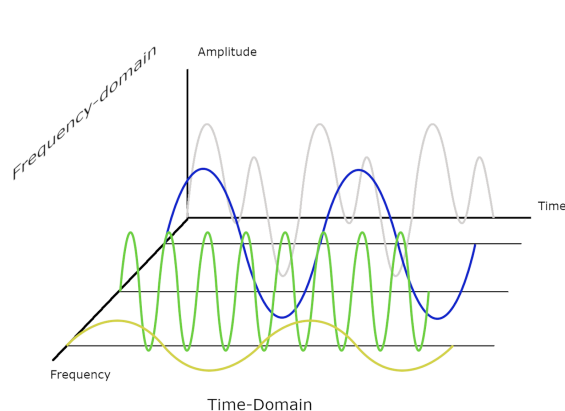
Skewness and Kurtosis: For each second of data, it computes the skewness (level of symmetry), and kurtosis (how heavily tailed).

Covariance: For each second of data, computes the covariance of the signal data. The covariance matrix is then flattened to become feature columns, only keeping the lower half of the matrix since the upper half is identical.

Eigenvalues: The eigenvalues are used for PCA for dimension reductions. Eigenvalues and vectors are used in comparing similarities.

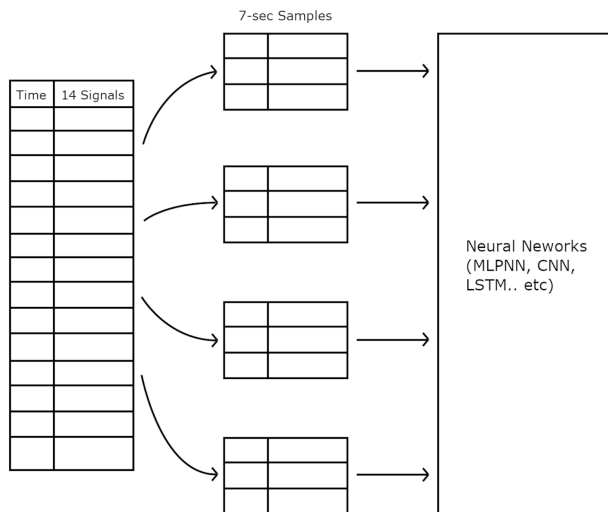
Logarithmic covariance: According to slides from the study linked below, this is useful for signals data prediction. <http://www.stat.rice.edu/~jrojo/4th-Lehmann/slides/Deng.pdf>.

Fast Fourier Transform: This transforms the signal data in time-domain to frequency-domain. The illustration below explains the difference between time and frequency domains.



Domain Change

In this illustration, the original signal (in gray), is broken up into 3 different sine waves, each having its own frequency and amplitude. The time-domain is viewing the signal data from the front, and the frequency domain is viewing the frequency data from the left.



7-second Sampling

For models that are trained on the raw data, we take the approach of splitting the raw data into samples of 7 seconds worth of signals data.

For CNN, we use Conv1D which handles data in a single temporal dimension. For LSTM, since it is designed to train on temporal data, this approach naturally fits as the input of the model.

It happens that all our data from GAMEEMO experiments have a fixed time duration, therefore the models don't need to handle variable length of data.

What is considered success / failure?

In order to have a standard on what we would consider a “successful” model we started by setting a conservative target for success criteria. Since there are 4 labels, a random guess will have a 25% chance of guessing it right. Therefore, our initial “success” criteria was an accuracy of 25%. We then increased this success criteria by creating and tuning a KNN model. KNN is one of the most simple classification models, thus any model that performs better will be considered a success and anything that performs worse will be considered a failure. However, in order to ensure that the KNN was a good base model we needed the KNN to perform better than the initial 25% accuracy of a random guess. After tuning the KNN model, we determined that a model with an accuracy greater than 72% would be deemed a success.

Evaluation Parameters

Our main evaluation parameter is the accuracy of the model when making classifications on the test data. Due to the nature of our data, there are no significant consequences caused by getting many false positives or false negatives. Therefore, we believe that using accuracy is the best metric because our main goal is to create a model that is able to predict correctly. During training, we also look at the categorical cross entropy loss on the validation and test dataset. Our metrics such as Precision/Recall/F1 scores are also collected.

Experiments

Hyperparameters tuning: We experimented with both manual and Keras hyper tuning. We methodically conducted several grid searches in order to both tune hyperparameter values and test alternate neural network structures when possible.

Adding layers: We experimented with adding new layers to a base neural network model and see if accuracy will improve. Besides simply adding additional Dense layers, we also tried increasing accuracy by adding transformative layers (such as convolution or normalization) or adding Dropout layers.

Datasets processing: One dataset was generated by a feature extraction script which extracts the statistical features of the raw data. Another dataset was generated by splitting the raw data into 7-second samples, so that each sample can capture more signal data. Which dataset can produce a more accurate predictive model?

Different sample size: For the 7-second samples dataset, we experimented with other sample sizes (other time intervals for each array to represent) and saw which performs the best.

Dimensionality Reduction: We tried using PCA and Random Forest to reduce the number of features and saw how the model performance could be affected. For the 7-second sampling dataset, we conducted a Random Forest in order to determine sensor importance. In other words, we determined which general region of the brain is more useful in predicting emotional state based on EEG signals.

Results

Hyperparameters tuning

Some hyperparameter tuning was done with grid search, but for the most part manual tuning was done. For most of the experiments with hyper tuning, we provide ranges of the possible values based on some manual tuning first, so that might limit hyper tuning performance. If we give a wider range of values, and more fine-grained increments for each value, hyper tuning might yield better results. However, that will require much longer time to run the tuning. For some models that take a long time to train like LSTM, such an approach will not be feasible for our project deadline.

Adding layers

For models that are trained on the features-extracted dataset and the 7-second samples dataset, adding layers help improve accuracy.

Datasets processing

On average, the models trained on the features-extracted dataset have a 11.71% accuracy improvement over models trained on the 7-second samples dataset.

Different sample size

For models that are trained on the raw dataset, we have tested on different sample sizes for MLPNN, CNN and LSTM models, while keeping all other hyper params constant. This table shows the results:

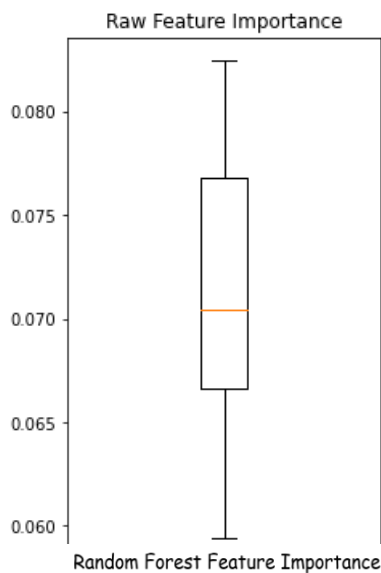
MLPNN	Accuracy	Loss	CNN	Accuracy	Loss	LSTM	Accuracy	Loss
3 secs	72.9%	5.47	3 secs	78.10%	3.18	3 secs	57.74%	1.28
7 secs	72.69%	5.14	7 secs	75.56%	7.54	7 secs	70.03%	1.09
14 secs	73.89%	6.38	14 secs	74.31%	15.38	14 secs	70.03%	1.09
42 secs	73.25%	10.45	42 secs	71.34%	41.38	42 secs	71.97%	1.14
147 secs	68.89%	54.97	147 secs	75.56%	386.28	147 secs	68.89%	3.65

It seems that different models can be impacted differently by the sample size. A sample size of 3 seconds does well for MLPNN and CNN, but not for LSTM. The Loss for MLPNN and CNN rise drastically as the sample size increases, where LSTM can maintain very low Loss over the board. The best model for the sampling approach is LSTM, which has both high accuracy and low loss. For the sampling size, it turns out 7 seconds works well for all models.

Dimensionality Reduction

In all cases, using PCA to reduce the number of features will also reduce the models' performance in Test accuracy.

Tests/Graphs/Discussions



Of the fourteen sensors, which is most useful in predicting emotional state?

Each of the EPOC's 14 sensors records EEG frequencies from a different region of the brain. Using our best performing Random Forest model, we sorted these fourteen features by importance in predicting our label. We wanted to see if we could pinpoint a region of the brain that is most associated with emotional response. Results are shown to the left.

There is little variance (0.000047) in the distribution of sensor importance. No specific sensor is notably more important in predicting emotional state than the others. In addition, no specific region of sensors appears to contribute more to the model than any other region. However, it should at least be noted that all of the T and F sensors appear in the top 5 most important sensors, which from most important to least are F3, T7, FC6, T8, and F4.

Does feature reduction aid in model prediction accuracy?

When we did the same thing using our dataset with the engineered features, we saw that many of these features have a negligible contribution to the model. The values are extremely skewed, in fact the mean is larger than the third quartile. Our machine is using processing power on computations related to many features that contribute little to the accuracy of our predictions. This suggests that it may be a strategic decision to reduce the number of features in the dataset for the sake of hypertuning efficiency.

A reduced version of this dataset was created which contained only features with values above 0.001, which approximately represents the top six percent of the original features. Using this dataset significantly reduced training time - at the expense of some record complexity. The low training time allowed for the much more thorough hypertuning using a methodical Grid Search approach. However, despite the efficient hypertuning, we were unable to create a model using the feature-reduced-dataset that could outperform the best model we created with the feature-full dataset (3738 features).

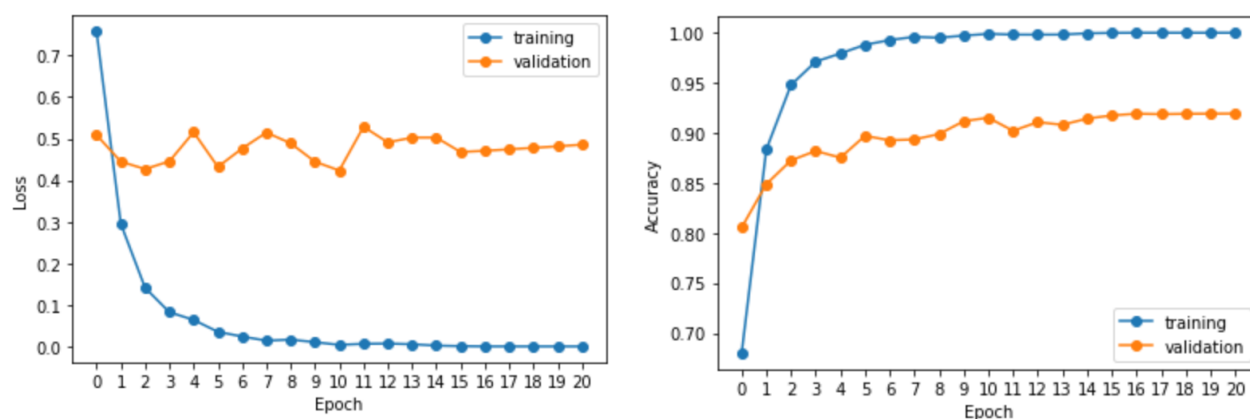
Which engineered features are the most useful for predicting emotional state and what do they represent?

Among the top 6% of most important engineered features, 73% of these features include the tag "covM" in their description. This is significant given that features that include the tag "covM" make up only 11% of all engineered features in this dataset. This is not surprising since those data points are covariance of signal data within the 1 second period.

What did the training of our best model look like?

Our best performing model was the MLPNN model trained on features-extracted dataset. It has 2 dense layers, with 512 and 64 units respectively. The model was trained with a learning rate of 0.001sec and a batch size of

30, using Adam optimizer and relu activation. The model's performance started to peak after about twenty epochs. Training progress is shown in the two graphs below.



Could models built on raw data ever outperform models built on data with engineered features?

In this experiment we found that our best model was trained on the data with engineered features and had an accuracy of 92%. This model was a MLPNN and with experimentation, we found that it performed best with just two hidden layers. We believe that the sheer volume of features (whose relevance to EEG readings is backed by neuroscience) available in this dataset is the reason further network complexity was unnecessary in order to train an effective model. It is likely also the reason that we were never able to achieve comparable accuracy (77%) with a model trained on the raw data.

In theory, since engineered features are just mathematical transformations of raw EEG signals, it should be possible for a neural network to be sophisticated enough such that raw EEG signals could be used to make comparable predictions. Although we were unable to achieve this, we do note that in general these models did improve as we added complexity by adding neurons or different types of layers. We also note that we could spend more time tuning raw data array-size, as there is a constant trade off between information in each array and amount of data to use for training.

Constraints

The biggest constraint we were faced with was computational performance, especially when tuning hyperparameters on the feature-extracted dataset. Due to the large number of features in this dataset, our tuning code would take hours to finish running. This lack of computational power is also the reason we chose to use Grid search instead of Random search (possibly more effective, but too computationally intensive in this case). But since even Grid search was a lengthy process, we would sometimes default to manual tuning. Our ability to choose optimal hyperparameters for our model was limited and therefore it is possible our model's performance could have been better than it was.

A different solution could have been to increase our computational power by purchasing a high performance computer or by utilizing a cloud environment such as GCP or AWS.

Standards

Tensorflow version 2.4.1
Keras 2.4.0

Comparison

The best model in terms of test accuracy and loss is MLPNN trained on the features-extracted dataset, with 2 Dense layers (512, 64 units), Adam optimizer and “relu” activation. It attains 92% accuracy and 0.50 loss. The second best model is LSTM+CNN+MLPNN trained on the features-extracted dataset, with 89.63% accuracy and 0.40 loss, but takes 40 times longer to train. One possible reason that a simple MLPNN model performs better than a more complex LSTM+CNN+MLPNN model is that the features-extracted dataset has done all the heavy lifting preparing the dataset, so that a simple 2 Dense layers model is enough to perform well.

The tables below show the comparisons between models. All models except KNN and Random Forest are configured to stop early when there are 10 epochs having increased loss during the training.

Models trained on features-extracted dataset

Model Name	Accuracy	Loss	Training Time	Hyper params (not a complete list)
KNN	72.00%		0.00755 sec	1 neighbor
MLPNN	92.65%	0.50	703 sec	2 Dense layers (512,64 units), Adam optimizer, relu activation.
MLPNN (PCA)	76.33%	1.33	83 sec	Same as MLPNN, PCA 95% of variance retained.
CNN	79.86%	0.87	179 sec	3 Conv layers (50,50,50 filters), kernel 3, stride 1, SGD optimizer
CNN (PCA)	67.96%	2.22	120 sec	Same as CNN, PCA 95% of variance retained.
CNN + MLPNN	88.82%	0.44	344 sec	3 Conv layers (100,100,100 filters), kernel 3, stride 1, 1 Dense layer (512 units), SGD optimizer, relu activation.
CNN + MLPNN (PCA)	80.51%	0.83	162 sec	Same as CNN+MLPNN, PCA 95% of variance retained.
CNN + LSTM + MLPNN	89.56%	0.41	1187 sec	3 Conv layers (100,100,100 filters), kernel 3, stride 1, 1 LSTM layer (256 units), 1 Dense layer (512 units), SGD optimizer, relu activation.
CNN+LSTM+MLPNN (PCA)	79.98%	0.85	399 sec	Same as CNN+LSTM+MLPNN, PCA 95% of variance retained.
LSTM	78.23%	0.79	1539 sec	1 LSTM layer (15 units), Adam optimizer.
LSTM (PCA)	69.93%	1.27	556 sec	Same as LSTM, PCA 95% of variance retained.
LSTM + MLPNN	85.88%	0.80	1648 sec	1 LSTM layer (15 units), 3 Dense layers (100,100,100 units), Adam optimizer, relu activation.
LSTM+MLPNN (PCA)	83.96%	0.82	588 sec	Same as LSTM+MLPNN, PCA 95% of variance retained.
LSTM + CNN + MLPNN	89.63%	0.40	3351 sec	1 LSTM layer (256 units), 3 Conv layers (100, 100, 100 units), kernel 3, stride 1, 1 Dense layer (512 units), SGD optimizer, relu activation.
LSTM + CNN + MLPNN (PCA)	79.53%	0.87	1230 sec	Same as LSTM+CNN+MLPNN, PCA 95% of variance retained.
Random Forest	81.00%		15.9 sec	min_samples_leaf=20, n_estimators=1000, max_depth = 150, bootstrap=True, oob_score=True, n_jobs=-1, random_state=seed, max_features=0.2

Models trained on raw dataset with 7-second samples

Model Name	Accuracy	Loss	Training Time	Hyper params (not a complete list)
MLPNN	72.26%	1.065	36 sec	2 Dense layers (512,256 units), SGD optimizer, relu activation.
CNN	75.56%	7.54	45 sec	1 Conv layer (1024 filters), kernel 3, stride 1, Adam optimizer, relu activation.
CNN + MLPNN	75.98%	1.32	26 sec	4 Conv layers (100,100,100,100 filters), kernel 3, stride 1, 1 Dense layer (512 units), SGD optimizer, relu activation.
CNN + LSTM + MLPNN	76.62%	1.05	52 sec	3 Conv layers (100,100,100 filters), kernel 3, stride 1, 1 LSTM layer (256 units), 1 Dense layer (512 units), SGD optimizer, relu activation.
LSTM	70.03%	1.09	64 sec	1 LSTM layer (15 units), Adam optimizer
LSTM + MLPNN	75.56%	0.92	112 sec	1 LSTM layer (256 units), 1 Dense layers (512 units), SGD optimizer, relu activation.
LSTM + CNN + MLPNN	75.35%	1.26	110 sec	1 LSTM layer (256 units), 3 Conv layers (100, 100, 100 units), kernel 3, stride 1, 1 Dense layer (512 units), SGD optimizer, relu activation.

Limitations of the Study

Although we were able to create an effective model for predicting emotional state with GAMEEMO data, deployment on the DREAMER data demonstrated that the transferability of our model might be more limited than what we had assumed (only 30% accuracy). Below we note some possible flaws in our study's design:

- **Different Stimuli:** Although both datasets were composed of the same 14 EEG sensor frequencies, the subjects of the two studies had their emotions triggered in different ways. The GAMEEMO subjects were stimulated with video games whereas the DREAMER subjects were stimulated with video footage. Perhaps the physical motions associated with playing a video game produced EEG patterns that added noise to those EEG patterns related to emotional state.
- **Activity:** Video games are more interactive than watching a video and therefore requires the users to be more involved and perhaps to think more which could intensify the brainwave lengths recorded by the EEG device.
- **Limitations of a Four-Quadrant Model:** Within the arousal-valence emotional model, each quadrant contains several related, but distinct emotions. Perhaps some of the emotions being stimulated are close to the x or y axis. Theoretically, these emotions would be more difficult to categorize into our four quadrant model because they are near the border between categorical labels.
- **Emotional State Labels:** The mapping of labels in order to utilize the same emotional model (Arousal-Valence) ignored the "Dominance" attribute recorded for each subject in the DREAMER study. Because of this, there is data related to emotional state that was not considered when we created the labels for our DREAMER dataset. Therefore, it is possible that deployment is failing because the data pre-processing in each of the experiments was done using different methods and the built in filters on the EEG device could have been setup differently during the experiment.
- **Subject Diversity:** The subjects that were tested for the GAMEEMO study were all university students in the software engineering department. Therefore, it is not likely the data collected from these subjects does a good job at reflecting on the population. Lastly, it was not taken into account whether each subject finds joy in the different genres. For example, a subject that enjoys horror might have a different reaction to the horror video game than a subject that dislikes horror.

Alternatively, we could speculate that perhaps the way that the brains of different people respond differently to alternate complex stimuli is just too variable in order to train a dependable universal model - one which could be deployed effectively on a diverse set of EEG emotion data.

Future Work

Since our data was generated by a small group of people in a very narrow age range (20-27), we believe that collecting more EEG data from a wider group of people is necessary to be able to create a model that can adapt to new, unseen data. Furthermore, it is important to also increase the number of activities that users are performing while the EEG data is recorded so that the model can learn to ignore EEG signals related to movement or complex thinking instead of those related to emotion. Our unsuccessful deployment further demonstrates the necessity of this as it appears our model is picking up on patterns relating to how the game is being played instead of how the subject feels about playing the game. Lastly, it could be helpful to indicate whether a subject is interested in the genre or activity for which the data is being collected in order to account for how stimulated the subject will be during that particular activity.

At the very least, this study shows that across datasets created by the same stimuli, classification is possible. This could eventually have application in realms where we need to know how a specific stimulus is impacting a person who cannot express their emotions. However, it looks like at the moment, as it pertains to EEG data, a different model would need to be created for each type of emotional-stimuli. Therefore, it is not currently possible to build a universal emotional-state-classifier off of EEG data.

References

GAMEEMO: <https://www.sciencedirect.com/science/article/abs/pii/S1746809420301075?via%3Dihub>

DREAMER:

S. Katsigiannis, N. Ramzan, "DREAMER: A Database for Emotion Recognition Through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices," IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 1, pp. 98-107, Jan. 2018. doi: 10.1109/JBHI.2017.2688239

Previous research:

<https://www.frontiersin.org/articles/10.3389/fphys.2021.823013/full#:~:text=Results%3A%20The%20EEG%20data%20of,%2D LSTM%20 also%20 reached%2094.4%25>.

Logarithmic covariance: <http://www.stat.rice.edu/~jrojo/4th-Lehmann/slides/Deng.pdf>

Feature engineering code: <https://github.com/jordan-bird/eeg-feature-generation>