# fMRI Brain Activity and Mood: A 3D-CNN based Decoder for BMI Applications

Akwasi D. Akwaboah aakwabo1@jhu.edu
Kevin Tran ktran35@jhu.edu
Feng-Chiao Lee flee38@jhu.edu

## Abstract

*In rehabilitative neuroscience, Motor Brain-Machine Interfaces (BMIs) have been extensively studied. They have shown promise at restoring lost mechanical body functions such as limb control and posture. Mood BMIs, on the other hand, are rather nascent technologies with implications for treating neuropsychiatric disorders such as depression and seizures. In this study, we implement a 3D convolutional neural network (CNN) based decoder trained from scratch for discriminating four music-induced valence-arousal mood states expressed in fMRI data (open source). We performed fMRI volume annotation to facilitate supervised learning for the decoder amidst significant "curse of dimensionality". Also, we explored support vector machine and k-Nearest Neighbor classifiers on the same dataset. We subsequently performed inference with the trained models and obtained reasonable recognition with the kNN and 3D-CNN models.*

## 1. Introduction

Brain-Machine interfaces (BMIs) generally refer to engineered constructs that facilitate a bidirectional link between the brain and either the state of an artificial actuator (prosthesis) or a desired affective/ emotional state. By this definition, BMIs are grouped into two categories – motor and mood. While mood BMIs have been disproportionately researched over the decades, there has recently been a growing interest to commensurately explore such interventional technologies for neuropsychological research and therapy [1] [2]. Mood BMIs typically comprise of a controller module and a plant (the brain) [1] as shown in Figure 1 During closed-loop neuroscience experiments and therapy, the brain state/ mood encoded in measured cerebral activity such as electroencephalogram (EEG), electrocorticogram (ECoG), functional Magnetic Resonance Imaging (fMRI), etc is measured and rectified. The controller comprises of a decoder that interprets the recorded neural activity into an affective state (mood) and feedback driver that determines the appropriate stimuli to drive the brain into a desired state.

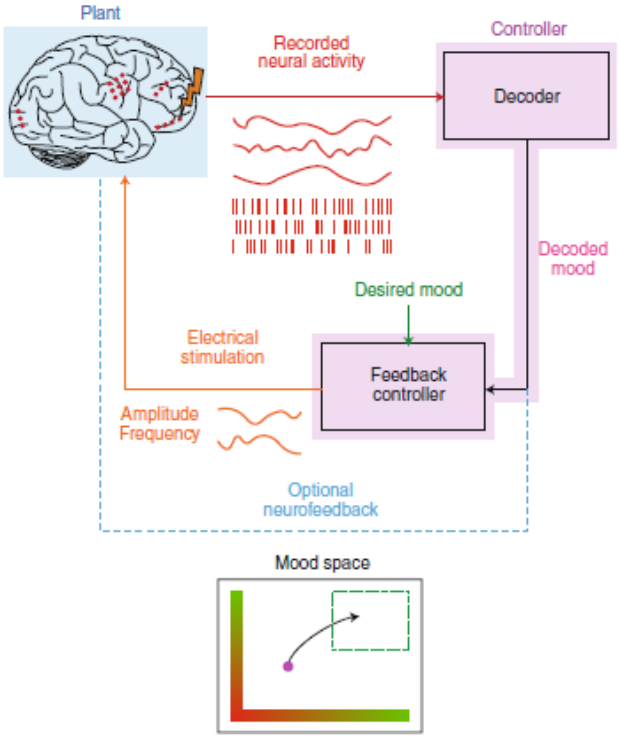


Figure 1. Mood BMI Illustration. [1]

Until recent times, the decoder block was implemented in the form of mathematical models such as generalized linear models (GLMs) and other simplistic non-linear regressors. The recent repopularization of neural networks (this time with the name deep learning) have shown promise at recognition in computer vision tasks with domain-specific application such as medical image segmentation and classification. By leveraging the prowess of deep learning (DL), decoding neural activity can be done efficiently. In this light, we proposed a DL-based (specifically a 3D-convolutional neural network (CNN)) decoder for discriminating music induced mood state expressed within the joint-fMRI data recently published by Daly et al. [3]. We initially proposed to include a backward step of synthesizing EEG signals from mood state using adversarial methods, but we eventually discontinued that aspect due to the exten-

sive work the decoder formulation required in such limited time.

## 2. Related Work

Convolutional neural networks are presently commonplace models that have been extensively used and validated in computer vision recognition tasks after their outstanding performance in the ImageNet LSVRC competition by Krizhevsky et al.[4] in 2012 and other subsequent variants/ advances [5][6]. Most of CNN applications are on 2D images, reasonably so because of the manner in which real life images are acquired. However, in domains such as medical imaging, volumetric representations of anatomy and physiology are often preferred as they hold more holistic biological information. A recent and closely related work by Vu et al. [7] present a 3D-CNN approach for classifying fMRI volumes obtained while subjects were performing four distinct sensorimotor tasks – left hand clenching, right hand clenching, auditory attention and visual stimulation, which are quite different from the contiguous tasks of musical stimulation and concurrent reporting of affective state spanning the four quadrants of the valence-arousal circumplex for the data used in this study. This notwithstanding, we took inspiration from adopting a 3D-CNN approach in implementing our mood decoder. It must be noteworthy that Vu et al. [7] performed classification on a reduced number of fMRI volumes (1,440) compared to ours (described in the "Methods" section) with and without preprocessing steps such as slice timing correction, motion correction, spatial normalization and spatial smoothing. For fMRI volumes with no preprocessing, a mean error rate of 26.2% for the 3D-CNN and substantially higher error rates for the other classifier explored – SVM - 75.0% and 1D-fcDNN - 75.0%.

## 3. Methods

### 3.1. Dataset and Annotation

We used the joint EEG-fMRI dataset recording during affective music listening as published by [3]. The dataset features simultaneous recordings of a subject's EEG, fMRI, and manual reporting through the use of a FEELTRACE device. The FEELTRACE is a device developed by Roody Cowie[8] that uses a joystick to record the subject's reported feeling(s) during a trial in a two dimensional plane. Each trial consisted of the following steps: 1). subject was presented with a fixation cross on a digital screen for at least 1 second, 2). stimuli music was played for 40 seconds while the subject listened, listened and reported, or only reported, 3). 0.5 second break before the start of the next trial. The reporting-only trials involved presenting the subject with their previously reported feelings from the FEELTRACE and having them recreate what they see without listening to music. This step was conducted in order to isolate the contribution of the reporting method to the EEG-fMRI data from the contribution caused by the music and emotional states. The testing protocol for a single trial is summarized in Figure 3 Sample fMRI volume slices expressing Blood Oxygen Level Dependent (BOLD) signals in the sagittal, coronal and axial planes have been shown in Figure 3.

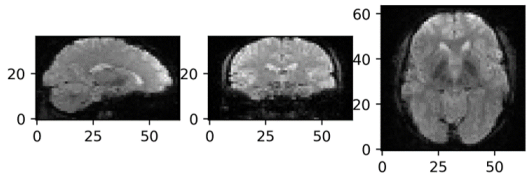


Figure 2. Sample data: fMRI volume slices in the sagittal, coronal and axial planes (left to right)

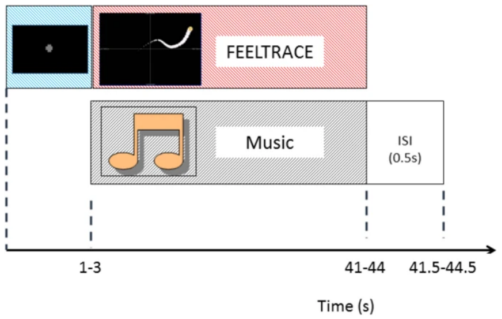


Figure 3. Timing of events within a single trial [3]

#### 3.1.1 Mood Space and FEELTRACE System

Mood space offers a simplified representation of a subject's mood. The mood space is constructed by two axes, the arousal axis, and the valence axis. The recorded arousal goes from very negative to very positive, while the recorded valence goes from very negative to very positive either. The mood space can basically be divided into four quadrants. Four quadrants encode distinct groups of emotions as shown in Figure 4. The first quadrant Positive Arousal- Positive Valence, shows the mood Delighted. The second quadrant Negative Arousal- Positive Valence, generally represents a relaxed moods. Third quadrant Negative Arousal- Negative Valence generally represents depressed moods. Fourth quadrant Positive Arousal- Negative Valence generally captures moods such as anger. The mood space allows the researcher to define the tester's current mood for future intervention where necessary. FEELTRACE is a recording system. FEELTRACE recorded the values of arousal and valence are non-negative ranging from one to nine. The minimum value means the very negative effect, and the maximum value represents the very positive. The researcher usu-

ally sets an offset point in the FEELTRACE axis helps the researcher transfer their FEELTRACE data into the mood space.

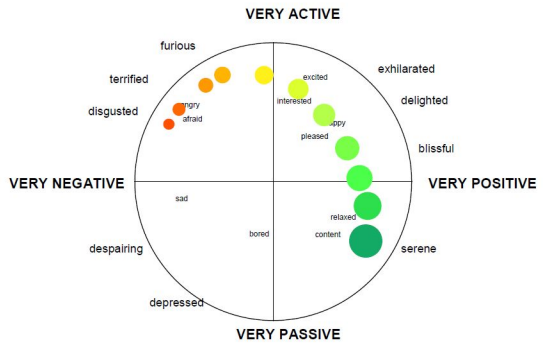


Figure 4. Mood Space Illustration [8]

### 3.1.2 Data Preprocessing

We used the EDF file provided in the dataset to extract the labels for each fMRI volume. The EDF file includes the EEG signal, FEELTRACE data, and the trial type information. The FEELTRACE data together with the trial information were used in extracted relevant trail sections in the data. In the EDF file, ft–arousal and ft–valence records the FEELTRACE data over time for the music only, music and reporting, and reporting only trials. Example EDF data from 'sub-01-Task-genMusic02-eeg.edf' is illustrated in Figure 5 .

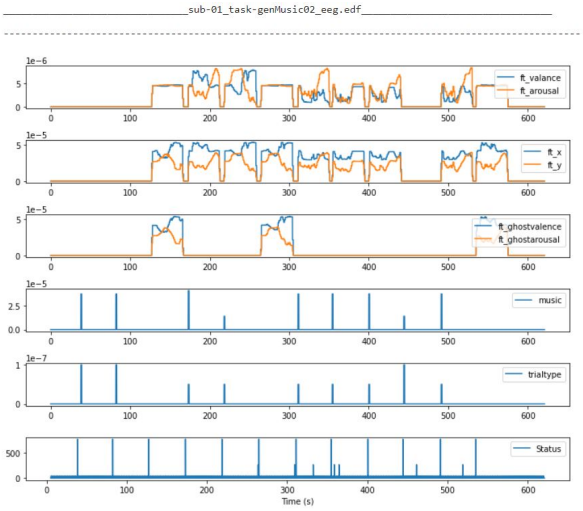


Figure 5. EDF Data Exploration

For this project, we extracted the music and reporting section of the data so that we can have the fMRI image response with music stimulation. We generated fMRI volume annotations based on the valence-arousal time instance within the four-quadrant in the mood space to classify the fMRI image. First, we extracted the music-and-reporting section in FEELTRACE data and used the reporting-only section to calculate the average FEELTRACE value as the offset value. The reporting-only section is captured appropriately in the ft–ghostvalence and ft–ghostarousal traces, which was obtained by subtracting the reporting-only trials from the FEELTRCE x-y traces. The music and reporting trials were then scaled appropriately and shown in Figure 6.

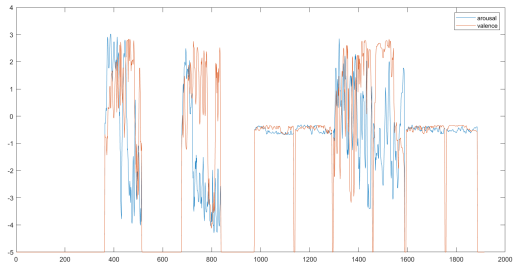


Figure 6. Music and Report Data Extract

The mood labels {0, 1, 2, 3} relate to the four-quadrants in mood space. The mood label 0 represents the Positive Arousal–Positive Valence quadrant. Mood Label 1 represents the Positive Arousal–Negative Valence quadrant. Mood Label 2 represents Negative Arousal–Negative Valence quadrant, and Mood Label 3 represents Negative Arousal–Positive Valence quadrant. The FEELTRACE data transferred to the mood space is illustrated in Figure 7.

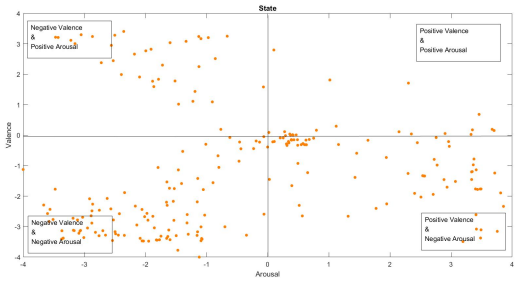


Figure 7. Transferred FEELTRACE data in Mood Space

### 3.1.3 fMRI Volume Extraction

Once the mood labels were extracted as a time series from the FEELTRACE data, we extracted the fMRI volumes that correspond to the exact time points for each label. Each '.nii.gz' file was first loaded and scanned to determine the

number of time slices in each set of fMRI volumes. The number of slices for the classical music trials were around 930 slices for each subject which aligned with the expected total time of these experiments. The number of slices for each 'genMusic' experiment was around 280 slices. While performing this inspection, it was discovered that the file named 'sub-01_task-genMusic01_bold.nii.gz' was corrupted in the dataset and was unusable for these experiments.

The repetition time (TR) of the fMRI was 2 seconds which corresponds to a sampling frequency of 0.5 Hz. Due to this, the fMRI volumes were sampled at a rate lower than that of the FEELTRACE data which means there are fewer fMRI volumes than number of FEELTRACE data points. Any boundary fMRI volumes (volumes that overlap with a change in FEELTRACE mood label) were assigned to the previous FEELTRACE label. Each slice index corresponding to a FEELTRACE label was stored into a comma-separated value (CSV) file for reference use in the experiments. The CSV file contained the file name, slice index, and class label which allowed us to easily access the data for use. In total, 11,107 fMRI slices were extracted from the available '.nii.gz' files from the 21 subjects.

#### 3.1.4 Data Balancing

The 11,107 fMRI volumes extracted from the mood labels had an unbalanced class distribution which is known to skew the training and testing results. To deal with this, we randomly extracted a total of 6,000 volumes from the 11,107 volumes, with each mood label subset containing 1,500 volumes (the minimum class count) for training. The raw unbalanced dataset and balanced version after random undersampling (using the *imblearn.under_sampling.RandomUnderSampler* module) are presented in Figure 8

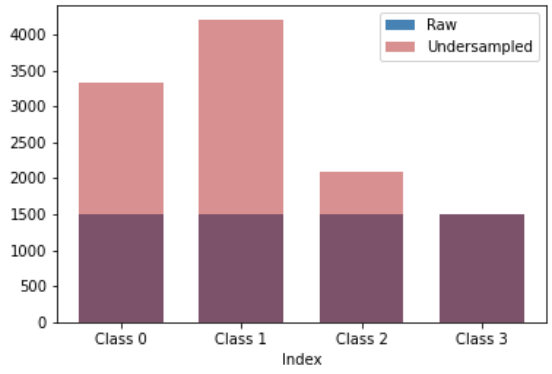


Figure 8. Data distribution by the various classes – raw and balanced

The dataset was balanced by randomly sampling 1,500 fMRI volumes within each class. The number of fMRI volumes to extract for each class was based on the total number of volumes within the smallest class (class label 3) which only contained 1,500 volumes. The resulting balanced dataset totaled 6,000 volumes in total while removing bias due to class imbalance.

### 3.2. Classifier Implementation

#### 3.2.1 kNN

K-nearest neighbors (kNN) is a supervised machine learning algorithm often used in classification and regression tasks. The main idea of KNN is that the proximal data points determine the classification/ predict for an input data point. kNN classifier determines the class of a data point by voting in the numbers. For example, if we have a K number with 5, the kNN algorithm will find the nearest 5 data points as voting members. The class with the highest vote total among neighbors will determine the class of the data point. A distance metric is used to find the nearest points. Distance metrics includes different types, the $\ell^p$- Minkowski Distance, $\ell^p$- Manhattan Distance, and other distances.

In this study, we used the kNN algorithm for classification. We split the train and test into the same 80%:20%. We used the k = 3, 5, and 11. The distance used the Manhattan Distance.

#### 3.2.2 SVM

A support vector machine (SVM) is a type of machine learning algorithm that operates by learning decision boundaries based on labeled training data [9]. SVMs use the training data to compute the optimal hyperplane that separates the vector space of the data points into the defined classes. Optimization of the hyperplane selection can be done by handling the data points surrounding the boundaries.

A maximum-margin hyperplane seeks simply to maximize the distance between the separating hyperplane and the nearest data vector [9]. A soft margin hyperplane modifies the maximum-margin hyperplane optimization procedure by allowing some training data points to fall on the incorrect side of the decision boundary [9]. Using a soft margin can help improve classification of data that cannot be separated cleanly into the desired number of classes but relies on a trade-off between the number of allowable misclassifications and the size of the margin between the hyperplane and the nearest vectors. Finally, kernel functions can be applied on the data to improve the separation of data into the desired classes over the previously described margin based hyperplane optimization. Kernel functions operate by projecting the data into a higher dimensional space which, depending on the kernel function, can cause the data

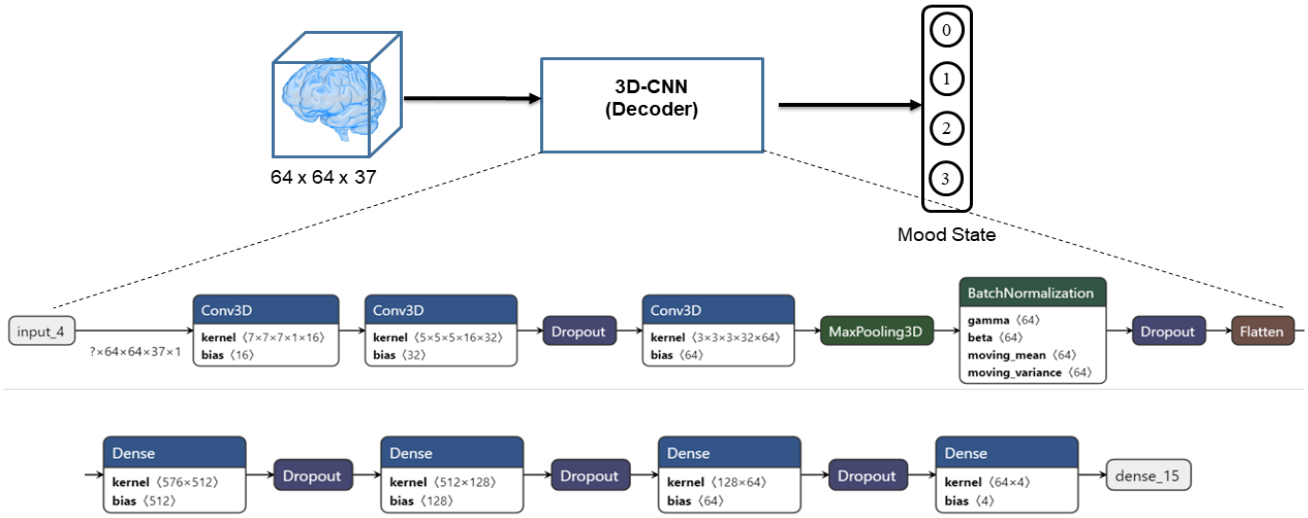


Figure 9. Baseline 3D-CNN architecture

to be separable with a plane in the higher dimensional space [9].

Linear kernels tend to require shorter training times but result in lower accuracy, while non-linear kernels tend to require longer training times but may achieve higher accuracy than linear kernels [9]. For the purposes of this project, we used a radial basis function (RBF) kernel for the SVM classifier to achieve the highest possible accuracy.

Principal component analysis (PCA) was conducted on the data input to the SVM in order to reduce the dimensionality of the data and enable reasonable training times. PCA features are computed by determining the transformation that maximizes the variance between the data points via eigendecomposition or singula value decomposition [10].

Implementation of the SVM classifier was done using the scikit-learn library for machine learning in Python [11]. The data was split into the same 80%:20% training:testing subsets as the other classifiers. The number of extracted PCA features was varied from 50 to 500 in intervals of 50. An additional SVM classifier was trained using PCA feature extraction but with excluding the first three PCA features, following the procedure in used for Eigenfaces [12]. In addition, an SVM classifier was trained using the full image data without PCA feature extraction in order to determine the trade off between classification accuracy and training time.

### 3.2.3 3D-CNN

We modified the 3D-CNN architecture used by the Vu et al. [7], which in turn is a modification to the LeNet5 architecture [13]. Our modifications were mainly along the lines of the number of feature maps, the number of fully connected (FC) layers and their hidden units; we doubled the number of feature maps and number of FC layers with more hidden units. Specifically, our baseline architecture comprised of an input layer that received an input volume of dimensions $64 \times 64 \times 37$, 3 preceding convolutional layers with a kernel sizes and feature maps of $7 \times 7 \times 7$ and 16, $5 \times 5 \times 5$ and 32, $3 \times 3 \times 3$ and 38 respectively. A stride length of 2 was used throughout. We included a dropout of 0.5 between the second and third layer synonymous with that of Vu et al.[7]. The convolutional block was terminated by 3D max pooling (pool size = 2) and batch normalization layers. The resulting feature map were then vectorized and passed to the first FC layer with 512 hidden units. The network was then terminated by 3 FC layers with hidden units of 128, 64 and 4. The ReLU activation function was used throughout with the exception of the final layer where we used a softmax activation.Also a dropout of 0.5 between the cascaded FC layers was adopted to address potential overfitting typical of networks with large number of parameters. The overall 3D-CNN architecture can be visualized in Figure 9.

The baseline (control) protocol for training the 3D-CNN is as follows – We split the data into training and test proportions with a ratio of $80\% : 20\%$. The test data was further split into a new test and validation sets with ratio of $80\% : 20\%$. A batch size of 50, categorical cross entropy loss (i.e. one-hot encoding of the labels), RMSprop optimizer and 2000 epochs were adopted. In our hyperparameter experiments, we formulated sub-control (reduced number/ scale) and supra-control (increased number/ scale) for hyperparameters – number of feature maps(FM), convolution kernel sizes (KS), Dropout rate (DR) and type of Optimizer (OPT). Our code implementation was done using tensorflow keras, with training done using the Google Colab Pro GPU.

## 4. Experiments and Results

**kNN:** The classification accuracy for the kNN classifier on the testing will summarized in the following figure.

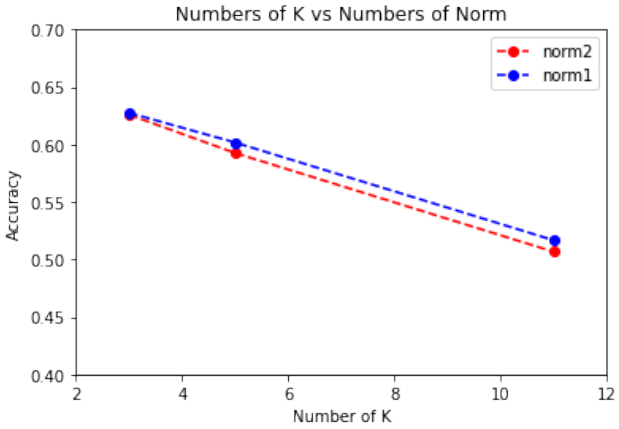


Figure 10. kNN Classification using different parameters

We did different experiments on the kNN classifier. k =3 with $\ell^1$- norm distance had an accuracy of 0.6275; k = 3 with $\ell^2$- norm distance had an accuracy of 0.6258. For k = 5 with $\ell^1$- and $\ell^2$- norm distance, accuracies were 0.6016 and 0.5925 respectively. The k = 11 with $\ell^1$- and $\ell^2$- norm distance had accuracies of 0.5166 and 0.506 respectively. The k = 3 with the $\ell^1$- norm with has the highest accuracy. This was an interesting observation as a simple classifier like kNN had test accuracies that paralleled that of the 3D-CNN and much better than the SVM.

**SVM:** The classification accuracy for the SVM classifier on the testing set when varying the number of PCA features extracted is summarized in figure 11.

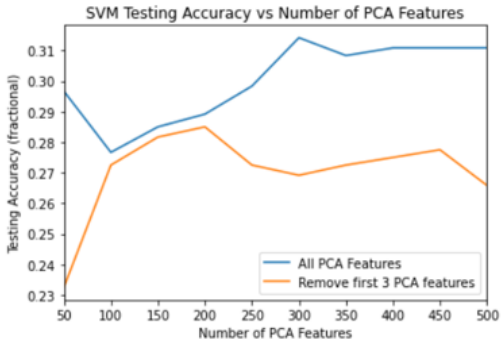


Figure 11. SVM classification accuracy using PCA features

In these experiments, PCA feature extraction seemed to take the majority of the training time and varied from 1 to 2.5 minutes when increasing the number of features to extract. After PCA extraction, training the SVM took less than 10 seconds in all cases. In comparison, when training an SVM classifier without PCA feature extraction, the total training time took upwards of 30 minutes to complete. As shown in the figure, the classification accuracy when using an SVM with PCA feature extraction reached a maximum of 31%. Interestingly, when following the procedure for Eigenfaces by removing the first 3 PCA features, the SVM classification accuracy seemed to get worse maxing out at 28%. The SVM classification accuracy was highest when training without PCA feature extraction at 46.8%. The higher accuracy without using PCA feature extraction may be due to the fact that PCA does not take into account between class variance and instead attempts to compute the maximum spread transformation for all data points. In addition, PCA lowers the number of available features which may increase the difficulty in defining a separating hyperplane for an SVM classifier.

**3D-CNN:** Experiments performed here involved tuning of selected hyperparameters generally known to be paramount in improving recognition rates while ensuring reduced overfitting. It must be noted that the chosen hyperparameters are not exhaustive; we limited the scope to four (FM, KS, DR and OPT) due to time constraints. Runtime for the control protocol training was about 1.1 hours to train, other protocols either had relatively higher or lower training times based on the number of trainable parameters. Figure 13 presents the training and validation courses over 2000 epochs from training using the control/ baseline architecture protocol.

The table presented in Figure 12 shows the number of trainable parameters, training and test accuracies for the control, sub-control and supra-control protocols for each of the four selected hyperparameters. In the case of optimizer type, we explored the Adam and Nadam optimizers available in the *keras.optimizers* module. The highest test accuracy obtained after all experiments was $59.4\%$ when the Nadam optimizer was used in training the baseline architecture. A corresponding training accuracy of $84.59\%$ was recorded, indicative of overfitting. This implies that introduction of more regularization interventions may increase the test accuracy. Figure 14 shows a breakdown of test accuracies by the four classes for the top accuracy protocol in a confusion matrix. While accuracies are not optimal, a somewhat balanced recognition is seen for the various classes.

## 5. Limitations, Challenges & Future Directions

Data preprocessing of fMRI volumes (slice timing correction, motion correction, spatial normalization and spatial smoothing, etc) are often manually done after inspecting the numerous examples. This is a tedious and extensive process. In our case, we extracted an initial 11,107 volumes, as such it would have not been feasible to manually

| Hyperparameter | Sub-control (*Sub*) | | | Supra-control (*Supra*) | | |
|---|---|---|---|---|---|---|
| | Train | Test | #Param | Train | Test | #Param |
| # feature Maps (FM)<br>*C=(16,32,64), Sub = C/2, Supra = 2C* | 0.496 | 0.503 | 254,900 | 0.367 | 0.393 | 1,153,412 |
| Kernel Sizes (KS)<br>*C = (7,5,3), Sub(3,3,3), Supra = (9,7,3)* | 0.648 | 0.543 | 439,524 | 0.507 | 0.500 | 448,580 |
| Dropout Rate(DR)<br>*C = 0.5, Sub = 0.3, Supra =0.7* | 0.8091 | 0.573 | 494,756 | 0.257 | 0.248 | 494,756 |
| Optimizer Type*<br>*C = RMSprop, Sub = Adam, Supra = Nadam* | 0.879 | 0.563 | 494,756 | 0.8459 | **0.594** | 494,756 |

Figure 12. Results from experiments – hyperparameter tuning and optimizer changes. *Optimizer type protocols do not follow the sub- and supra- line of thinking. Control protocol metrics: Train: 0.559, Test: 0.508, Param: 494,756

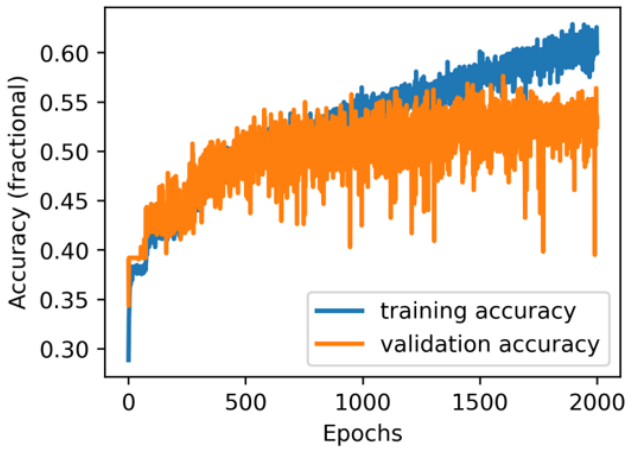


Figure 13. Baseline model: Train-Validation courses over 2000 epochs

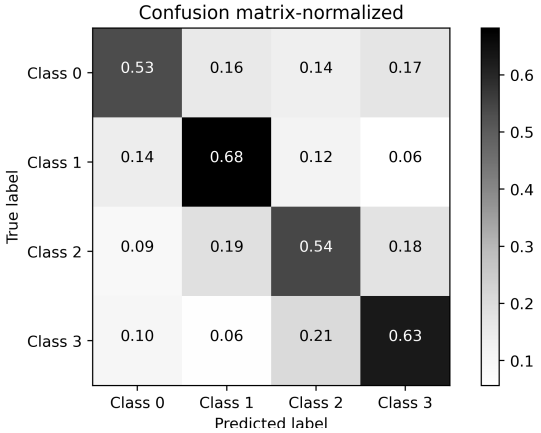


Figure 14. Breakdown of class accuracies for the top accuracy protocol (Nadam optimizer). A balanced recognition can be seen albeit sub-optimal accuracies

preprocess each volume. We, however, assumed the CNN could learn to identify general features unique to the various categories in the midst of noisy experimental factors. The three dimensional nature of the data hindered deploying pretrained models trained with 2D images. e.g AlexNet, VGG, ResNet, etc. Another challenge we faced was the limited computational resources relative to the memory requirements of the data we used. fMRI data are often expansive due to the increased data dimensionality. There was therefore competition for compute memory between the data and number of trainable parameters. A trade-off between these two had to be upheld which in turn constrained the number of convolutional fully convolutional layers we could deploy. Along this line of thought, the high data dimensionality was not commensurately met by the number of examples ($64 \times 64 \times 37$ verus $6,000$), thus, we executed this project amidst significant "*Curse of dimensionality*" [14]. Also, there was initially an imbalance in the class distributions as reported earlier, by undersampling the data to ensure a class-balanced data, we may have lost important general class features. While, it may be argued that oversampling could have been used, this would have further constrained the already limited memory we had. We recommend the use of compute hardware with high memory capacity to allow deeper networks to be explored. Also, we noticed significant overfitting during training even amid regularization strategies such as dropout rate. Increasing the proportion of regularization strategies may obviate this observation and possibly improve the recognition rate of the decoder. Additionally, an interesting tangent initially proposed for the project would be the generation of EEG adversarially from mood states. This is particulary useful in leveraging the spatial richness of fMRI inferences to generate EEG equivalents with high temporal richness that fMRI

are known to lack.

## 6. Conclusions

The importance of decoding neural activity in BMIs cannot be overemphasized; it allows for understanding of the intricate signals often recorded from the brain. Appropriate interpretation of neural signals can better inform feedback control in the course of rehabilitation. In this project, we explored the use of machine learning techniques specifically by using 3D-CNNs in decoding four music-induced valence-arousal mood states encoded in an open source fMRI data. We achieved a balanced recognition over all four classes albeit at sub-optimal accuracies. Results, however, obtained indicate that with more data, computational resources and regularization; recognition rates may be significantly improved. Code available at **https://github.com/Adakwaboah/DL_SP21_Mood_Decoder_Project**.

## 7. Member Contributions

All authors contributed equally. A breakdown of the various contributions are described below,
**Akwasi D. Akwaboah:** constructed the 3D-CNN and accompanying training and hyperparameter tuning; collaborated with other authors in data prep; report writing; and team coordination.
**Kevin Tran:** extracted fMRI images aligned with mood-state labels; conducted SVM experiments with and without PCA feature extraction; report writing
**Feng-Chiao Lee:** Data preprocessing and generating the mood labels for the classification; conducted KNN experiments with different numbers of nearest neighbors and with different norm distance; report writing

## References


[1] M. M. Shanechi, "Brain–machine interfaces from motor to mood," *Nature neuroscience*, vol. 22, no. 10, pp. 1554–1564, 2019. 1

[2] M. A. Lebedev and M. A. Nicolelis, "Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation," *Physiological reviews*, vol. 97, no. 2, pp. 767–837, 2017. 1

[3] I. Daly, D. Williams, F. Hwang, A. Kirke, E. R. Miranda, and S. J. Nasuto, "Electroencephalography reflects the activity of sub-cortical brain regions during approach-withdrawal behaviour while listening to music," *Scientific reports*, vol. 9, no. 1, pp. 1–22, 2019. 1, 2

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012. 2

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 2

[7] H. Vu, H.-C. Kim, M. Jung, and J.-H. Lee, "fmri volume classification using a 3d convolutional neural network robust to shifted and scaled neuronal activations," *NeuroImage*, vol. 223, p. 117328, 2020. 2, 5

[8] E. D.-C. Roddy Cowie, S. Savvidou, E. McMahon, and M. S. . M. Schröder, "'feeltrace': An instrument for recording perceived emotion in real time," *Conference Paper*, 2000. 2, 3

[9] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006. 4, 5

[10] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987. 5

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011. 5

[12] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997. 5

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 5

[14] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966. 7