A TECHNICAL REPORT ON DESIGN OF A MENTAL WORKLOAD CLASSIFICATION MODEL USING CNN AND LOGISTIC MODELS

PREPARED BY

YUNUS ALADE OMOFOWOBI

DATE:

MARCH, 2023

ABSTRACT

This report introduces a logistic regression and convolutional neural network (CNN) based model for classifying mental workload. The study uses a dataset provided for this study purpose, and the models are trained and tested using the dataset. The results of the model's performance evaluation are discussed in the report, highlighting its effectiveness (accuracy) in classifying mental workload.

The need for accurate and efficient classification of mental workload has become increasingly important in recent years due to the rise in jobs that require high levels of cognitive demand. The classification model proposed in this report addresses this need by utilizing both logistic regression and CNN models, which have been implemented in this classification task.

To ensure the accuracy of the model, the dataset used for training and testing was carefully selected to include a diverse range of mental workload levels. The dataset includes physiological signals, such as electroencephalography (EEG), which are commonly used to measure mental workload levels.

The logistic regression model was first trained and tested using the dataset, followed by the CNN model. The results show that both models were able to accurately classify mental workload levels, with the CNN model outperforming the logistic regression model.

The proposed classification model has several potential applications, including optimizing workloads in various job settings, improving user interfaces, and developing personalized training programs. Overall, the model presented in this report demonstrates the potential of combining logistic regression and CNN models for accurate classification of mental workload levels.

INTRODUCTION

The field of neuroscience has extensively employed Electroencephalography (EEG) signals to study various cognitive processes like attention, memory, and decision-making. Machine learning techniques have recently gained attention for analyzing EEG data to enhance the accuracy and efficiency of EEG-based cognitive assessments.

In this project, we aim to develop and compare two models for EEG-based mental workload classification. First, we will program a logistic regression model from scratch in Python, using separate functions for each step (e.g., forward propagation and gradient calculation). We will also select a publicly available package/toolbox to adapt a deep learning model for mental workload classification. We will evaluate the performance of these models on a carefully selected dataset using five-fold cross-validation and compare their classification accuracy. The dataset consists of physiological signals, such as EEG, and includes a diverse range of mental workload levels to ensure the accuracy of the models.

Logistic regression is a commonly used linear model for binary classification problems. It assumes that the relationship between the input variables and the output variable is linear. Deep learning models, on the other hand, are a class of neural network models that are capable of learning complex non-linear relationships between the input and output variables. Deep learning models have been shown to outperform traditional machine learning models in many tasks, including image recognition, speech recognition, and natural language processing.

The results of this project have the potential to improve our understanding of mental workload and inform the development of tools and interventions for optimizing cognitive performance in high demand settings. By comparing the performance of logistic regression and deep learning models, we can determine which approach is more effective for mental workload classification based on EEG data. The proposed models can be used to optimize workloads in various job settings, improve user interfaces, and develop personalized training programs.

METHODOLOGY

This study evaluates the logistic regression and CNN-based model as a tool for classifying mental workload. The study uses EEG-Based Multi-Class Workload Identification dataset - Using Feature Fusion and Selection - provided for this study purpose, and the models was trained and tested using the dataset. This section discusses the step by steps of processing, model architecture, and training process it also includes the evaluation metrics used to measure the model's performance.

COLLECTION AND PREPROCESSING OF THE DATASET

The dataset used in this study, titled "EEG-Based Multi-Class Workload Identification Using Feature Fusion and Selection," was previously utilized by Pei et al. in their 2021 publication in IEEE Transactions on Instrumentation and Measurement. The dataset was obtained from the following link: https://essexuniversity.box.com/s/n7ubo7cf2ta2hj25yg7fsvbr0rmwcf8x.

IMPORTING THE DATASET INTO PYTHON ENVIROMENT

To import the dataset into Python, the scipy.io package was used. The dataset comprises of two variables: "Data" and "Label". The "Data" variable holds EEG signals, whereas the corresponding labels are stored in the "Label" variable. The dimensions of the data are 62 x 512 x 360, where 62 is the number of channels, 512 is the number of data points in each sample, and 360 represents the total number of samples. The label uses 0 to indicate the low workload class and 1 to indicate the medium workload class.

CONVOLUTIONAL NEURAL NETWORK (CNN)

This is a type of neural network that excels in image classification tasks. It comprises several layers of neurons, including convolutional, pooling, and fully connected layers. The convolutional layers are responsible for recognizing local patterns and features in the input image by applying a set of filters. These filters learn various features such as edges, corners, and shapes during the training process. The pooling layers reduce the dimensionality of the feature maps by taking the maximum or average value of a small region of the feature map.

Finally, the fully connected layers combine the features learned from the previous layers to make the final classification decision. CNN-based models are trained using a backpropagation algorithm with optimization algorithms like stochastic gradient descent. Regularization techniques such as dropout and weight decay can be used to avoid overfitting.

PREPROCESSING THE DATASET

The 4-dimensional NumPy array ('data_q') with dimensions (360, 62, 512) is reshaped to have the correct shape (360, 512, 62, 1) for the input to a convolutional neural network model, where the dimensions correspond to the number of samples, height, width and channels of the input data. The last argument '1' specifies that the fourth dimension has a size of 1 which means that the data is grayscale (i.e., it is a one-color channel)

SPLITTING THE DATASET

The sklearn package was used to split both the data and label variables into testing and training set

DEFINING THE MODEL SCRUCTURE

The sequential function is used to create the neural network, and then the layers are added one by one. The model starts with a '**Conv2D**' layer with the 32 filters of size 3x3, which applies **ReLU** activation function to the input data. The '**input_shape**' parameter specifies the shape of the input data as a 3D tensor of size (512, 62, 1). The 'BatchNormalization' layer is added after each convolutional layer to improve the performance and speed up the training process. The '**MaxPooling2D**' layer with a pool size of 2x2 is then added to reduce the size of the feature maps produced by the convolutional layers. This sequence of convolutional and pooling layers is repeated twice, with the 64 and 128 filters respectively.

A '**Flatten**' layer is added to convert the 2D feature maps into 1D vector, which is then passed through two fully connected '**Dense**' layers with 256 and 128 neurons respectively. A dropout rate of 0.5 is applied to the second Dense layer to prevent overfitting. Then a 'Dense' layer with a single neuron and sigmoid activation function is added to produce the binary classification output.

COMPILING AND TRAINING THE DESIGNED MODEL

Using **Keras** the ‘**compile**’ method is used to specify the loss function, optimizer, and evaluation for the model. the loss function is set to ‘**binary_crossentropy**’, the optimizer is set to ‘**adam**’ which is a stochastic gradient descent optimization algorithm. Once the model is compiled, it is ready to be trained on the training dataset using the ‘**fit**’ method.

LOGISTIC REGRESSION

Binary classification is a common problem in machine learning where the goal is to predict the occurrence of a binary outcome, such as yes or no, true or false, or 0 or 1. One popular technique for solving binary classification problems is logistic regression. This technique models the probability of the binary outcome as a function of the input features. It uses a sigmoid function to map the output of a linear function to a probability value between 0 and 1. The sigmoid function is defined by the equation: Sigmoid(x) = 1 / (1 + e^(-x)), where x is a linear combination of the input features and their respective weights.

The logistic regression model can be trained using maximum likelihood estimation, where the goal is to find the values of the weights that maximize the likelihood of the observed data. During the training process, the weights are adjusted to minimize the difference between the predicted probability and the true label. Once the model is trained, it can be used to predict the probability of an event occurring for new input data. If the predicted probability is above a certain threshold, the model will predict a positive outcome; otherwise, it will predict a negative outcome.

Logistic regression is a linear model that is widely used in machine learning for binary classification problems. It is easy to interpret and implement, and it can be trained efficiently even on large datasets. However, it assumes that the relationship between the input features and the output variable is linear, which may not be true in some cases. In such cases, more complex models such as deep neural networks may be used to capture the non-linear relationships between the input features and the output variable.

PREPROCESSING THE DATASET

The EEG signal stored in the data variable was processing to reduced it dimension to a two-dimensional array.

SPLITTING THE DATASET

I used the sklearn package to split both the data and label variables into testing and training set.

DESIGNING THE MODEL

The Model Comprises of Three (3) Fundamental Steps. Which Are Listed Below:

- **Forward Propagation:** In logistic regression, it involves computing the weighted sum of the input features and the corresponding weights, the sigmoid function is then applied to the result. The sigmoid function outputs value 0 and 1 which represents the probability of the input belonging to a certain class. This process was repeated for each input in the dataset to generate the predicated probabilities for each class. The predicated probabilities are then used to compute the loss function which measures the difference between the predicted and actual class labels.
- **Backward Propagation:** In logistic regression, backward propagation is used to calculate the gradient of the loss function with respect to the weights and biases of the model. The gradient represents the direction and magnitude of the change required in the weights and biases to minimize the loss function. To perform backward propagation, the partial derivatives of the loss function with respect to the weights and biases are calculated using the chain rule. The gradients are then used to update the weights and biases of the model using an optimization algorithm such as gradient descent.
- **Gradient descent:** is an iterative optimization algorithm that uses the gradient of the loss function to update the weights and biases of the model. The algorithm adjusts the weights and biases in the opposite direction of the gradient, such that the loss function is minimized with each iteration. The learning rate, which determines the step size of the update, is also an important hyperparameter in gradient descent. A higher learning rate may lead to faster convergence but may also result in overshooting the optimal weights and biases, while a lower learning rate may converge slowly but provide more accurate results.

OBSERVATION

The CNN gave a more accurate result compare to the logistic function hence making it a better model to use

REFERNCE

Li, Y., Chen, F., Li, Y., Ma, H., & Li, Y. (2018). Classification of mental workload levels using electroencephalogram signals and support vector machine. International Journal of Industrial
Ergonomics, 65, 69-78.

Yu, J., Ang, K. K., Ho, C. S., Guan, C., & Li, X. (2018). EEG-based mental workload classification using deep belief networks. Neurocomputing, 275, 1263-1271.

Zhou, X., Liu, S., Li, X., Wang, Y., Zhang, Z., & Li, X. (2019). Classification of mental workload levels using heart rate variability signals and random forest. Frontiers in Neuroinformatic, 13, 76.

Pei, Z., Wang, H., Bezerianos, A., & Li, J. (2021). Dataset: [WLDataCW.mat]. Available at: https://essexuniversity.box.com/s/n7ubo7cf2ta2hj25yg7fsvbr0rmwcf8x