Master's project report

Data augmentation for ERP data in brain-computer interface (BCI)

Anup Kumar

anup.kumar@uranus.uni-freiburg.de

Matriculation number: 4051181

Supervisors: Dr. Michael Tangermann and David Hübner

University of Freiburg
Brain State Decoding Lab
Department of Computer Science

Contents

Abstract

Brain-computer interface (BCI) is a programmed device that acts as a bridge between the brain and external activities. It deciphers signals (like electroencephalography (EEG)) emanating from the brain and translates them into activities like moving the cursor on a screen or moving a wheelchair. It brings an immense help to the people with disabilities to communicate with the environment. The EEG signals captured over the scalp are weak and they tend to be contaminated by disparate sources of noise like the muscle or eye artifacts and non-neural phenomenon. Moreover, signals like the ERP (event-related potential) and motor imagery to calibrate a BCI are limited in number due to a difficult process of data acquisition. On the other hand, to make BCI work robustly we need a large set of signals. To solve this scarcity, we explore possibilities to generate artificial signals using the original ones. This process is called data augmentation. Additionally, we expect an increase in classification accuracy after enhancing the original signals with artificial ones [1, 2]. So far, multiple techniques have been proposed in the areas of image processing, computer vision and speech recognition to enlarge the original data by applying inventive domain-specific and class-preserving transformations which include warping of characteristics like rotation, scaling and translation for images, frequency and amplitude for signals and even adding noise to original data. Likewise, in BCI too, we propose to do few transformations of original signals like modulating amplitudes and time shifting to create novel ones and train our classifier on a larger set of signals. In addition to the evaluation of classification performance with the data augmentation, we take a note of the risks involved in creating novel signals and processing a larger set of signals.

1 Introduction

1.1 Motivation from history

Professor Hans Berger, in 1924, at the University of Jena discovered that the brain signals could be recorded (as EEG) and used for brain research. This discovery laid the foundation of one of the most promising areas of research and interest in recent times. It has a noble cause primarily directed at differently abled people who need aid in hearing, movement or sight.

1.2 Electroencephalography (EEG)

The EEG captures electrical potentials over the scalp using electrodes originated from the brain activities. These electrical potentials originate from the ionic currents (ion movements) of mostly $Na+$ (sodium), $K+$ (potassium), $Ca++$ (calcium), and $Cl-$ (chloride) ions. The active neurons (brain cells) give rise to these ionic currents which are recorded as the EEG signals. The measured signals need to be amplified because they are weak even though they require many neurons to be active to produce them. The EEG signals are widely used for the diagnosis of many neural diseases like epilepsy and seizures. Being non-invasive makes it readily usable and risk-free [3].

1.3 Brain-computer interface

Brain-computer interface (BCI) is a device that decodes these EEG signals to perceive the underlying mental intention and translates them to do a work (like moving the cursor on a computer screen) without the need of neuro-muscular or hormonal activity. This workability comes as a colossal help to the people with disabilities to successfully interact with their environment like moving a wheelchair. The workflow of a BCI comprises of analysing the EEG signals to extract useful features, training a classifier on these extracted features to get a trained model, using this model to categorise new signals, giving communication commands to perform an external activity and eventually receiving feedback (in some cases) [4]. Nowadays, it is expanding its reach to become accessible in human-computer interaction and entertainment where devices are controlled by the mind.

1.4 Challenges in brain-signal processing

Recent years have seen the field of BCI research attracting many people from across the globe and establishing itself as an absorbing area of interest, all dedicated to understanding one of the most complex systems known to us - the human brain. While possessing a huge potential for further studies, it bears challenges too which include long calibration time of the BCI enabled devices, inter-subject and inter-session variations in captured signals, not being universally accessible and practicable, not being able to classify germane information from the signals, availability of fewer signals, recording being inconvenient and time-consuming and high dimensionality and non-stationarity of the signals [1, 5, 6]. Out of these challenges, we explore multiple ways to work out the problem of availability of fewer signals in this study.

1.5 Event-related potential (ERP) data

The ERP data [1] represents deflections in the EEG which originates from an external stimulus (or absence of it). These deflections constitute a response by the brain to the stimulus. They are categorised based on the timing of deflections. The early component around $100ms$ after the stimulus is called $N100$. Any uncertainty in the stimulus gives rise to $N100$. If the stimulus is predictable, $N100$ is weak. Otherwise, it is strong. The deflections that occur later show an advanced level of neural processing. For example, $P300$ represents a positive deflection that occurs at around $300ms$ after presenting the stimulus. Having a higher concentration on the part of any subject produces larger amplitudes for this component [7].

[1] ERP signals are referred to as ERP data. In the later sections, we will refer to trials and epochs instead of signals.

The usage of the ERP data possesses certain advantages such as non-requirement to train users before being a part of the recording sessions. But, on the other hand, they suffer from few drawbacks as well like not being strong (in the order of micro-volts) and having low signal to noise ratio (SNR) due to the presence of artifacts. Moreover, they also suffer from inter-subject and inter-session variations in delay and waveforms [8].

ERP data is a set of epochs and arranged as a spatio-temporal matrix. An epoch represents 1.4 seconds of continuous data obtained from the scalp. Its dimensionality is $R^{T \times C}$ where C is the number of scalp electrodes (also called channels) and T is the time-steps. If the epochs are sampled at $100Hz$, T is 140 (each time-step is equal to $10ms$). At $1000Hz$ sampling frequency, T becomes 1400 (each time-step is $1ms$). For each user, the number of epochs is 3240. If data is sampled at $100Hz$, we work with a matrix of size $R^{140 \times 63 \times 3240}$. Each epoch has one label or belongs to a class. For our ERP data, there are two classes - target and non-target. Figure 1 shows a sample epoch for channel (or electrode) 'P3' which sits over the left parietal lobe of our brain. Figure 2 shows the positions and names of the EEG electrodes (channel P3 in the bottom left).

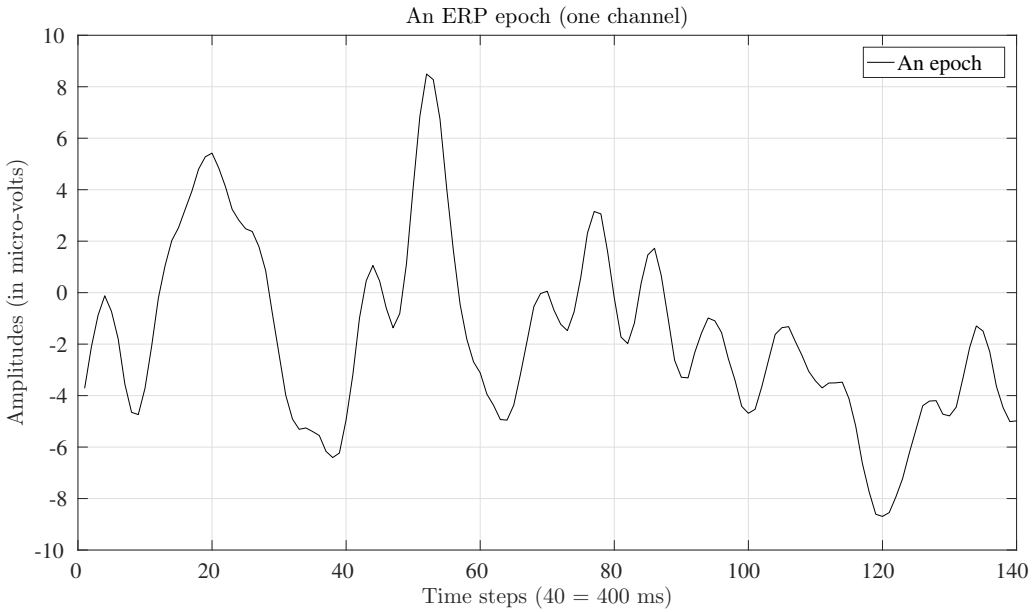

Figure 1: *An ERP epoch (channel P3).*

The number of epochs for ERP data for any user is low (3240). Moreover, the number of target epochs are considerably less compared to the non-target ones. In order to increase the number of epochs to train a classifier on a larger number of epochs, we propose to generate artificial (or novel) epochs from the original ones using a number of domain-specific transformations like amplitude modulation, time warping and so on.

1.6 Motivation for data augmentation

The brain data recording sessions require users to maintain high concentration which ensures that the true mental intentions get captured in a precise way and the recordings are of good quality. But, this necessity of retaining high concentration for the experiment limits the amount of (good quality) data. One of the reasons behind this is the difficulty for humans to concentrate for a long period of time. Acquiring good quality data becomes more ambitious when we deal with the people having severe disabilities. The ERP data suffers from this scarcity.

This scarcity can be reduced by generating artificial data (epochs) by doing multiple domain-specific transformations on the original data (epochs). These transformations like amplitude modulation, time

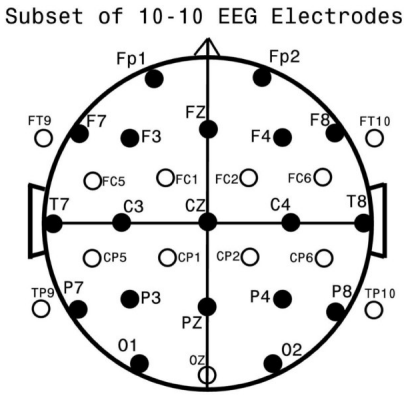

Figure 2: *The names and positions of standard EEG electrodes (channels). Some notations for electrodes code names - Fp: Frontopolar; F: Frontal; C: Central; T: Temporal; P: Parietal; O: Occipital, Z: Midline: FZ: Midline Frontal; CZ: Midline Central; PZ: Midline Parietal; OZ: Midline Occipital. Even numbers for the right part and odd for the left part of the brain [9].*

shifting etc. preserve the original labels of the epochs. In general, this process of artificially enlarging the original data is called data augmentation.

In the field of machine learning (especially computer vision and speech recognition), data augmentation is widely used to achieve specific purposes like to balance the data in terms of classes (each class has roughly the same amount of data), create more variations in the data for the classifier to learn and prevent overfitting which eventually leads to a better generalisation on the unseen data. We illustrate few examples where data augmentation has been successfully used.

Overfitting is a phenomenon which occurs when a classifier tries to memorise the training data. Because of this, classification accuracy on training data increases but decreases on test data. The classifier fails to generalise on the unseen data. In the work by Alex Krizhevsky [10], data augmentation is used to reduce overfitting. There are as many as 60 million parameters to learn and the original data (images) is insufficient to learn these many parameters. To generalise better and create more variations in the training set of images, they are transformed by doing translations and horizontal reflections to generate an enlarged set (by a factor of 2048). Another strategy in the same work is to vary the *RGB* (red, green and blue channels) intensities of images by employing principal component analysis (PCA) on the *RGB* intensities. The latter approach makes use of an important property of images that the identity of any object present in an image is invariant to the colour and illumination of that image. It reduces the top-1 error rate by over 1%. When the predicted class with the highest probability does not match with the true class, it attributes to the top-1 error. Similarly, the top-5 error rate is calculated for the top 5 predicted classes not matching with the true class.

For speech recognition, data augmentation is used to achieve an improvement in classification accuracy. An approach, vocal tract length perturbation (VLTP), is used to artificially generate a replica of the original data by multiplying the original vocal tract length of every speaker with a warping factor α. The values of α are chosen randomly from $[0.9, 1.1]$. The generated data is slightly perturbed as compared to the original data [11]. In [12], a slightly different mechanism of choosing warping factor is used. It is deterministic and not random as in the previous approach. The vocal tract length normalisation (VLTN) of each speaker is altered by first estimating the VLTN warping factor α and then taking out the values of α as:

$$\alpha \longleftarrow \{\alpha - 4, \alpha - 2, \alpha + 2, \alpha + 4\} \tag{1}$$

This data augmentation approach measures a drop of about 0.5% to 1% in the error rate using convolutional neural network.

In addition to the above two, a successful use of data augmentation for voice detection is explored in [13]. The authors use a mixture of strategies to enlarge training data that include data dependent

(pitch-shifting and time-stretching of audio data) and data independent (dropout - setting inputs to 0 with a probability and adding gaussian noise) strategies. One more interesting way is mixing two training samples (one positive and another negative) to make a new sample which inherits more properties from one of the samples and retains its class label. For example, the new sample takes more components from the positive sample and less from the negative sample. Its label remains positive. The results are reported for different strategies. For example, pitch shift ($\pm$10%) achieves over 2% reduction in error rate (an improvement) while time stretch ($\pm$50%) increases the error rate by over 1% (not an improvement).

The work in [14] done more than a decade back also establishes that the data augmentation can improve generalisation by making a classifier learn transformation invariance. Two augmentation strategies explored in this work are the affine and elastic transformations of MNIST digit images. The affine transformations include translation and rotation of original images and elastic transformation distorts images while preserving their labels. The elastic distortion performs better by recording lower classification error rate than affine transformations. Convolutional neural networks (CNN) with elastic and affine distortions record 0.4% and 0.6% error rates respectively on MNIST images. Figure 3 shows the transformed images (b, c and d) of digit 6 generated from an original one (a).

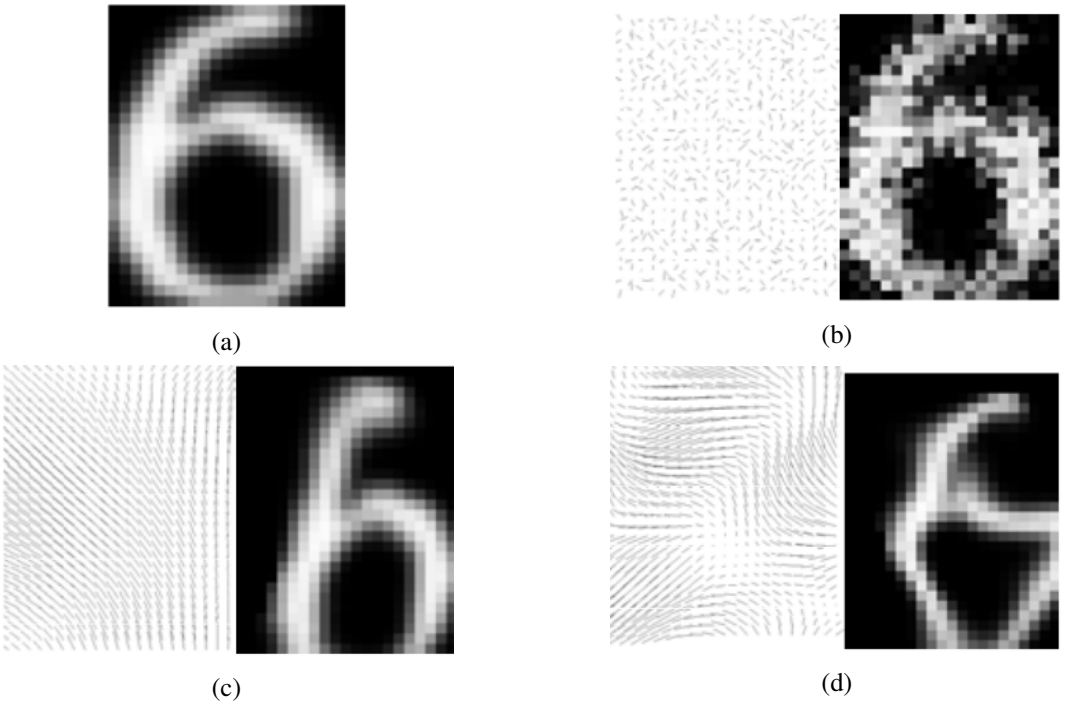

Figure 3: Generating artificial images by transforming an original image [14]. Figure (a) is the original image showing digit 6. The other images (b), (c) and (d) are the artificial ones created using the respective displacement fields having various ways of smoothing.

So far, we have discussed the successful implementation of many data augmentation strategies in various fields. Now, let's have a look at a couple of data augmentation studies conducted in the field of BCI in the following section.

2 Data augmentation in BCI

Fabien Lotte in [1] crafted a mechanism for generating artificial EEG trials in BCI which intermixes segments of original trials to create new EEG trials. Each EEG trial has a dimensionality of $R^{C\times S}$ where C is the number of channels and S is the number of samples in each trial. An EEG trial belongs to a class or label. Each EEG trial is divided into multiple segments and these segments are consecutive and non-overlapping. To generate new trials, a fixed number of segments are chosen from random trials and

are concatenated (figure 4). The approach follows these steps:

1. Bandpass filtering of EEG trials in $8-30Hz$.
2. Feature extraction using common spatial pattern (CSP) filter.
3. Classification using linear discriminant analysis (LDA).

It uses covariance matrix shrinking (CMS) for CSP and LDA to have an improved estimate of covariance matrices. It achieves higher classification accuracy as compared to an approach which uses only the original trials especially when the number of training trials per class is less than 40 [1]. For example, when the number of training trials per class is 20, the increase in classification accuracy is about 2%. However, when the number of training trials increases, the classification performances of artificial data generation and covariance matrix shrinking remain the same.

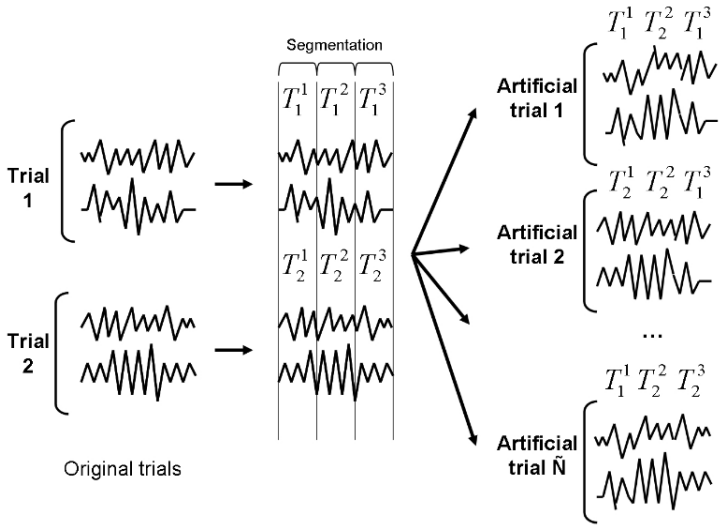

Figure 4: *Generating artificial EEG trials in BCI [1].*

Another approach described in [2] transforms each EEG trial into riemannian[2] space and then interpolates to create new EEG trials. These novel EEG trials are within the convex hull of the original EEG trials (in riemannian space) and densify the space for each class. Classification is done using multi-layer perceptron and variants of support vector machine and linear discriminant analysis. Again, this work records better accuracy when there is only a few number of EEG trials per class. Moreover, this works to decrease imbalances in the number of EEG trials belonging to each class (for ERP data). The augmented trials are classified with multi-layer perceptron and the classifier records about 1.5% increase in classification compared to no augmentation (also with multi-layer perceptron). This study also finds out that the augmentation works better when the number of training EEG trials per class is low.

We conclude from these approaches used in the BCI and other fields that higher classification accuracy can be achieved when a classifier is trained on an enhanced dataset. Hence, motivated by these successful studies, we propose to use the following strategies to augment ERP data:

- Amplitude modulation
- Time warping
- PCA modulation
- ICA modulation

[2] `http://www.matematik.lu.se/matematiklu/personal/sigma/Riemann.pdf`

3 Methods

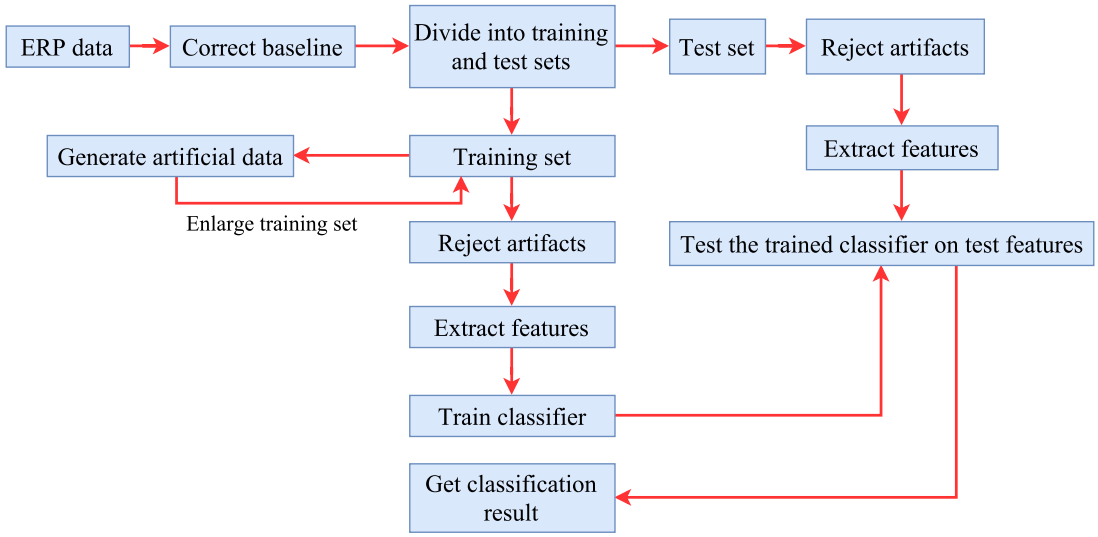

Figure 5: *Data augmentation workflow.*

The overall workflow of our data augmentation strategy is shown in figure 5. From now on, we refer to epochs instead of EEG trials. First, we correct the baseline of the ERP data (complete set of epochs). Then, under 5-fold cross-validation, these epochs are divided into training and test epochs (or sets). Artificial epochs are created using the original training epochs following a data augmentation strategy. Artifacts are removed from these artificial epochs. Then, artifacts are rejected from the original training epochs. The artificial epochs together with original training epochs form an enhanced set of training epochs. Features are extracted from this enhanced set. A classifier is trained on the features extracted from the enhanced set to get a trained model. Separately, artifacts are also rejected from the test epochs and their features are extracted. Finally, the trained model is evaluated on the test features to get a classification accuracy. In the following sections, we describe multiple ways in which we can generate artificial epochs from original epochs.

3.1 Amplitude modulation

The epochs are recorded as varying amplitudes (in micro-volts) over time using multiple electrodes placed on the scalp. To generate artificial epochs, we modulate the amplitudes of original epochs. To achieve that, we choose a factor and name it as an amplitude modulation factor α. This factor is randomly chosen from $[0.9, 1.1]$. Each novel epoch has a different modulation factor. All the channels in an epoch have the same modulating factor. Together for all artificial epochs, these factors follow a uniform distribution. The expected value of these factors is close to 1. We generate a new epoch using the following equation:

$$epoch^{new} = \alpha \cdot epoch^{original} \tag{2}$$

where $epoch^{original}$ is an original epoch, α is a random amplitude modulating factor and $epoch^{new}$ is a novel epoch. α is a scalar and an epoch has dimensionality $R^{140\times63}$. We vary the amplitudes in an amount to create small variations. Otherwise, if we modulate any epoch by a large amount, we run the risk of distorting it to an extent where it does not remain a representative of its class. Figure 6 demonstrates an example of amplitude modulation for one channel in an epoch. To an original epoch, a modulating factor ($\alpha = 0.90$) is multiplied to get a new epoch. Amplitudes of the novel epoch is smaller in magnitude as compared to the original epoch.

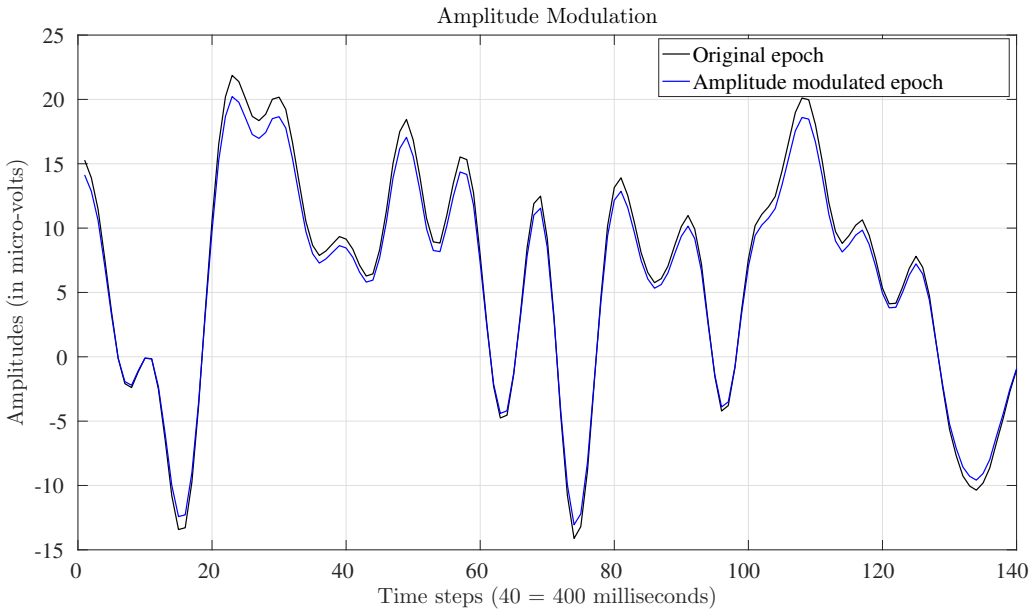

Figure 6: *Amplitude modulation (for a channel).*

3.2 Time warping

The electrical potentials over scalp are recorded over time and they show latencies in describing the processing of the stimulus presented to a user. The recorded epochs are non-stationary which means that these latencies shift in time by different amounts in different sessions (user data recording sessions) for a user or from user to user. In another scenario when a user loses concentration (due to tiredness) while recording, we can expect time-shifts in the processing of the stimulus. Taking hints that these time-shifts occur naturally in the recorded epochs, we propose to induce time-shifts in the original epochs to create novel ones. These time-shifts can either be forward (shift right) or backward (shift left) in time. Since the onset of the stimulus, these shifts start off slowly (low in magnitude) and gradually increase over time. It means that in an epoch of length 1.4 seconds, the time-shifts in the beginning are low and towards the end of the epoch (away from the stimulus presented at $time = 0$), they are large. We set a maximum achievable time-shift to $200ms$ at the time-step $1200ms$. The reason that explains it is that when we extract features, we consider an epoch only upto the 1200^{th} time-step and discard the samples beyond that time-step. If we perform time-shift of $200ms$ at the 1200^{th} time-step either forward or backward, we retain the original values from the epoch in the time-shifted epoch. Beyond the 1200^{th} time-step, the shift remains at $200ms$ till 1400^{th} time-step. For example, in an epoch to time-shift the value at 1200^{th} time-step forward by $200ms$, we need value from the 1000^{th} time-step to replace and the value at 1200^{th} time-step will move to 1400^{th} time-step. If we shift backwards by $200ms$ at 1200^{th} time-step, the amplitude value at 1400^{th} time-step will move to 1200^{th} time-step. To enforce this time-shifting method, we will have a linear function which allows us to have a maximum shift of $200ms$ at 1200^{th} time-step. Equation 3 gives the slope of this linear function. Figure 10 shows this linear function.

$$slope = \frac{200-0}{1200-0} = \frac{1}{6} \tag{3}$$

We implement the following steps to generate novel epochs shifted in time using the original epochs. Figure 13 shows an original and time-shifted epoch (for a channel).

1. Extract a user's data at $1000Hz$ sampling frequency.
2. Generate normally distributed random samples with 0 mean. The number of samples depends on the number of novel epochs to be created and an offset. An offset is added in order to avoid the

initial region of the samples and its value is set to $70,000$. It is kept in a multiple of total time-steps ($1400ms$) of an epoch. Figure 7 shows these random samples.

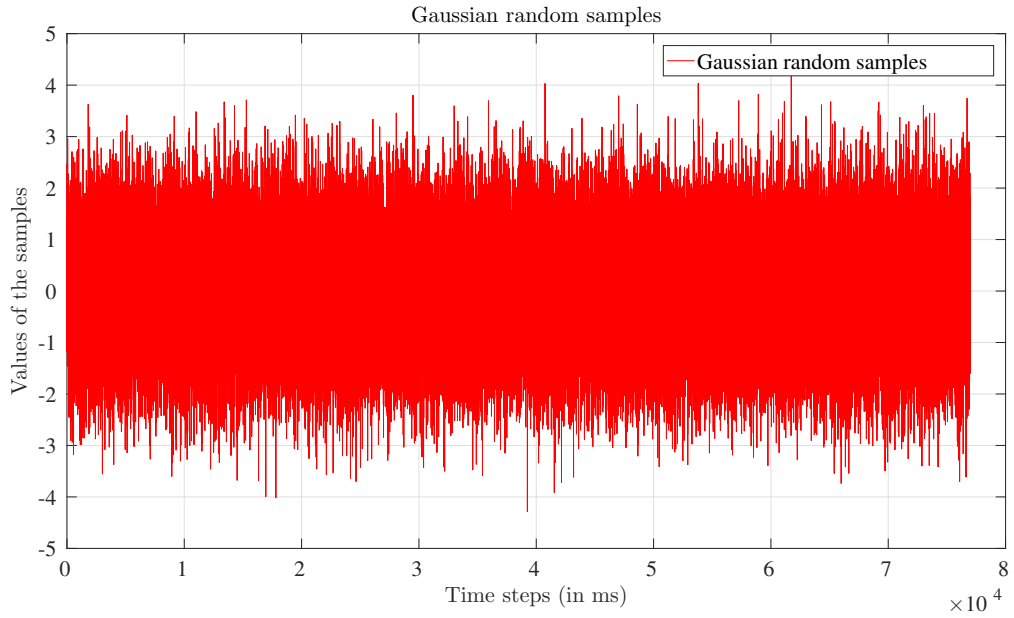

Figure 7: *Gaussian random samples.*

3. Filter a curve in the frequency range $[0.2Hz - 0.5Hz]$ at $1000Hz$ sampling frequency from the random samples. A low-pass filter is used with a filter-order of 8. Figure 8 shows this filtered curve.

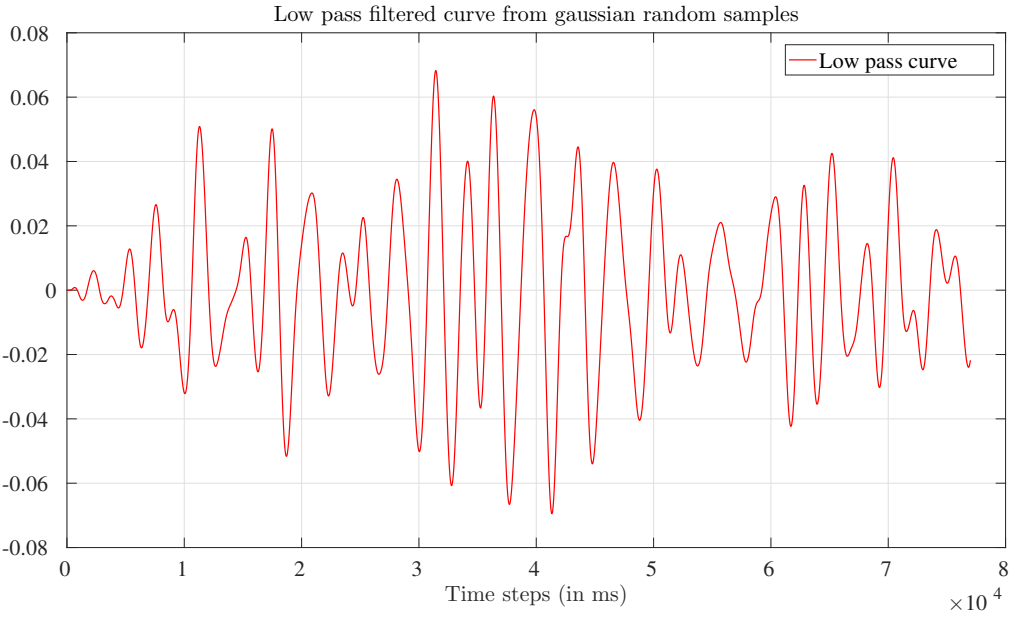

Figure 8: *Low pass filtered curve from gaussian random samples.*

4. Take out sections (or parts), each of length $1400ms$, from the filtered curve. Scale each section between -1 and 1 because this scaled curve when multiplied to the linear function will give time-

shift defining curve (or warp defining curve). Each section corresponds to each epoch to be time-shited. Now, let's call each section as the scaled curve. Figure 9 shows one scaled curve.

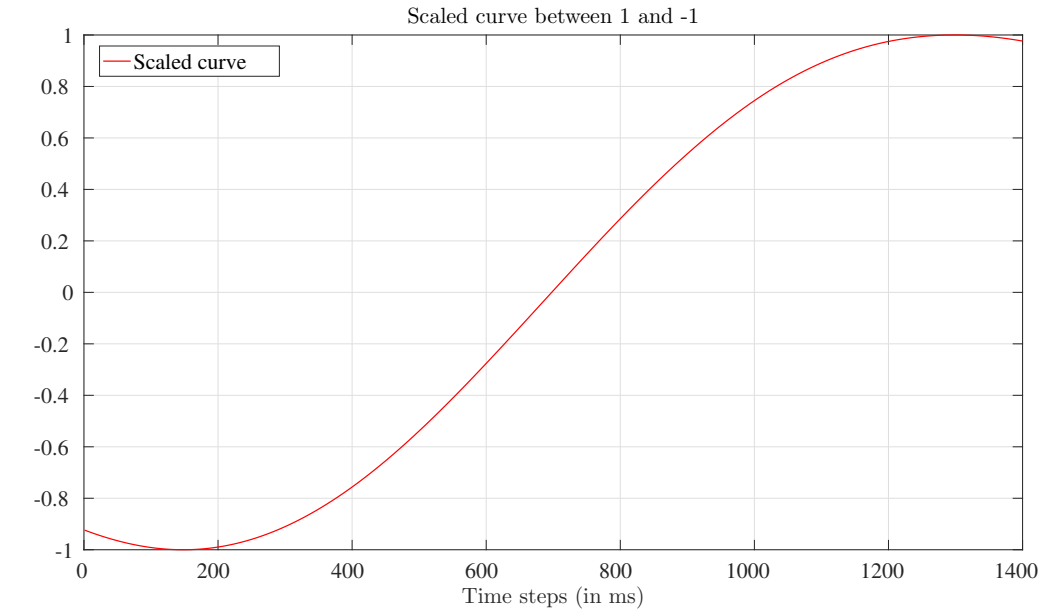

Figure 9: *Scaled curve between -1 and 1.*

5. Draw a linear function (with a slope $1/6$) so as to allow a time-shift of maximum $200ms$ at 1200^{th} time-step. It allows $200ms$ constant shift from 1200^{th} to 1400^{th} time-step. Figure 10 shows the linear function.

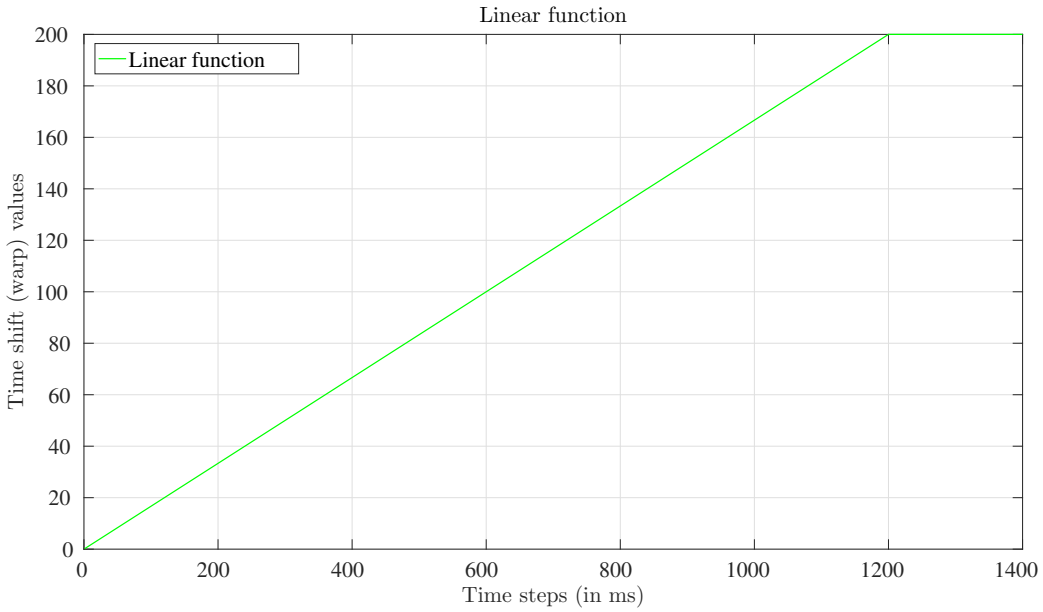

Figure 10: *Linear function. This linear function is multiplied with the scaled curve (figure 9) to get warp defining curve.*

6. Multiply the scaled curve (from step 4) with the linear function (from step 5) to get warp defining curve for each epoch. All the channels of an epoch are warped (or shifted) using the same warp

defining curve. Each warp defining curve has a size $R^{1\times1400}$. The value at each of these 1400 positions depict a shift value for each time position of an epoch. Figure 11 shows the warp defining curve for an epoch.

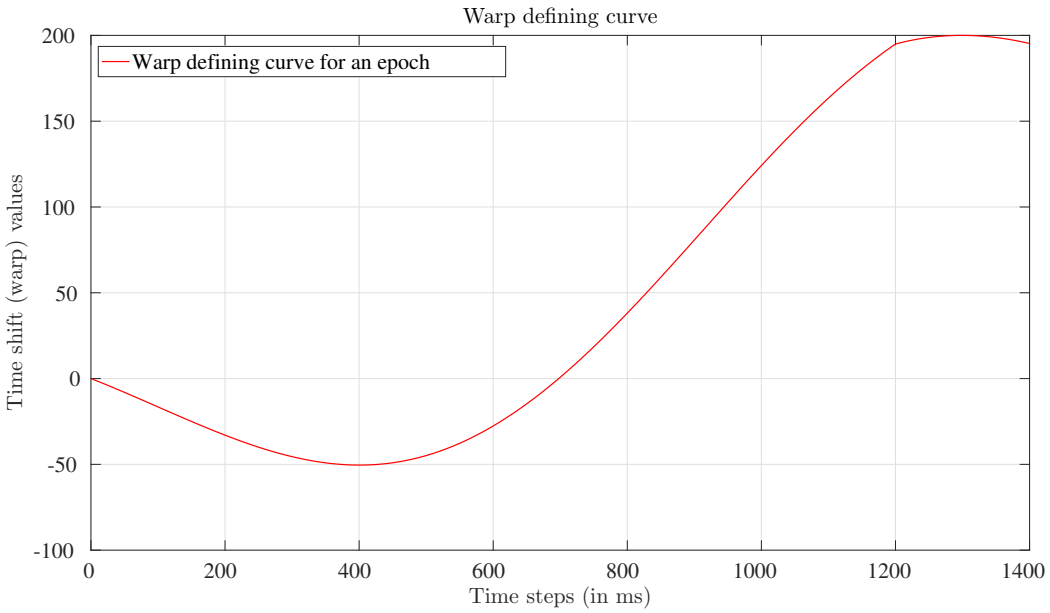

Figure 11: *Warp defining curve.*

7. Now, create a matrix with $14,000$ time-steps (oversample) and 63 channels ($R^{14,000\times63}$). Generate new indices for the original epoch in this matrix using equation 4. 10 is multiplied because we oversample the original epoch (which is $1400ms$ long) to $14,000ms$.

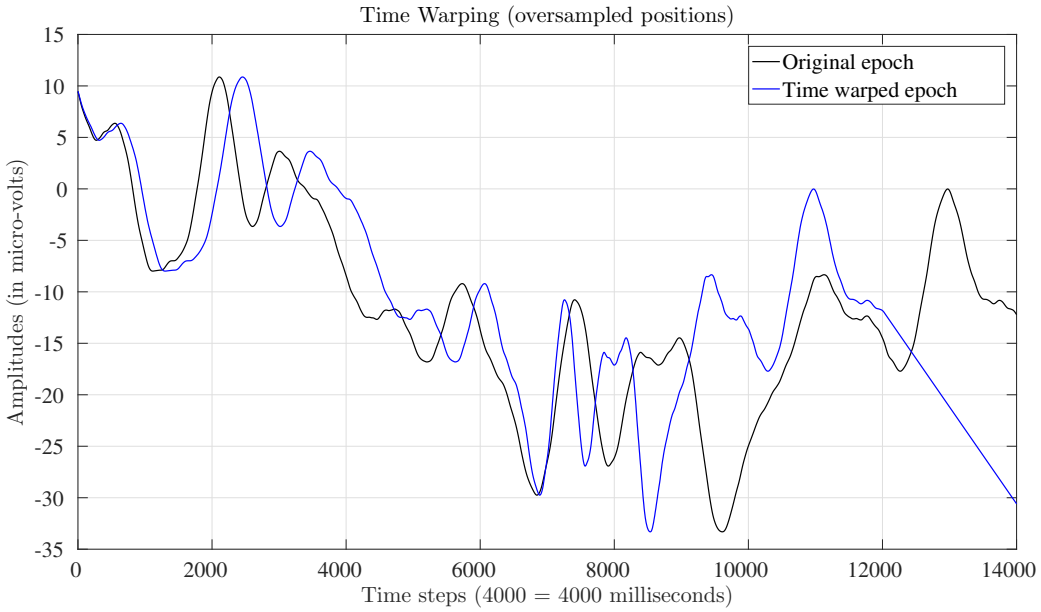

Figure 12: *Time warping (for a channel) in oversampled positions.*

$$index^{new} = 10 \cdot index^{original} \tag{4}$$

8. Place the original amplitudes in this oversampled matrix at their respective new positions (from equation 4). For example, an amplitude value at $1200ms$ in the original epoch is placed at $12,000ms$ in the new oversampled matrix for all channels. Most of the values in the oversampled matrix will be empty (*NaN* - not a number) as only 1400 values are filled out of $14,000$ available indices (for all channels) in the oversampled matrix. The usage of *NaN* helps while interpolating (comes later in the steps) to find the unfilled positions (indices) in the oversampled matrix so that they can be filled with interpolated values.

9. Multiply warp defining curve by 10 after rounding it off to 1 digit after decimal and use it to shift the positions in the oversampled matrix. The new positions of amplitudes are calculated by subtracting the shift values from the original positions in the oversampled matrix.

$$position^{new} = position^{old} - shift \tag{5}$$

   If the value of shift is positive at any time-step, the amplitude value is shifted towards the left (subtraction) and if negative, it is shifted towards the right (addition).

10. Perform shifting for all the time-steps and channels. For each epoch ($R^{1400X63}$), the same warp defining curve is used to shift all the channels.

11. Use linear interpolation to fill in values at *NaN* (not a number) positions. This is an oversampled time-shifted epoch. Figure 12 shows the oversampled original and time-shifted epochs with $14,000$ time-steps.

12. Downsample the time-shifted epoch from $10,000Hz$ to $1000Hz$. This is the time-shifted epoch from the original epoch. Figure 13 shows an original and time-shifted epoch. The horizontal arrows denote the shift directions (forward or backward) and the length of these arrows depict the magnitude of time-shifts. Figure 12 and 13 show this downsampling from $10,000Hz$ to $1000Hz$.

13. Downsample again the original, novel and test epochs from $1000Hz$ to $100Hz$. This is done to be consistent with the other data augmentation strategies as they all use epochs sampled at $100Hz$.

14. Generate novel epochs using original ones following this strategy and add them back to original epochs to enhance the training set.

Elaborating the algorithm above, we generate a large number of normally distributed random samples proportional to the product of the number of novel epochs wanted and the length of time-steps of each epoch ($1400ms$ in this case). Some offset is added so as to avoid the initial parts (low in magnitude) of the filtered and scaled curve. Using these samples, a low frequency ($0.2Hz - 0.5Hz$) curve is filtered out using a low-pass filter at a sampling frequency of $1000Hz$ and filter-order 8. This sampling frequency is kept same ($1000Hz$) as the sampling frequency at which user's data is extracted. A section of length $1400ms$ from this curve is reserved for each original epoch we want to time-shift to create a novel (time-shifted) epoch.

Our aim is to allow a maximum time-shift of $200ms$ at 1200^{th} time-step and $200ms$ beyond 1200^{th} time-step. We scale the filtered curve between 1 and -1. Then, we generate a linear function with a slope of $1/6$ (which allows $200ms$ shift at 1200^{th} time-step). Beyond 1200^{th} time-step, it contains a constant value of $200ms$. Sections of length $1400ms$ are cut out from the filtered and scaled curve. Each section is multiplied by the linear function to generate warp defining curve. For each epoch, there is a different warp defining curve.

We have our shift values as the warp defining curve and an original epoch. Now, we can take an epoch ($R^{1400\times63}$) and a warp defining curve ($1400ms$ long) to do the time shifting. But, we need to shift the matrix positions (which are integers) to the left, right or remain at the same position depending on the values present in the warp defining curve (which are real numbers). In order to shift the indices in the matrix, the warp defining curve must contain integers instead of real numbers. To solve this problem,

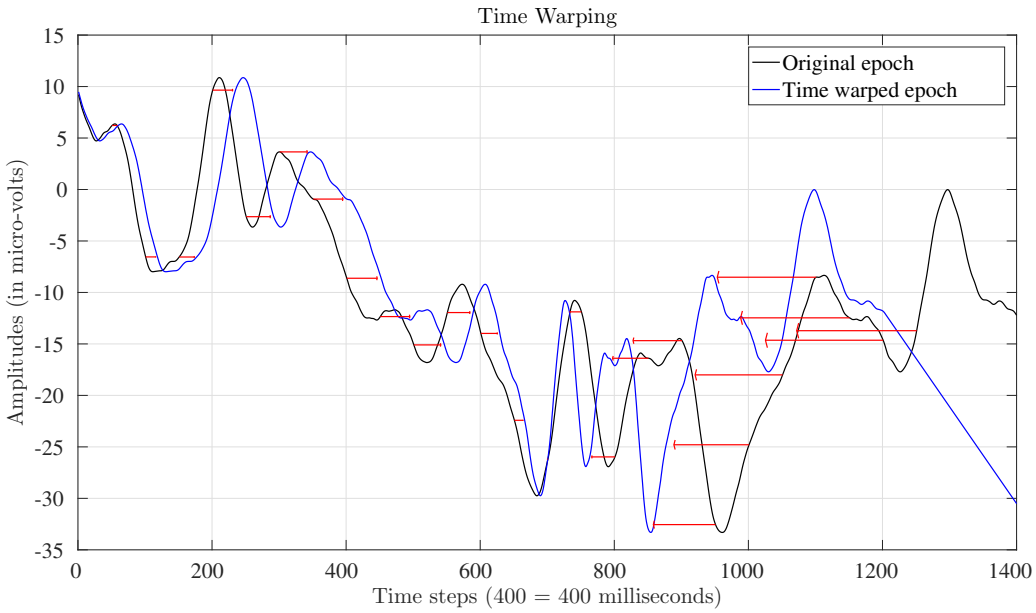

Figure 13: *Time warping (for a channel).*

we work in an oversampled matrix. We can easily understand it by looking at figure 14. Here we have an array with 10 items with their respective positions shown just above. Now, we create an array with 10 times the size of the original array. Then, we fill the oversampled array using the original one by following the below equation:

$$oversampled\_array[10 * position] = original\_array[position] \tag{6}$$

1	2	3	4	…	6	…	8	...	10
-0.34	0.2	0.5	0.6	0.9	1.2	1.7	2.3	2.7	3.0

Original array

1	…..	10	…..	20	…..	30	…..	40	….
NaN	NaN	-0.34	NaN	0.2	NaN	0.5	NaN	0.6	NaN

Oversampled array

Figure 14: *Oversampling.*

To work with a better precision, we:

- Oversample the original epoch to $10,000Hz$. Now, it contains $14,000$ time-steps instead of 1400.
- Round off the real values of the warp defining curve up to 1 digit precision after decimal so that after multiplication by 10, they become integers. Now, the shifting gets easy as the original positions in the overampled matrix and the shift values in the warp defining curve are both integers.

Using these integer shift values from the warp defining curve, we shift the indices of the amplitude values in the oversampled matrix. All the channels of an epoch are shifted using the same warp defining

curve. The negative values in the warp defining curve do shifting to the right, positive to the left and 0 to the same position.

We can understand this idea using this example - let's have a shift value (a real number) from warp defining curve as 66.72 for the time-step 870*ms* (figure 11) for an epoch. In the oversampled matrix, these values become 667 and 8700 (both are multiplied by 10). To time-shift the position, we use equation 5 which gives the new position as $8700 - 667 = 8033$. Hence, the amplitude value present at 8700 will shift to a new position at 8033 in the oversampled matrix. We see left shift because our shift value is positive (figure 12). This is done for all the channels and time-steps for an epoch.

Now we have shifted all the positions in the oversampled matrix using the warp defining curve. But, our oversampled matrix is sparse containing *NaN* values at most of the positions. To fill these positions, we use linear interpolation. It uses the available values to find values for the *NaN* positions. In some situations when there is no data between $12,000ms$ and $14,000ms$ positions (happens due to the left-shift), the linear interpolation extrapolates to fill in these empty positions. This data loss does not cause any issue because while extracting features, we consider amplitudes only up to 1200*ms*. This extrapolation is visible in the time-shifted epoch (blue color) between 1200*ms* and 1400*ms* (figure 13).

After linear interpolation, the oversampled matrix has values for all the positions ($R^{14,000\times 63}$) in an epoch. We downsample it to $1000Hz$ by taking every $10th$ value. The generated epoch is the time-shifted version of an original epoch. Now, we have original and time-shifted versions of epochs sampled at $1000Hz$. We downsample these to $100Hz$ to be consistent with the other data augmentation strategies. Similarly, we downsample the test epochs as well in the same way. This downsampling is done before the features are extracted.

3.3 Principal component analysis (PCA) modulation

PCA is used to project multi-dimensional data into a lower dimensional representation while retaining the most of the information present in the data. An epoch of the ERP data has the dimensionality 140×63 where 140 denotes total time-steps (each equal to 10*ms*) and 63 are the channels. Using PCA, each epoch is transformed into its lower dimensional space having 30 components (also called principal components) [15] and at the same time, more than 99% of the variance in the epoch is retained. Hence, the dimensionality of the epoch changes to 140×30. In this new space of principal components, we increase and decrease the power of each of these 30 principal components of epochs to generate new epochs in the same PCA space. The novel epochs are then projected back to the original dimensionality (140×63). To increase and decrease the power of each component, we use factors from the gamma distribution.

3.3.1 Gamma distribution

The histogram of the gamma distributed random factors is shown in figure 15. The gamma distribution has two parameters - *shape* and *scale*. The *shape* parameter alters the shape of the distribution. For $shape = 1$, the distribution gets an exponential nature. With $shape > 1$, the distribution acquires a skewed nature. The other parameter is *scale* which stretches or compresses the range of the distribution.

$$E[x_i] = shape \cdot scale \tag{7}$$

where $E[x_i]$ is the expected value (a scalar) of the gamma-distributed random numbers.

$$s^{new} = \gamma \cdot s^{old} \tag{8}$$

where s^{new} is the new value of the s^{old} source multiplied with γ (random factor following the gamma distribution). In our experiments, each epoch is decomposed to 30 principal components (and sources for ICA modulation). Each principal component or source is $R^{140\times 1}$ where 140 is the time-steps, each time-step being 10*ms* long.

There are two reasons for using this distribution to generate random factors:

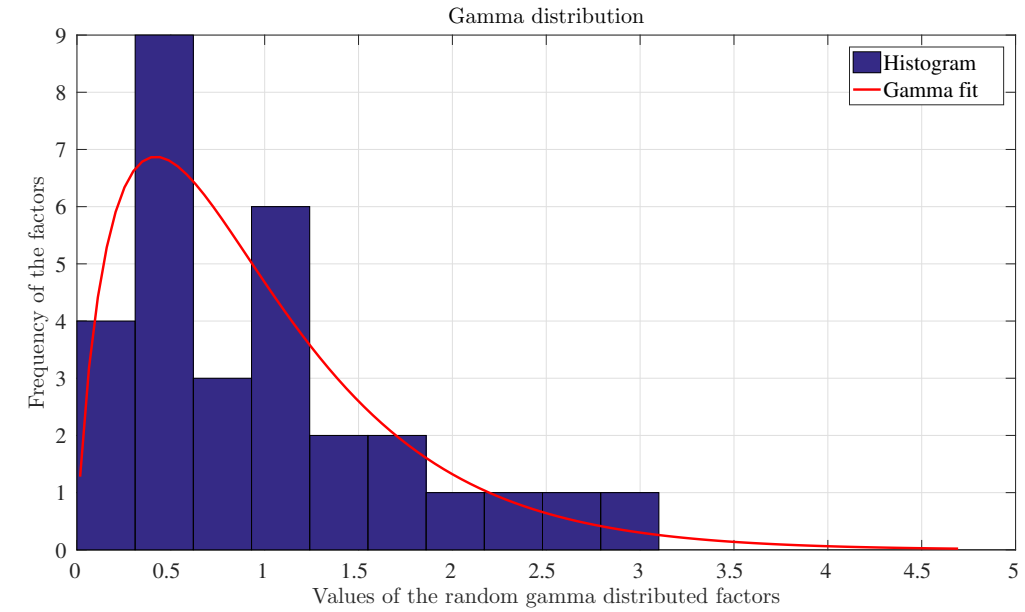

Figure 15: *Histogram for gamma distribution random factors (30).*

- The expected value of these random factors is 1.
- The number of factors which are smaller than 1 is more than the number of factors which are greater than 1.

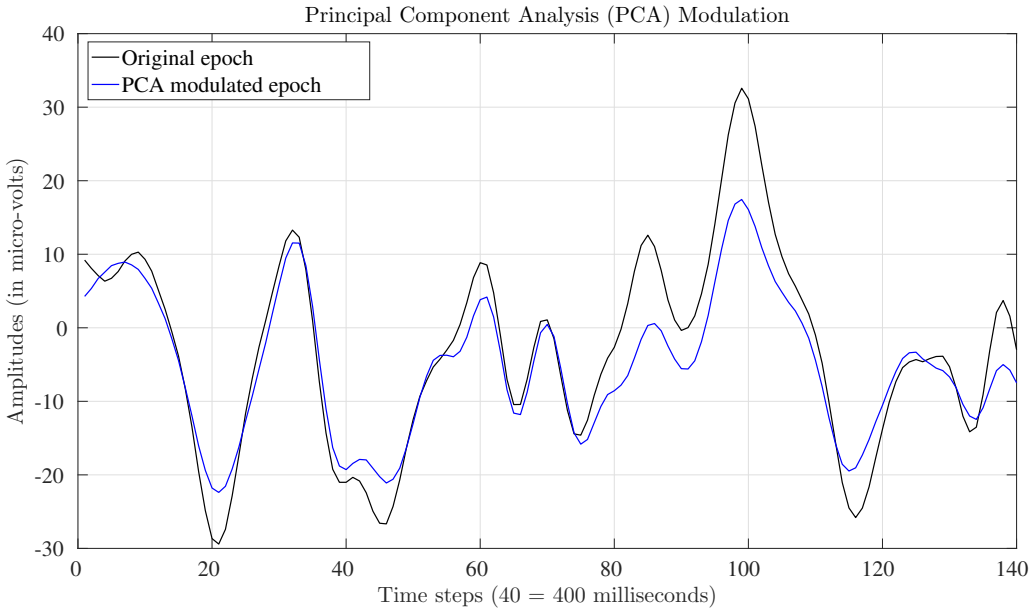

Figure 16: *PCA modulation (for a channel).*

These two properties ensure that when we increase and decrease the power of the components of an epoch, the expected power (of principal components in PCA modulation and of sources in ICA modulation) for an epoch remains the same. Essentially, we weigh the components using these random factors. This distribution is used in ICA modulation as well for the same reasons to modulate (increase and

decrease) the power of sources. Figure 16 shows an original epoch and PCA modulated epoch.

3.4 Independent component analysis (ICA) modulation

The data recorded over scalp is a mixture of neural signals and artifacts. For capturing scalp potentials, multiple electrodes are fixed on the scalp assuming that each one would record data corresponding to a source inside the brain. But, in practice, the data recorded by each electrode exhibit linear relationship due to an electrical impulse spread (volume conduction) over the scalp and the proximity of electrodes to each other.

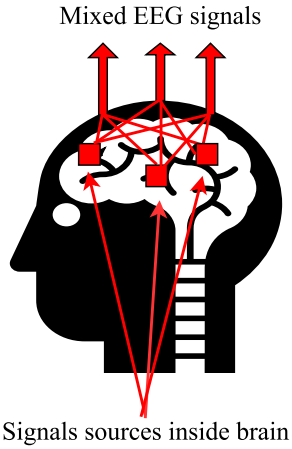

Figure 17: *Mixing of brain signals from multiple sources.*

From picture 17, it can be seen that the data captured at the scalp using electrodes is a mixture coming from multiple sources inside the brain. In order to separate this data into its sources, independent component analysis (ICA) is applied. This algorithm is one of the techniques used for blind source separation (BSS). It is termed as BSS because we have the recorded data but are unaware of the actual sources as well as the mixing coefficients of these sources. Both of these are estimated using the recorded data. We would now refer to epochs as before instead of data or signals.

$$s = W \cdot x \tag{9}$$

where s is a set of sources ($R^{140\times30\times3240}$) for all epochs, W is the demixing matrix ($R^{63\times30}$) and x is a set of epochs ($R^{140\times63\times3240}$) for 3240 epochs, 30 sources, 63 scalp channels and 140 time-steps. While we separate the epochs using ICA, we assume that they are linearly mixed and the sources are independent of each other. The cross-covariance matrix of the sources is a diagonal matrix as the cross-covariance of each source with all other sources is close to zero. Before applying ICA on these epochs, a dimensionality reduction algorithm, principal component analysis (PCA), is applied (from the previous section). The epochs are decomposed into a smaller set of components which explains most of the variance (approximately 99%). Using these lower dimensional epochs, actual sources are computed using ICA. The original epochs are transformed into 30 sources. Now, each epoch has a dimensionality $R^{140\times30}$.

So far, we have discussed how using the PCA and then ICA reduces the epochs into their source components. We come back to our data augmentation strategy which makes use of the epochs in source space to create novel epochs (in the same space). For generating these novel epochs, we apply weights to all the sources by multiplying with random factors taken from the gamma distribution (described in the PCA modulation section).

We alter all the sources of an original epoch to create a novel epoch. The novel epochs are still in the source space (30 sources). In order to project them back to the original space, mixing matrix ($R^{30\times63}$) is multiplied to the novel epochs. Now, as we have our novel epochs in the original space (63 channels),

we look for artifactual epochs in the entire set of novel epochs. The presence of artifactual epoch(s) cannot be ruled out because we increase the strength of some of the sources by multiplying them with factors greater than 1. This results in higher amplitudes for some sources. Assuming these sources to be artifactual in nature, we discard such novel epochs altogether even if one channel (out of 63) shows strength greater than a threshold. Figure 18 shows the ICA modulation. The black line is an original epoch and the blue line shows the ICA modulated novel epoch.

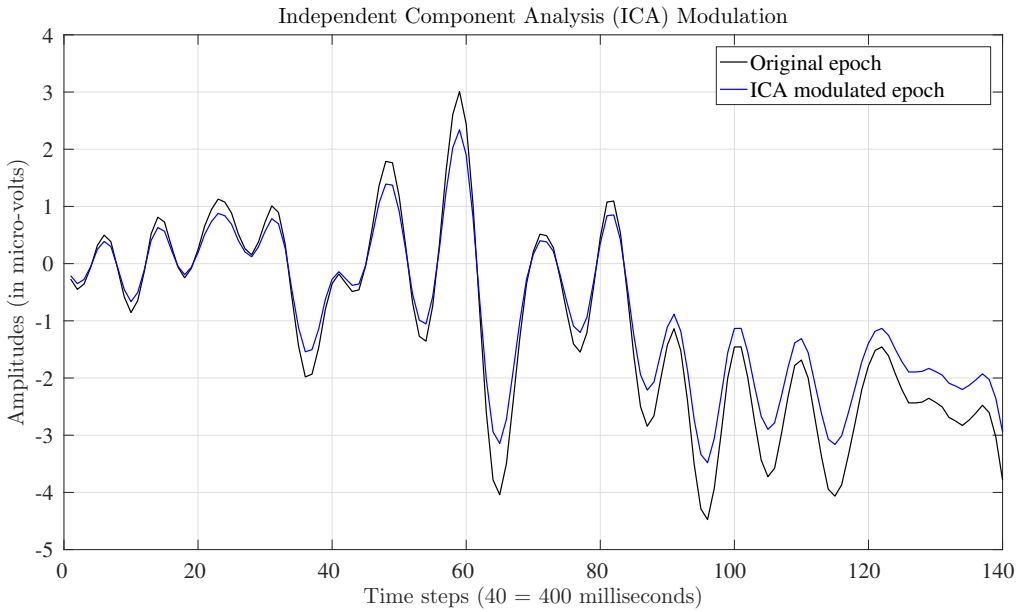

Figure 18: *ICA Modulation in source space (for a source).*

4 Regularised linear discriminant analysis (RLDA) classifier

RLDA is a linear classifier which separates two classes by a hyperplane:

$$w^T \cdot x + b = 0 \tag{10}$$

where w is the projection vector, x is an input vector (feature vector) and b is the bias. The output class is obtained by $sign(w^T \cdot x + b)$. The weight vector is given by:

$$w = \sum_c^{-1} \cdot (\mu_2 - \mu_1) \tag{11}$$

where μ_1 and μ_2 are the mean vectors of the respective classes, $\sum_c = 0.5 \cdot (\sum_1 + \sum_2)$ and $\sum_1$ and $\sum_2$ are the respective covariance matrices for the two classes. Since there are multiple channels and each channel has multiple features, the covariance matrix estimation can become wrong when the data is not sufficient compared to its dimensionality. Since LDA needs accurate estimate of the covariance matrix to predict the classes of test data in a reliable manner, it becomes important to devise a way to accurately estimate this matrix. To solve this issue, a shrinkage parameter (λ) is used. Using this parameter, the covariance is calculated as:

$$\bar{\sum}(\lambda) = (1-\lambda)\sum_c + \lambda \nu I \tag{12}$$

where λ is the tuning parameter varying in the range $[0,1]$ and $\nu = trace(\sum_c)/d$, d is the dimensionality of feature space. ν is also called a scaling parameter. When $\lambda = 1$, the covariance matrix is diagonal and when $\lambda = 0$, it is the covariance matrix estimated from the data [16].

5 Experimental setup

In the following section, we discuss cross-validation, feature extraction and grand average performance for each data augmentation strategy for 10 users.

5.1 Training and validation

In our experiment, we apply four data augmentation strategies on the ERP data of 10 users. The epochs of each user are divided into training and test sets using 5-fold cross-validation. In general, k-fold cross-validation divides the data into k partitions of equal size. Out of these k partitions, one is used for testing and the rest is used for training. This is repeated for k times so that each partition is used as the test set once.

We apply our data augmentation strategies only on the training epochs. We take out specific numbers of training epochs (say 500) and use only these epochs to generate (say 10,000) novel epochs. These novel epochs are added to the 500 original training epochs to get an enhanced training set. The 500 original training epochs are chosen randomly from the complete set of original training epochs (2592) because the epochs are arranged chronologically and the random 500 is assumed to contain epochs from all parts of the complete original training epochs. The novel epochs are generated in small to large numbers (100, 200, 400, 800, 1600, 3200 and 10,000) compared to the number of original training epochs. As we deal with randomnesses like random amplitude modulation factors or random selection of training epochs, it is necessary to run the experiment multiple times and take an average of the classification accuracies to stabilise the classification accuracy values. Thus, for each size of original training and novel epochs, 100 runs are performed. Inside each run, 5-fold cross-validation is performed. In a similar way 1100, 1800, 2592 original training epochs are used in the different iterations of the experiment to generate novel epochs in the size mentioned above. $Algorithm1$ displays the pseudo-code for the general data augmentation strategy. For the complete code, please go to github[3].

5.2 Feature extraction

Features are the average potentials computed over different time intervals for all channels. These time intervals like 100ms-180ms are carefully chosen to have the most discriminative features. In our experiment, there are 8 such time intervals - 100ms-180ms, 190ms-300ms, 301ms-450ms, 450ms-560ms, 561ms-700ms, 701ms-850ms, 851ms-1000ms and 1001ms-1200ms. Within these intervals, the scalp potentials are averaged to form features across all the channels which reduce the dimensionality from $R^{140\times63}$ to $R^{8\times63}$ for an epoch. The features are extracted separately for training and test epochs.

5.3 Grand average performance

We evaluate the data augmentation strategies on the ERP data from 10 users. For each user, we get classification results as a matrix ($R^{4\times8}$) where 4 is the number of different sizes of original training epochs (e.g. 500, 1100, 1800 and 2592) and 8 is the number of different sizes of novel epochs (e.g. 0, 100, 200, 400, 800, 1600, 3200, 10000). In order to measure an average classification performance over 10 users, we sum up the classification results (each of size $R^{4\times8}$) of each user and divide the sum by 10.

[3] `https://github.com/bsdlab/aphasia_analysis/tree/data_augmentation/analysis/example`

Algorithm 1 Data augmentation

function AUGMENTEPOCHS($user\_num$) ▷ Take any value in (1-10) for user_num.
  $epo = load\_User\_Epochs(user\_num)$ ▷ Load user's epochs.
  $augment\_data\_size = [0, 100, 200, 400, 800, 1600, 3200, 10000]$ ▷ Different sizes of novel epochs to be created.
  $original\_data\_size = [500, 1100, 1800, 2592]$ ▷ Different sizes of original training epochs to be used to create novel epochs.
  $experiment\_runs = 100$ ▷ A number of runs.
  $folds\_cv = 5$ ▷ A number of folds for cross-validation.
  for $original\_data\_size = 500$ to 2592 **do**
    for $augment\_data\_size = 0$ to $10,000$ **do**
      for $run = 1$ to $experiment\_runs$ **do**
        for $k = 1$ to $folds\_cv$ **do**
          $[epo\_tr, epo\_te] = select\_Training\_Test\_Epochs(epo)$ ▷ Extract training and test epochs.
          $random\_idx = randperm(epo\_tr\_size, original\_data\_size)$ ▷ Generate random epochs indices from all the training epochs to a size of epochs needed. For example, 500.
          $epo\_tr\_selected\_size = proc\_selectEpochs(epo\_tr, random\_idx)$ ▷ Select a set of random epochs (say 500) from all training epochs
          $training\_epochs\_corrected = reject\_Artifacts(epo\_tr\_selected\_size, 60)$ ▷ Reject artifacts from the training epochs. 60 micro-volts is the threshold value for rejecting epochs.
          if $augment\_data\_size > 0$ **then**
            $novel\_epochs = create\_Novel\_Epochs(epo\_tr\_selected\_size, augment\_data\_size)$ ▷ Create novel epochs using original ones following a data augmentation strategy. These novel epochs are already corrected for artifacts.
            $training\_epochs\_corrected = append\_Epochs(novel\_epochs, training\_epochs\_corrected)$ ▷ Append the portion of original training and novel epochs. These epochs are artifact corrected.
          end if
          $test\_epochs\_corrected = reject\_Artifacts(epo\_te, 60)$ ▷ Reject artifacts from test epochs.
          $features\_training = extract\_Features(training\_epochs\_corrected)$ ▷ Extract training epochs features.
          $features\_test = extract\_Features(test\_epochs\_corrected)$ ▷ Extract test epochs features.
          $trained\_model = train\_Classifier(features\_training, @classifier)$ ▷ Train a classifier using training features.
          $classification\_accuracy\_cv[k] = evaluate\_Model(features\_test, trained\_model)$ ▷ Get classification accuracy on the test epochs using a model trained on the training epochs.
        end for
        $mean\_cv\_accuracy[run] = mean(classification\_accuracy\_cv)$ ▷ Get mean accuracy for 5-fold cross-validation for one run
      end for
      $mean\_run\_accuracy = mean(mean\_cv\_accuracy)$ ▷ Get mean classification accuracy for 100 runs for one size of original training and novel epochs
    end for
    $all\_aug\_size\_accuracy$ ▷ Get classification accuracy for all augmented sizes (0, 100,..., 10000) for one size of original training epochs (say 1800)
  end for
  $classification\_accuracy$ ▷ Get classification accuracy for all sizes of original and novel training epochs
end function

6 Results and analysis

6.1 ERP plots

The ERP plots are used to visualise the mean of epochs over multiple channels for target and non-target classes in the entire time interval from $-200ms$ to $1200ms$. They also depict the evolution of patterns on the scalp (from second until the last row of the figures 19 and 20) showing the spread of potential in the selected (shaded columns in the first row of the plots) time intervals. There are 8 such time intervals (also used for feature extraction) corresponding to the 8 scalp plots for target and non-target classes. The plots show two channels Cz (central channel on the scalp) and $CP5$ in the top section of the plots. Figure 19 shows the ERP plot for original training epochs and Figure 20 shows the ERP plot for novel training epochs (generated by using ICA modulation of the original epochs in Figure 19). Approximately, the same number of epochs (2100) have been used to generate these. They show differences in potential spread for corresponding scalp patterns in target and non-target classes.

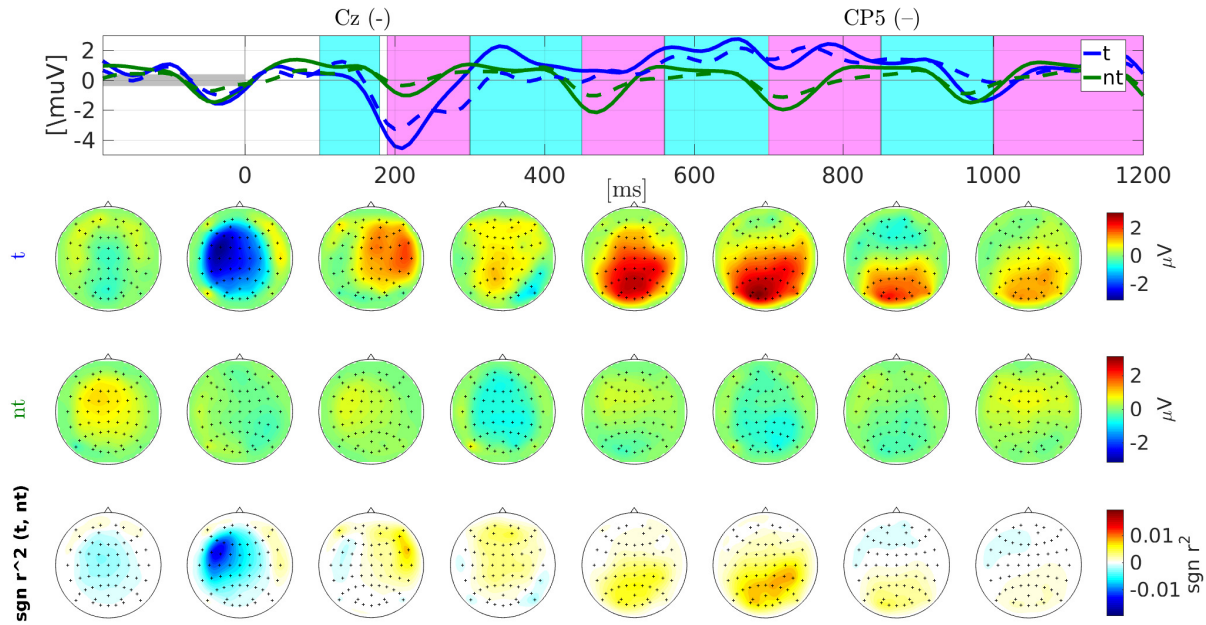

Figure 19: *ERP plot for original training epochs (2165 epochs) for user 1.*

6.2 Data augmentation by amplitude modulation

We generate novel epochs by modulating amplitudes of original ones using factors chosen randomly from the range $[0.9 - 1.1]$. These factors follow a uniform distribution and the expected value of these random factors is close to 1. Figure 21 shows the grand average performance for this strategy applied to the epochs from 10 users. Each line plot corresponds to one size of original training epochs. For example, the black line plot takes 500 random epochs from the complete set of original training epochs and uses them for generating novel epochs in different sizes (100, 200 and so on). Further, we increase the number of original training epochs until we use all the available training epochs to generate novel ones. Each size of original training epochs is shown in different color. Each line plot is an average over 10 users. This method of generating grand average of classification results we will use for other data augmentation strategies as well.

From bottom to top in figure 21, the performance increases because at every step, the number of original training epochs is increased. But, from left to right, the performance decreases for all line plots. It indicates that this strategy of data augmentation does not improve the generalisation ability of the classifier. For example, when we use 2592 original training epochs (red line plot) for generating

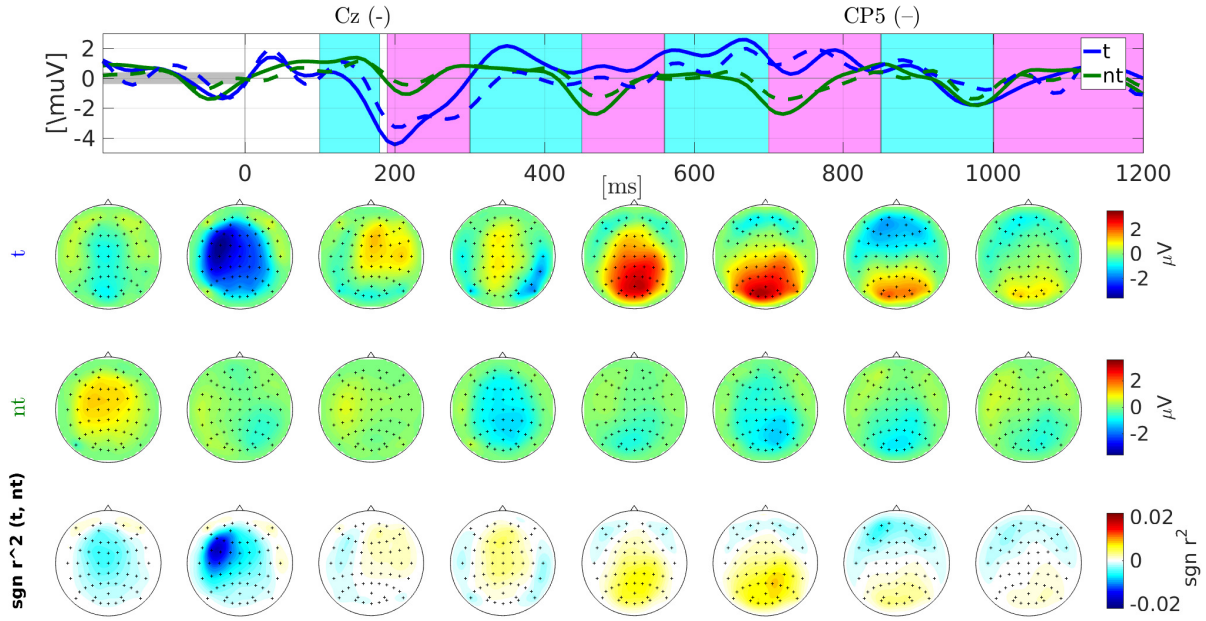

Figure 20: *ERP plot for novel epochs generated by ICA modulation (2100 epochs) for user 1.*

novel ones, overall classification accuracy goes down by 0.7% from no novel epochs (0 on the x-axis) to 10,000 novel epochs. For 500 original training epochs, this drop is more severe (over 2%). The results are averaged over 100 runs of the experiment for each user.

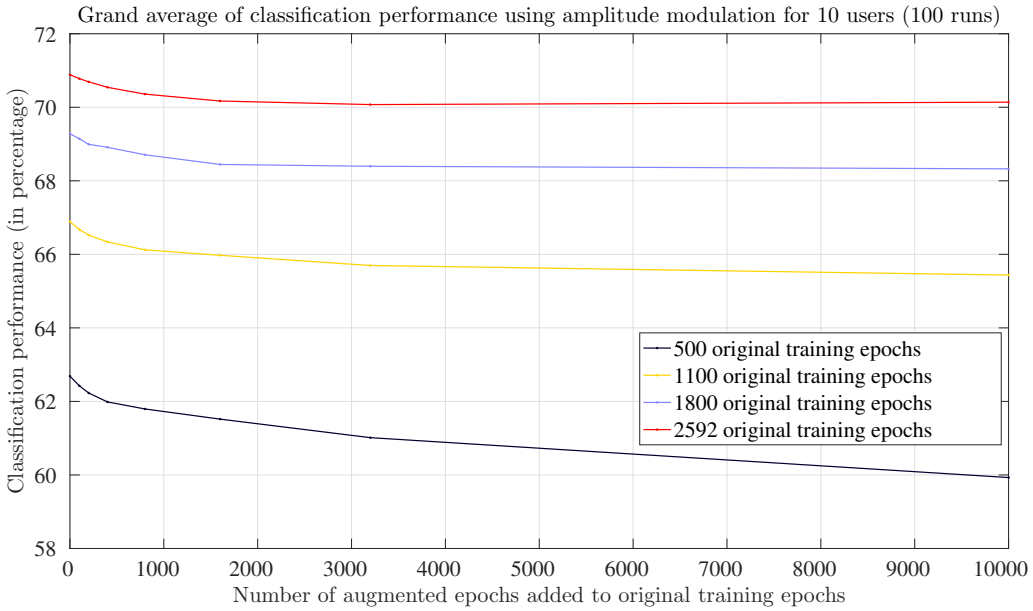

Figure 21: *Grand average performance using amplitude modulation.*

6.2.1 Case study: performance of 3200 novel epochs using different sizes of original training epochs

Figure 22 shows a scatter plot comparing the classification performances by training on only original epochs (500, 1100 and so on) and the original and novel epochs (3200) using amplitude modulation data

augmentation strategy for all 10 users. The horizontal axis shows the classification performance using X (500, 1100, 1800, 2592) number of original training epochs and the vertical axis shows classification performance by training on X number of original training epochs and 3200 novel epochs. The different original sizes used to generate 3200 novel epochs are shown in different colors. For example, blue circles denote the performances on 1800 original training epochs (x-axis) and 1800 with 3200 novel epochs (y-axis). The results are averaged over 100 runs. We conclude from this figure that this data augmentation strategy does not improve the classification as almost all the circles remain below the blue line. The diagonal (blue line) defines a boundary to show whether the plotted points are significant depending on their positions - above or below the diagonal. If any point is above the diagonal, it is significant. Otherwise, it is not. Being significant means that the classification performance of original training and novel epochs together is better than just using original training epochs.

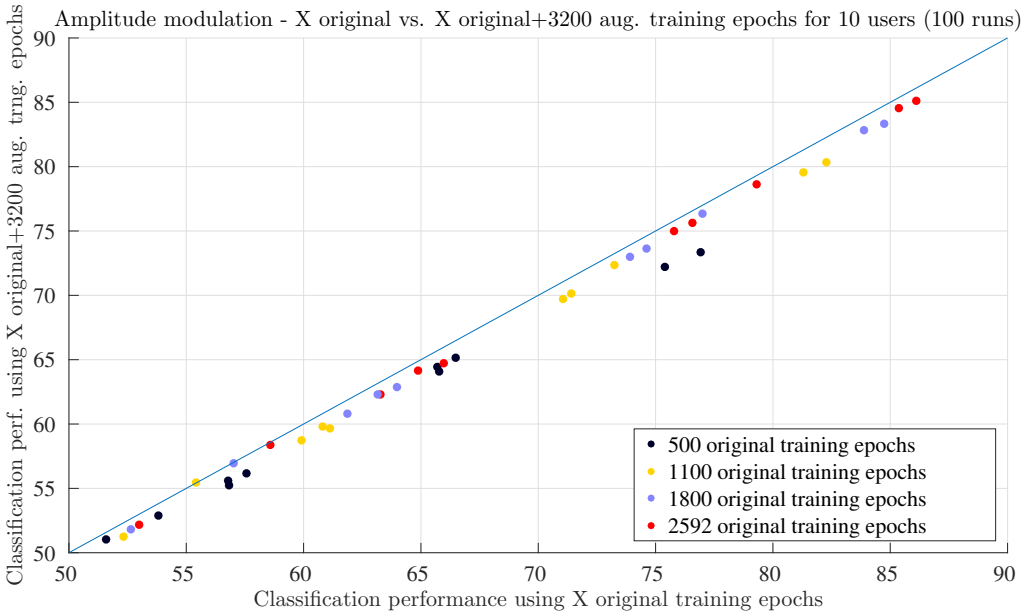

Figure 22: *A scatter plot for amplitude modulation comparing classification of original training epochs (e.g. 500) vs original training (e.g. 500) + augmented (3200) epochs.*

6.2.2 Case study: mean and covariance using original (500) and novel epochs (3200)

In this case study, we specify four conditions using different combinations of mean and covariance of original and novel epochs and each condition gives us a new classifier which is used to evaluate the performance on test epochs. We have these conditions:

- Condition 1: mean and covariance calculated only from original training epochs (original mean and original covariance).
- Condition 2: mean from original training epochs and covariance from novel epochs (original mean and augmented covariance).
- Condition 3: mean from novel epochs and covariance from original training epochs (augmented mean and original covariance).
- Condition 4: mean and covariance calculated only from novel epochs (augmented mean and covariance).

Figure 23 shows a scatter plot which compares the performances of condition 1 versus 2, condition 1 versus 3 and condition 1 versus 4. In this figure, 500 original training epochs are used to create 3200 novel epochs (following amplitude modulation data augmentation strategy) for all 10 users. The classification values are averaged over 100 runs. The horizontal axis in the figure shows the performance using condition 1. The vertical axis shows the performance using conditions 2, 3 and 4. The comparisons (1 versus 2, 1 versus 3 and 1 versus 4) are shown in different colors. For example, red circles show the comparison between condition 1 (original mean and covariance) and condition 4 (augmented mean and covariance). The yellow circles (condition 1 versus 3) perform close to the blue line which means that condition 3 performs almost equally as the original training epochs. The mean of the novel epochs remains close to the mean of original training epochs (which are used to generate novel ones). The other conditions, conditions 2 and 4 do not perform comparably to the original training epochs (condition 1) as they are below the blue line. The performances of condition 2 and 4 overlap as shown by the overlapping red and black circles. The two conditions, 2 and 4, have augmented covariance in common and original and augmented mean respectively. The similar performances of these two conditions also suggest that the original mean (500 epochs) is close to the mean of novel (3200) epochs.

Please note that the different conditions described above will remain same for other data augmentation strategies studied in this work for this case study.

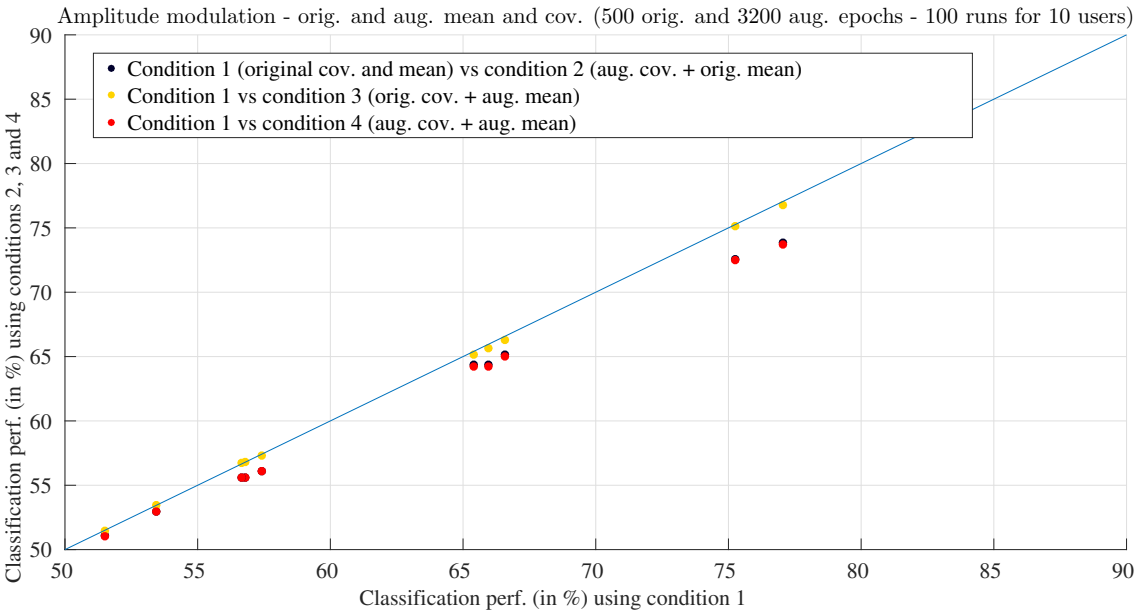

Figure 23: *A scatter plot for amplitude modulation showing comparison of classification performances using different combinations of mean and covariance of original (500) and novel (3200) epochs.*

6.3 Data augmentation by time warping

This strategy brings about time-shifts in the original epochs to generate novel epochs. A maximum of 200*ms* time-shift (either backward or forward in time) is allowed. The grand average performance from figure 24 shows that all the line plots representing different sizes of original training epochs measure a drop of about 1% in classification performance from no augmentation to 10,000 novel epochs (from left to right). For example, the yellow line in the figure represents 1100 original training epochs which are used to create novel epochs. For no augmentation (0 on the x-axis), it shows a performance of around 67% which drops to around 66.2% after adding 10,000 novel epochs.

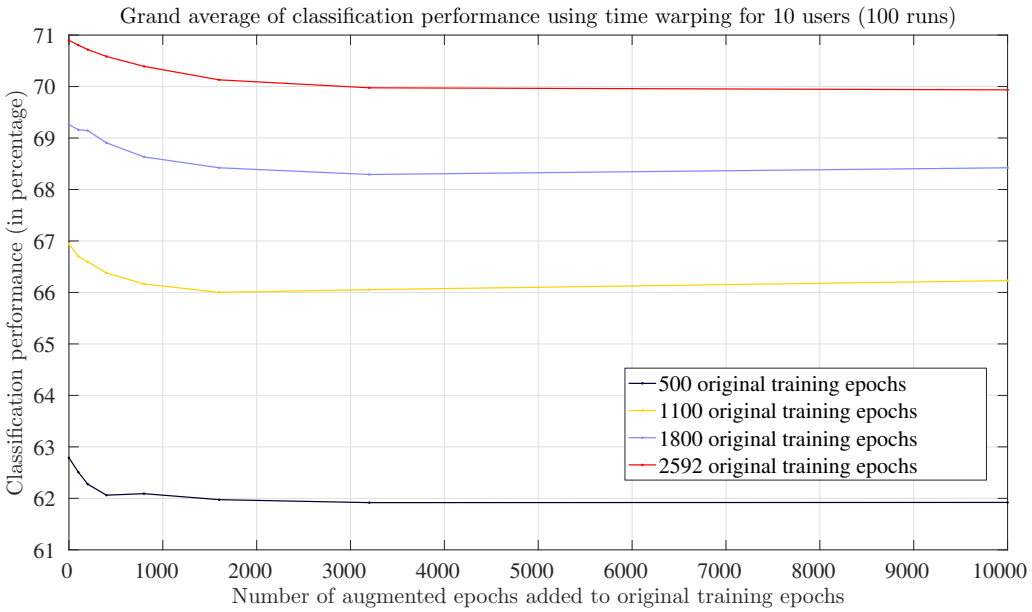

Figure 24: *Grand average performance using time warping.*

6.3.1 Case study: performance of 3200 novel epochs using different sizes of original training epochs

Figure 25 shows a scatter plot which compares the performances of generating 3200 novel epochs using different sizes of original training epochs following the time warping strategy. The plot shows results from all the 10 users and is averaged over 100 runs. Most of the circles remain below the blue line which suggests that generating novel epochs (3200) using any size of original training epochs following this data augmentation strategy is not beneficial. Though some circles measure performance above the blue line (which means they perform better than just using original training epochs), but they do not show any pattern. For example, just 2 users (2 black circles for 500 original training epochs) measure better performance after adding 3200 novel epochs. Only 2 users (2 red circles for 2592 original training epochs) have the performance better than the original training epochs (2592). The other red circles are below the blue line.

6.3.2 Case study: mean and covariance using original (500) and novel epochs (3200)

We can see from the figure 26 that the novel epochs generated by time warping improve the covariance estimate (condition 2) as 5 out of 10 users perform better with novel training epochs compared to the original training epochs. The other 3 users' performances remain same. The performance with the other two combinations, condition 3 (covariance from original training epochs and mean from novel epochs) and condition 4 (covariance and mean from novel epochs) do not beat the performance of just using original training epochs (condition 1) for estimating the covariance and mean. We conclude from the figure 24 that time warping strategy does not improve the classification when we enhance the training epochs with novel epochs. But, this case study suggests that if we just use the covariance estimate from the novel epochs and mean estimate from the original training epochs, we can improve the classification results for 500 original training epochs.

6.4 Data augmentation by PCA modulation

In this idea, we modulate the epochs in PCA space by increasing and decreasing the strength of principal components using random factors from gamma distribution to create novel epochs. Figure 27 shows

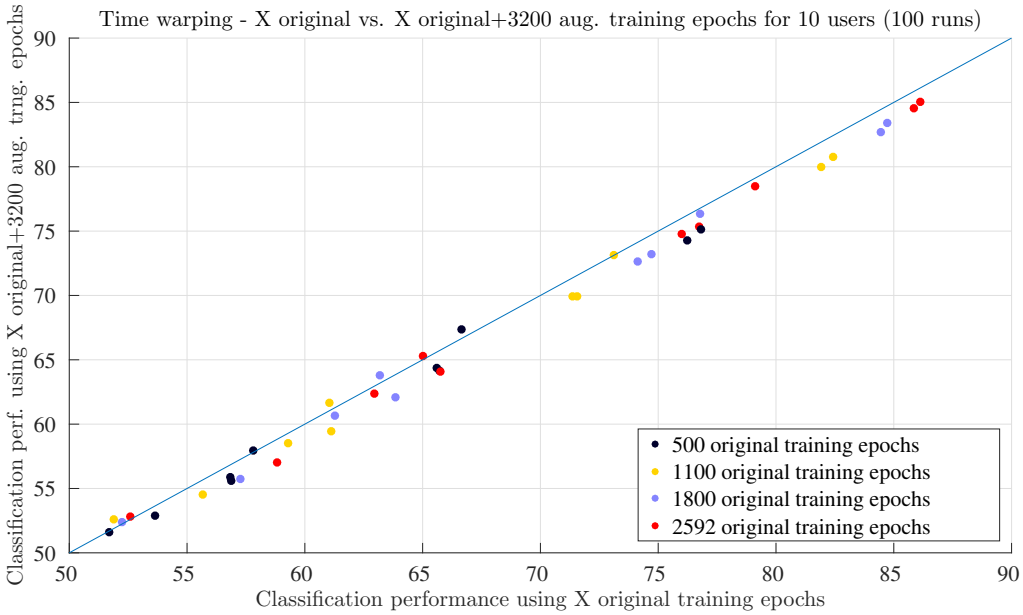

Figure 25: *A scatter plot for time warping comparing classification of original epochs vs original + augmented (3200) epochs.*

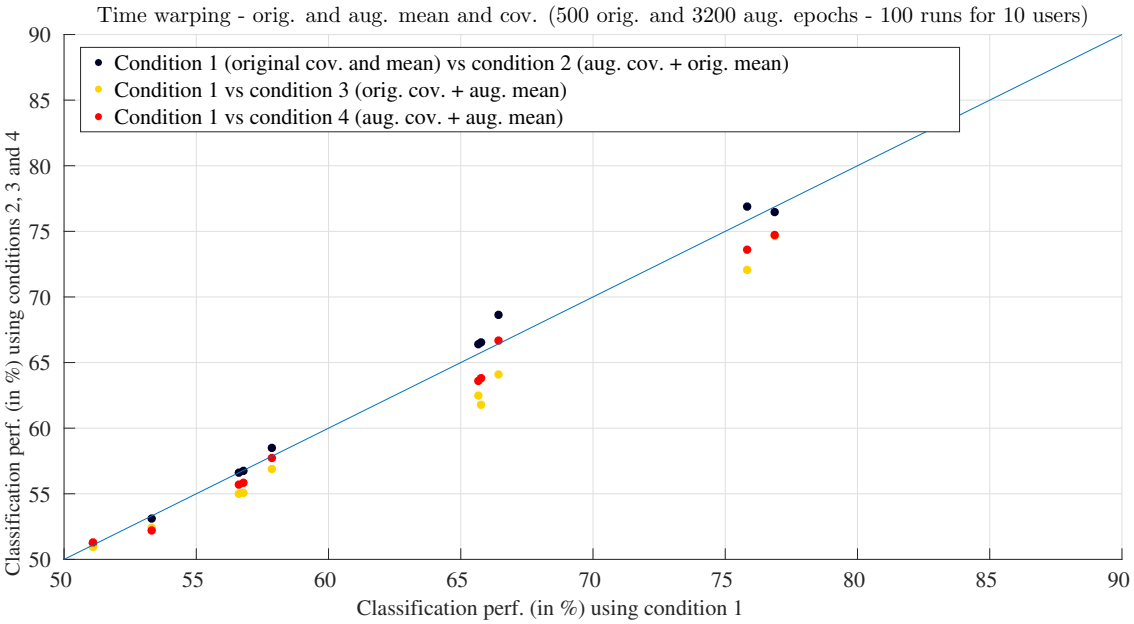

Figure 26: *A scatter plot for time warping showing comparison of classification performances using different combinations of mean and covariance of original (500) and novel (3200) epochs.*

the grand average classification performance for 10 users using PCA modulation strategy. The results are averaged over 100 runs of the experiment. The figure shows that when we use 500 original training epochs to generate novel epochs, the classification performance goes up from around 62.8% to 64% (going from left to right). The line plot with 1100 original training epochs also shows a meagre increase but the other line plots (blue and red) show decline in performance by about 0.5%. The early stages of data augmentation where we add 100 or 200 epochs show sudden peaks, especially for the lower two line plots. The line plots smoothen as we average the results over more number of runs. There is a separate

discussion on it later in the report.

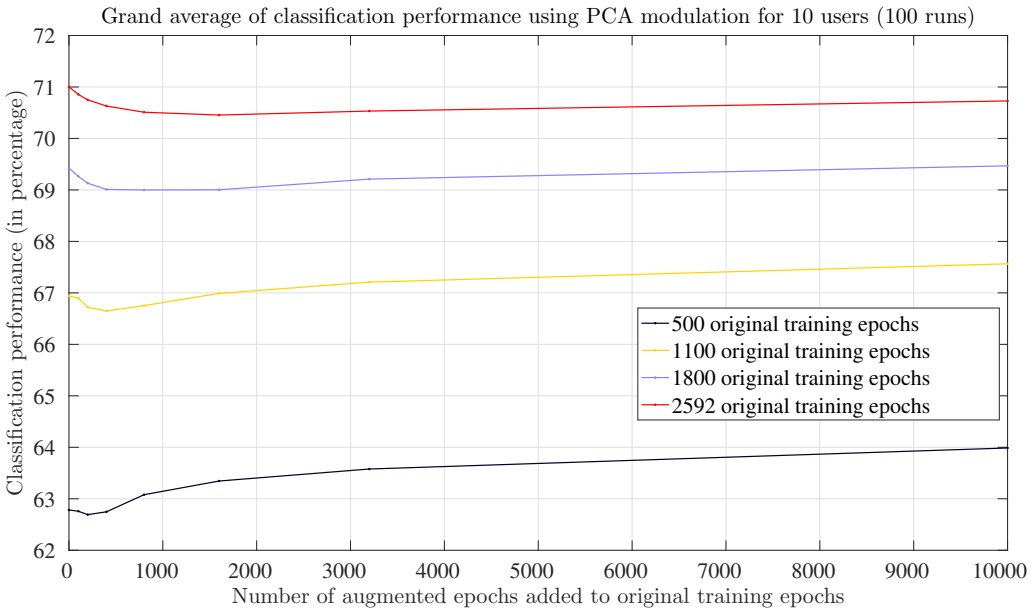

Figure 27: *Grand average performance using PCA modulation.*

6.4.1 Case study: performance of 3200 novel epochs using different sizes of original training epochs

Figure 28 shows a plot comparing the addition of 3200 novel epochs to different sizes of original training epochs for 10 users. The results are averaged over 100 runs. It is clear from the figure that adding novel epochs generated using just 500 original training epochs is beneficial. Most of the black circles (which represent 500 original training epochs) are above the blue line which ascertains that the performance improvement is significant. The usage of 1100 original training epochs is also beneficial but this is restricted to only a few users. The circles belong to other sizes of original training epochs (blue and red) stay either on the blue line or below it.

6.4.2 Case study: mean and covariance using original (500) and novel epochs (3200)

Figure 29 shows a scatter plot depicting whether the mean or covariance or both improve using the novel epochs. In this plot, the condition 4 performs better than the original training epochs (condition 1) as most of the red circles remain above the blue line for many users. This suggests that the covariance and mean estimate together from the novel epochs are better than those from the original training epochs where the red circles are above the blue line. If we see the augmented covariance and mean separately in conditions 1 versus 2 and conditions 1 vs 3, they do not improve upon the original estimates as they measure performance lower compared to the original training epochs.

6.5 Data augmentation by ICA modulation

We modulate the original epochs in source space by increasing and decreasing the strength of the sources using random factors from gamma distribution to generate novel epochs. Figure 30 shows that the grand average performance using 500 original training epochs measures 0.5% increase as we go towards the right (at $10,000$ novel epochs). However, the other line plots register a drop of about 0.5% as we move towards the right (the direction in which more novel epochs are added).

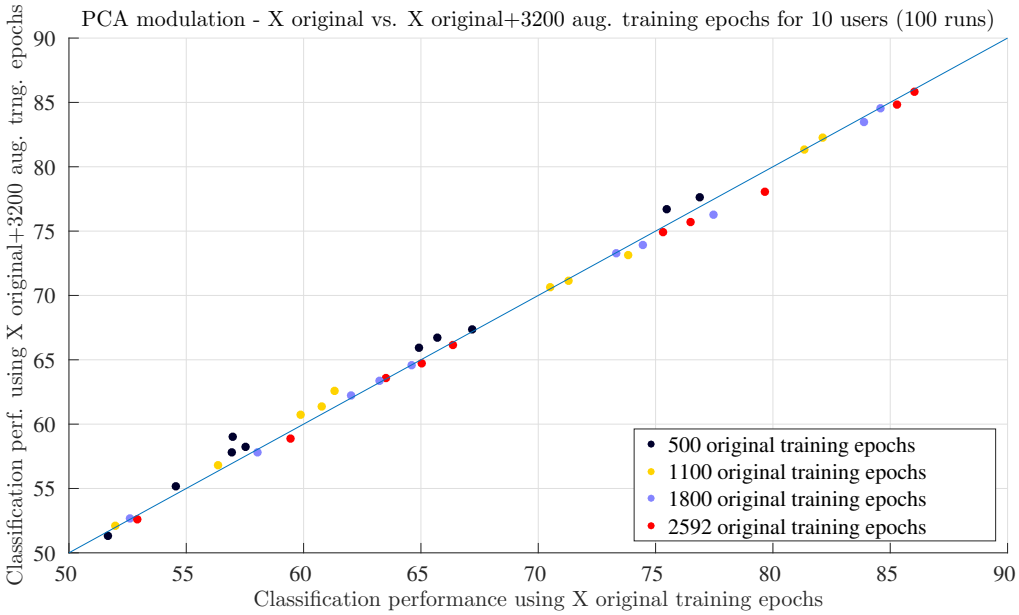

Figure 28: *A scatter plot for PCA modulation comparing classification of original epochs vs original + augmented (3200) epochs.*

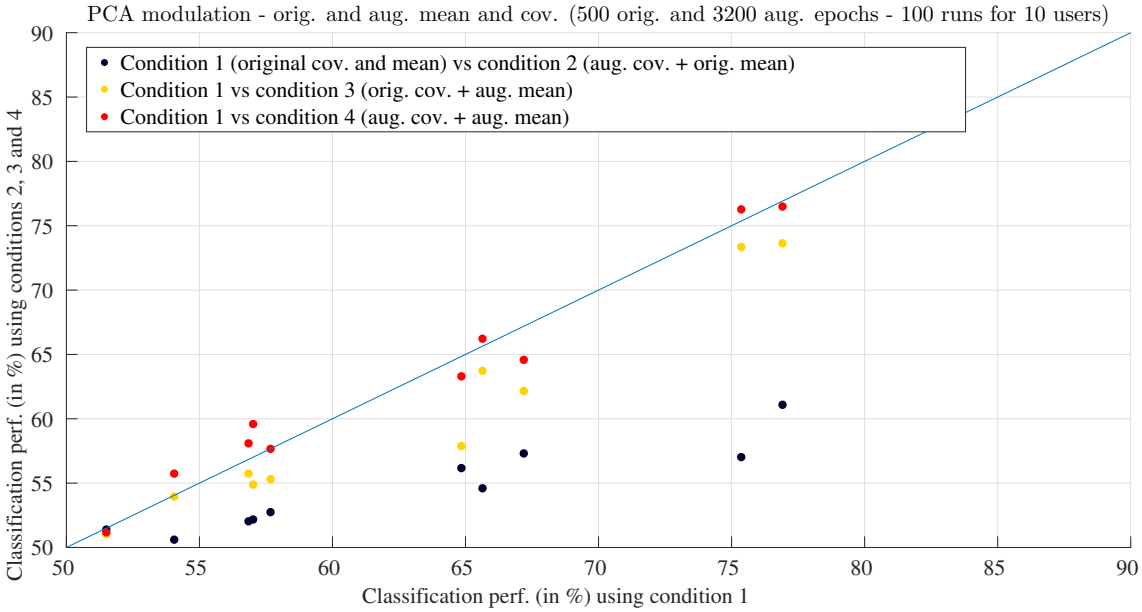

Figure 29: *A scatter plot for PCA modulation showing a comparison of classification performances using different combinations of mean and covariance of original (500) and novel (3200) epochs.*

6.5.1 Case study: performance of 3200 novel epochs using different sizes of original training epochs

Figure 31 shows the comparison of adding 3200 novel epochs using different sizes of original training epochs for 10 users. The results are averaged over 100 runs. For the size of 500 original training epochs, 5 users show an improvement over just using original training epochs while 4 of them stay on the blue line. This suggests, similar to PCA modulation, that using a lower number of original training epochs to generate novel epochs is beneficial while using a higher number like 1800 is not. Moreover, using a

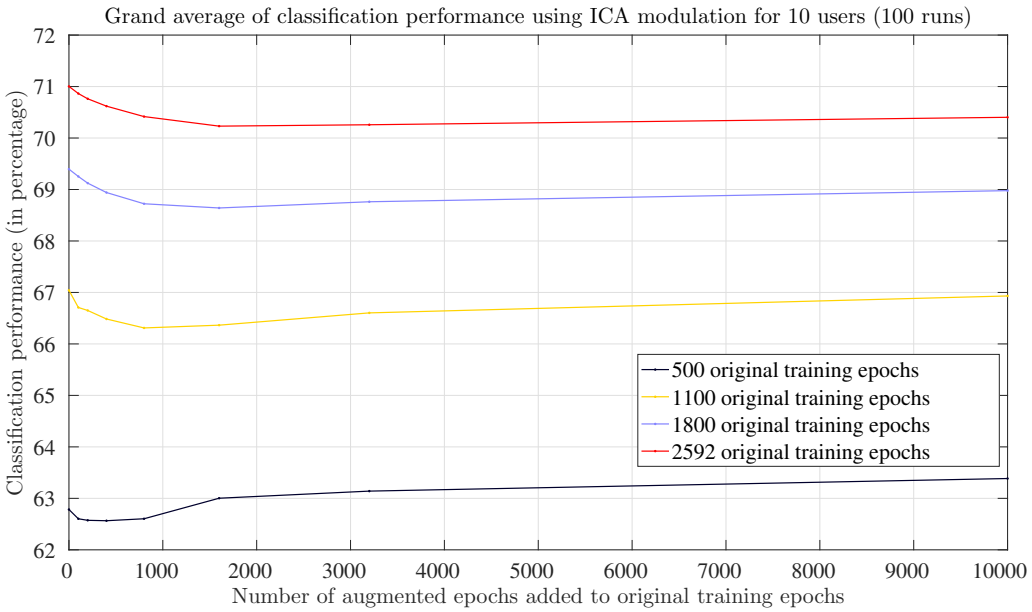

Figure 30: *Grand average performance using ICA modulation.*

higher number of original training epochs actually decreases the classification performance.

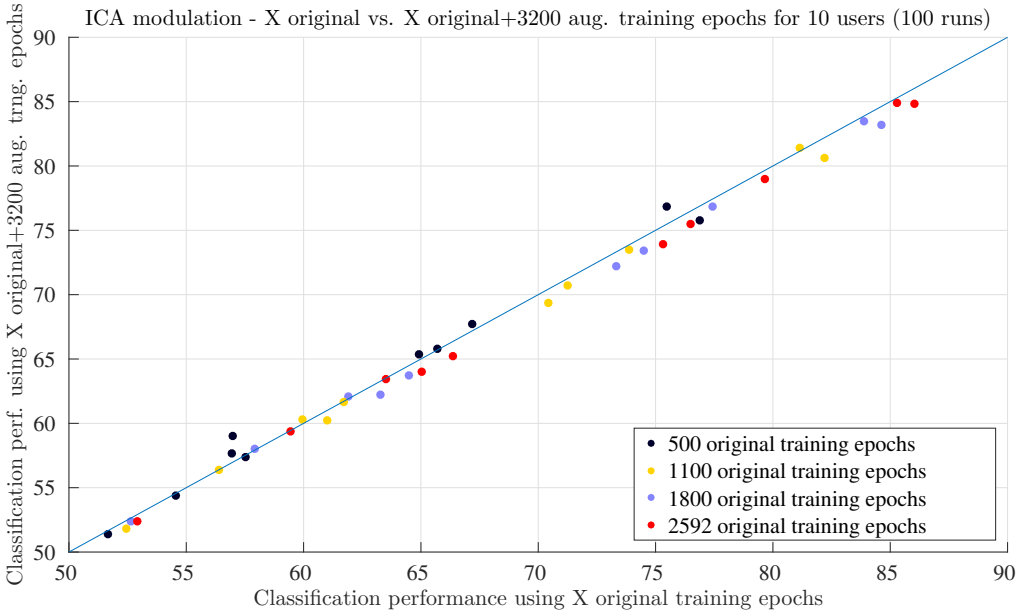

Figure 31: *A scatter plot for ICA modulation comparing classification of original epochs vs original + augmented (3200) epochs.*

6.5.2 Case study: mean and covariance using original (500) and novel epochs (3200)

Augmented epochs generated using ICA modulation improve the mean and covariance estimate as suggested by the figure 32 where condition 4 (augmented mean and covariance from 3200 novel epochs) performs better than original training epochs (500). The conditions 3 (original covariance and augmented mean) and 4 perform closely for some users but condition 4 is always better than 3. From this also, we

can conclude that covariance estimate from the augmented epochs becomes better than covariance estimate just from the original training epochs. The improvement shown by condition 4 does not come only by the better estimate of covariance but also by the better estimate of the mean. This is shown by the performance of condition 2 which performs well below the original training epochs performance as it uses just the augmented covariance.

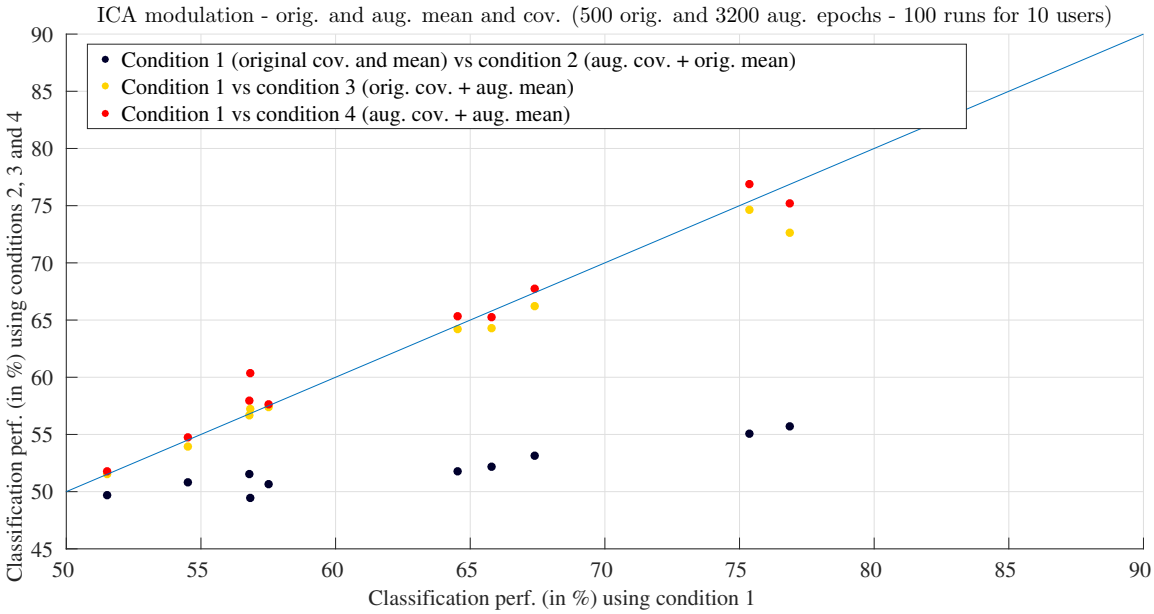

Figure 32: *A scatter plot for ICA modulation showing comparison of classification performances using different combinations of mean and covariance of original (500) and novel (3200) epochs generated.*

6.6 Comparison of the data augmentation strategies for each user

Figure 33 shows a bar plot comparing the classification performances of original training epochs (500) vs. novel epochs (3200) generated using all the data augmentation strategies discussed in the previous sections - amplitude modulation, time warping, PCA modulation and ICA modulation. Each user has five bars, each showing a classification value (in %). The first bar for each user shows performance using only original training epochs (no augmentation). The subsequent bars shows performance using original and augmented training epochs. 500 original epochs have been used to generate 3200 augmented epochs using different augmentation strategies. From the plot, we can conclude that PCA modulation bars (blue color) achieve the best performance for all the users (except user 4) among all the data augmentation strategies. For user 4, the classification with original training epochs itself remains close to 50% (uncertain). The ICA modulation bars (red color) also beat the performance of original training epochs for a few users such as user 1, user 2, user 8 and user 10. The other two strategies amplitude modulation and time warping do not beat the performance of original training epochs for most of the users.

6.7 Effect of multiple runs of experiment

In this section, we will see how averaging the classification results over multiple runs become better as we increase the number of runs (5, 20, 100 and 200). Before that, we will discuss briefly why do we need to run our experiments (4 evaluation pipelines for 4 data augmentation strategies) multiple times and take an average of all the results.

For all our data augmentation strategies, we use randomization at two stages:

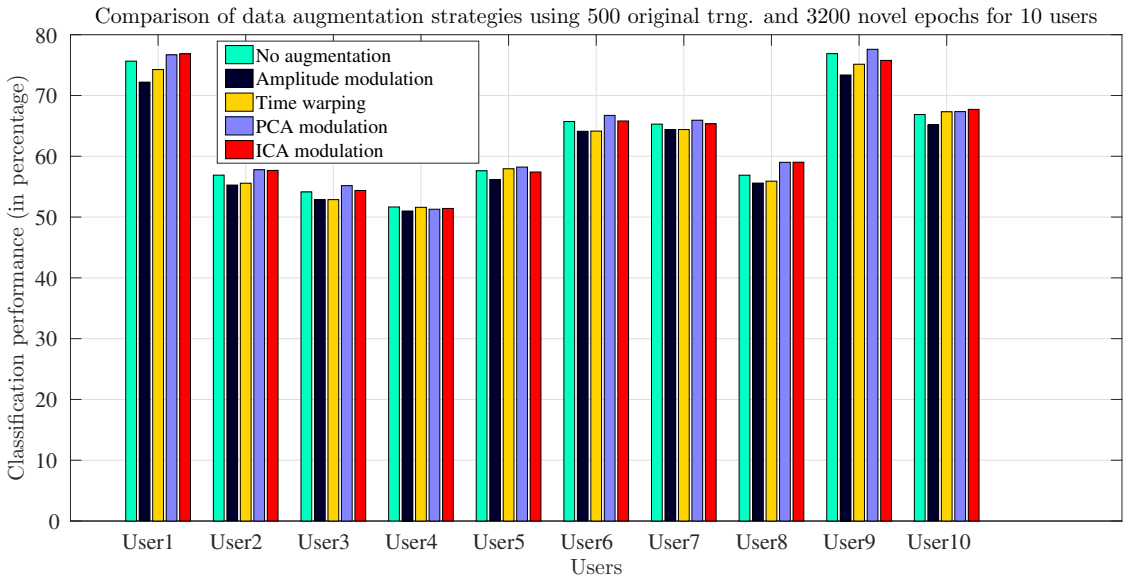

Figure 33: *A bar plot showing comparison of classification performances using all data augmentation strategies vs. no augmentation for all 10 users using 500 original training epochs to generate 3200 novel epochs.*

- Choosing epochs in a random way. For example, choosing 500 random epochs from 2592 original training epochs.
- Generating random factors from a range ($[0.9, 1.1]$ in amplitude modulation) or from gamma distribution (in PCA and ICA modulation) or gaussian random samples (in time warping).

In order to cancel the randomness caused by these two steps in our evaluation pipelines and stabilize the results, we average the classification results over multiple experiment runs (100). Though it helps to stabilize the results (our line plots become more smooth as the number of runs increases), the running time increases linearly with the number of runs.

We can see the averaging effect in the following grand average results from PCA modulation for 5, 20, 100 and 200 runs.

Figure 34 shows results for 5 runs. We can clearly see sudden peaks in the line plots especially for 500 and 1100 original training epochs. The line plot for 2592 original training epochs is more smooth because, for each run, the same epochs are selected, just with the different permutation. But for the other line plots, for example, in 500 original training epochs line plot, 500 epochs are randomly selected from 2592 original training epochs. Hence, the original training epochs can contain different epochs in a different run. This sampling accounts for the high variance in the 500 and 1100 original training epochs line plots. To stabilize the results, we average over multiple runs.

Figure 35 shows the results averaged over 20 runs. The line plots become more smooth especially the 500 and 1100 original training epochs line plots. Figure 36 shows the results averaged over 100 runs. We can see the line plots are more stable especially the 500 original training epoch line (black line). Figure 37 average results over 200 runs. The 500 original training epochs line plot shows a clear increase without dropping the performance anytime. It stays at the same value for small number of novel epochs (100 or 200 novel epochs) and increases for the higher number of novel epochs (1600, 3200, 10,000).

Visualizing these results, we can expect the classification results to become more stable once we further increase the number of runs (beyond 200) for all the data augmentation strategies.

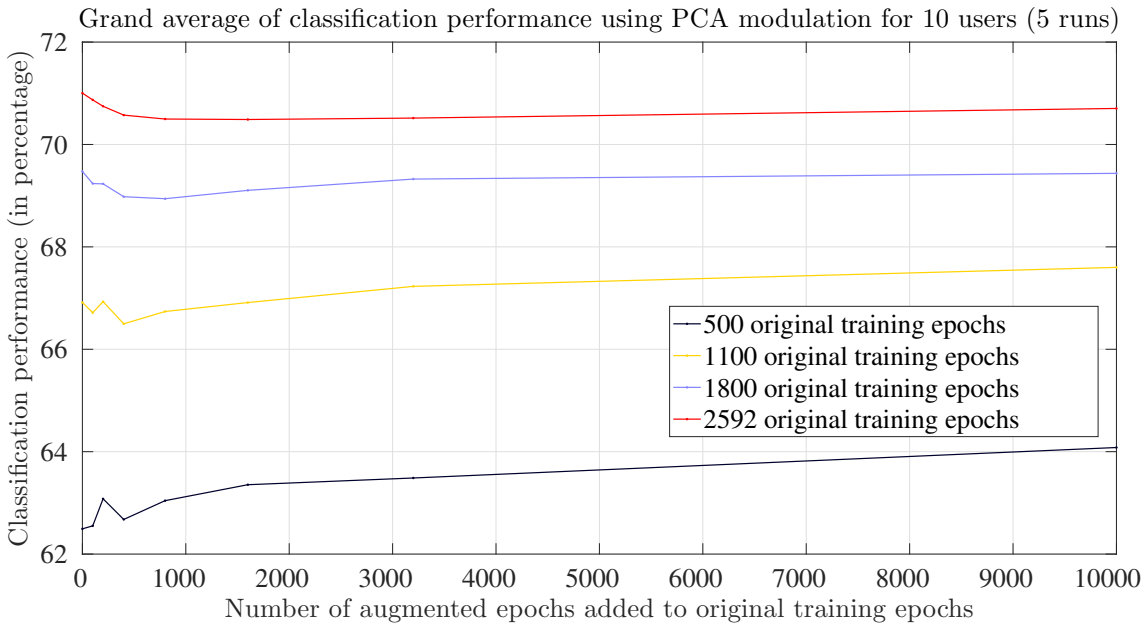

Figure 34: *Grand average performance for PCA modulation over 5 runs.*

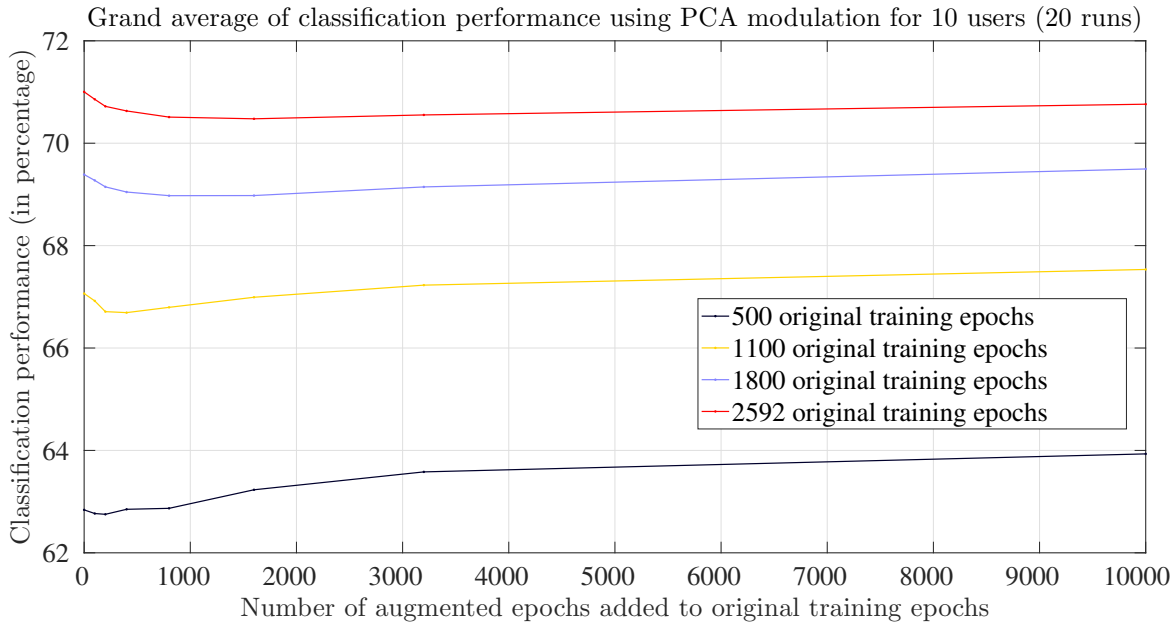

Figure 35: *Grand average performance for PCA modulation over 20 runs.*

6.8 Risks of data augmentation

There are potentials risks involved in generating novel epochs by warping the original ones:

6.8.1 Wrong data generation

It poses a potential risk as we generate novel epochs by distorting the original ones. It might happen that we create epochs which do not occur in real-time or the features do not correspond to the class label of the epochs. Another issue could occur if the classifier learns more features from the novel epochs rather than from the original epochs. These could lead to a drop in overall classification accuracy because the

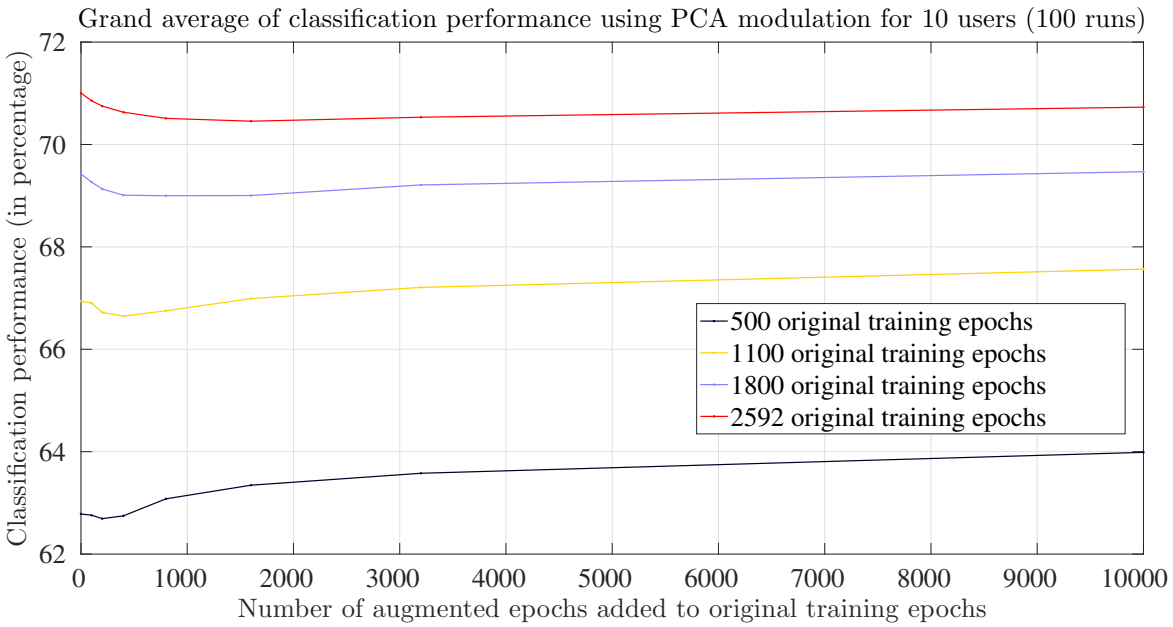

Figure 36: *Grand average performance for PCA modulation over 100 runs.*

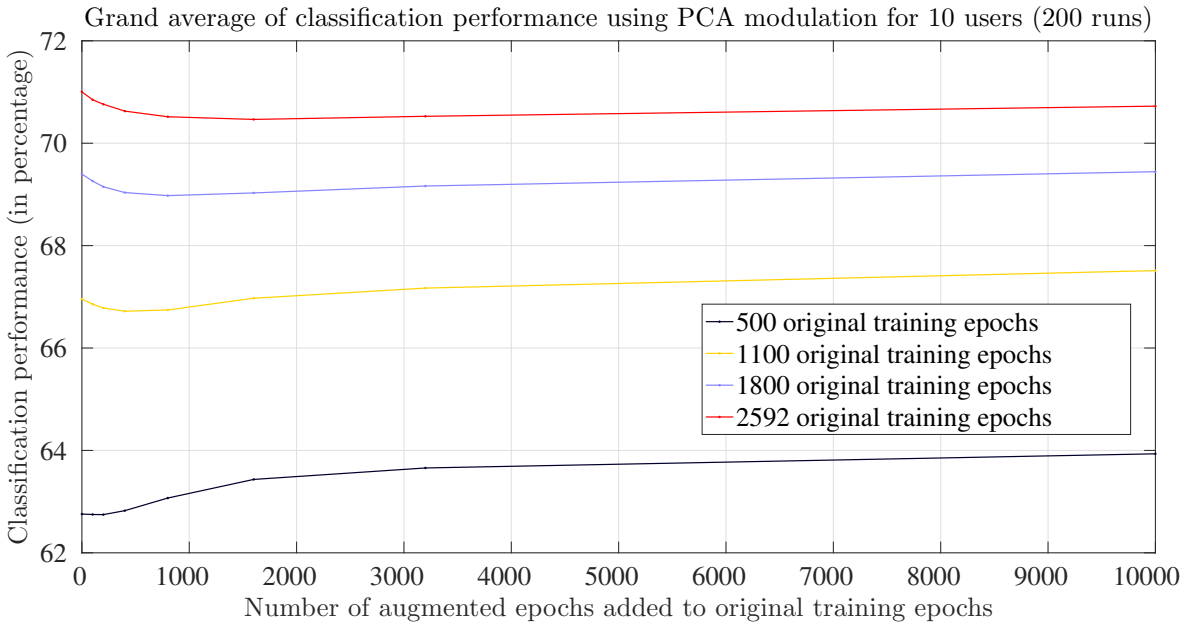

Figure 37: *Grand average performance for PCA modulation over 200 runs.*

classifier does not learn the variations that are present in the test epochs. To avoid such cases, we alter the original epochs in a small amount. For example, in amplitude modulation, the factors are chosen from the range $[0.9 - 1.1]$.

6.8.2 Non-convergence of ICA algorithm

For our ICA modulation approach, we decompose the epochs into their source components (30 sources) using ICA algorithm. This algorithm works in a non-deterministic way. It runs for a maximum number of 300 iterations for convergence to find each source component. Sometimes, the algorithm does not

converge for one or more independent component(s). Due to this non-deterministic behaviour, the number of independent components can be less than 30. The ICA algorithm (FastICA) used here is based on Hyvarinen's[4] fixed-point algorithm.

6.8.3 Dealing with outliers

We remove artifacts (for example eye-blinks) from our epochs so that our classifier learns features from the true mental state and not from artifacts. Once the novel epochs are generated, artifacts are removed from the training (original as well as novel) epochs and test epochs. If the difference between minimum and maximum amplitude is more than $60\mu V$ for any channel in an epoch, the entire epoch is discarded. We perform this artifact rejection at the end (after novel epochs generation) because we use fixed numbers of original training epochs to generate novel ones. The number of removed epochs due to the artifact rejection differ from user to user and because of this, we are not certain of which numbers to choose for the sizes of original training epochs so that it works for all the users.

6.8.4 Space complexity

In general, we deal with more number of epochs as compared to original epochs. The new enhanced training set needs more space. If we generate $10,000$ epochs, the dimensionality goes to $R^{140\times63\times10,000}$. Moreover, the epochs generated using the time warping strategy need even more space because they are sampled at $1000Hz$ and dimensionality increases from $R^{140\times63}$ to $R^{1400\times63}$ for each epoch.

7 Conclusion

We inspired ourselves from the other fields like the computer vision, speech recognition etc. which successfully implemented various data augmentation strategies to generalise the classifiers. Taking cues from these fields, we created 4 different pipelines which took into account the different characteristics of the epochs like amplitudes, time-shifts and principal components and sources. The takeaways from this study are:

- Generating novel epochs by doing transformations on original epochs.
- Evaluating four different ideas (amplitude modulation, time warping, PCA modulation and ICA modulation) of data augmentation.
- Comparing the ERP plots of original with novel epochs.
- Understanding the risks of incorrect data generation.

Moreover, compared to the amplitude modulation approach, the ICA modulation approach results in a better classification with the novel epochs especially for the lower percentages of the original training epochs. For example, a training set of 500 original epochs with $10,000$ augmented epochs measures 63.39% accuracy for ICA modulation while amplitude modulation with the same set measures only 59.93%. The accuracy with only original training epochs remains the same approximately at 62.7% for both the approaches. With the same number of original training epochs and novel epochs, PCA modulation achieves better performance compared to ICA modulation for all the sizes of original training epochs. Time warping does not improve upon the performance of the original training epochs but still better than amplitude modulation. For 500 original training epochs and 10,000 novel epochs, time warping measures 61.86%, about 2% more than amplitude modulation.

The novel epochs generated by the PCA and ICA modulation improve the estimate of covariance and mean which results in better classification when we choose a lower number of original training epochs

[4] `http://www.cis.hut.fi/projects/ica/fastica/`

to generate novel ones. The amplitude modulation and time warping strategies do not improve upon the classification performances for any size of original training epochs.

In order to remove the sudden peaks or drops in the classification line plots, we average the classification results over multiple runs. Using more number of runs (> 100) can make the classification line plots more stable.

8 Future Work

8.1 Changes in the evaluation pipelines

Our functional pipelines give us the freedom to alter the values of key parameters like modulating factors, time shifts, gamma distribution parameters and so on and evaluate the classification performance with different combinations of these parameters. We can go back to these ideas to look for changes to be made at the levels of epoch generation, artifact rejection or epoch classification.

We should also find out the reason why classification improves when we use a lower number of original training epochs to generate novel epochs (in PCA and ICA modulation) but if all the original training epochs are used, classification goes down. Moreover, in these approaches, we can increase or decrease the number of principal components and sources to see whether we improve classification performance. For the time warping strategy, which does not show any improvement, we can try to reduce the amount of the maximum shift from 200*ms* to somewhere between 100*ms* and 200*ms* so that we can retain more information in the novel epochs.

We have discussed in the previous section ("Risks of data augmentation") that the ICA algorithm does not always return the number of sources as we need. The FastICA versions we use in this study can be replaced by Kernel-ICA or ICA based on Infomax.

In this study, our data is imbalanced in the number of epochs per class. The non-target epochs outnumber the target epochs. We can direct the data augmentation towards balancing the epochs in terms of their classes. For example, we augment more target epochs to increase their percentage in an overall set of epochs compared to non-target epochs.

We have seen the performance comparison of different combinations of mean and covariance of original training and novel epochs. In amplitude modulation, the novel epochs seem to retain the mean (of original training epochs). In the PCA and ICA modulation, augmented covariance and mean become better. So, rather than adding them back to the original training epochs, we can use the novel epochs just to improve the mean or covariance depending on the augmentation strategy.

8.2 Different classifiers

We can use a different implementation of the LDA classifier or a different classifier altogether like the support vector machine (SVM) or artificial neural networks (generative adversarial networks). Generative adversarial network has two models - one learns the data distribution (generative) and another determines if a sample comes from original training set or the data distribution.

9 Appendices

9.1 Tables for grand average results

The tables show grand average of classification results (in %) used in the classification line plots above for 10 users. These results are averaged over 100 runs which means that each classification result (each value in the table) is an average of 100 values. The column 0 shows the baseline accuracy (with just original epochs and no novel epochs) for different numbers of original training epochs. The other columns like 200 show accuracy after adding 200 novel epochs to the original training epochs used. Each row corresponds to a certain number of original training epochs (e.g. 1800) which is used to generate novel epochs (200 or 3200).

9.1.1 Grand average results for data augmentation by amplitude modulation (100 runs)

# of original training epochs \ # of augmented epochs	0	100	200	400	800	1600	3200	10,000
500	62.69	62.43	62.23	61.99	61.79	61.52	61.01	59.93
1100	66.89	66.67	66.52	66.34	66.12	65.97	65.70	65.44
1800	69.28	69.14	68.99	68.91	68.71	68.44	68.40	68.33
2592	70.88	70.78	70.69	70.54	70.36	70.17	70.07	70.14

9.1.2 Grand average results for data augmentation by time warping (100 runs)

# of original training epochs \ # of augmented epochs	0	100	200	400	800	1600	3200	10,000
500	62.79	62.51	62.28	62.06	62.09	61.97	61.92	61.92
1100	66.94	66.70	66.59	66.38	66.17	66.00	66.05	66.23
1800	69.26	69.16	69.14	68.91	68.63	68.42	68.29	68.42
2592	70.90	70.80	70.72	70.58	70.39	70.13	69.98	69.94

9.1.3 Grand average results for data augmentation by PCA modulation (100 runs)

# of original training epochs \ # of augmented epochs	0	100	200	400	800	1600	3200	10,000
500	62.78	62.76	62.69	62.75	63.08	63.35	63.58	63.99
1100	66.94	66.90	66.72	66.65	66.75	66.99	67.21	67.57
1800	69.42	69.27	69.13	69.01	69.00	69.00	69.21	69.47
2592	71.00	70.86	70.75	70.63	70.51	70.45	70.53	70.73

9.1.4 Grand average results for data augmentation by ICA modulation (100 runs)

# of original training epochs \ # of augmented epochs	0	100	200	400	800	1600	3200	10,000
500	62.78	62.60	62.57	62.57	62.60	63.00	63.14	63.39
1100	67.04	66.71	66.65	66.48	66.31	66.36	66.60	66.93
1800	69.40	69.25	69.12	68.94	68.72	68.64	68.76	68.98
2592	71.00	70.86	70.76	70.62	70.42	70.23	70.26	70.40

9.2 Notations

- Hz - Hertz
- $\sum$ - Covariance matrix
- R^{MXN} - Matrix of real numbers with M rows and N columns
- $E[x]$ - Expectation of all the values taken by x

- μ - Mean
- *ms* - Milliseconds
- μV - Microvolts
- *sign* - Sign of a real number, positive or negative

References

[1] F. Lotte, "Generating Artificial EEG Signals To Reduce BCI Calibration Time," *5th International Brain-Computer Interface Workshop, Graz, Austria*, 2011.

[2] Emmanuel Kalunga, S. Chevallier, and Q. Barthelemy, "Data augmentation in Riemannian space for Brain-Computer Interfaces," *ICML Workshop on Statistics, Machine Learning and Neuroscience*, 2015. [Online]. Available: https://hal.inria.fr/hal-01225255

[3] M. Teplan, "Fundamentals of EEG Measurement," *Measurement Science Review*, 2002.

[4] J. Wolpaw and E. W. W. (Editors), "Brain-Computer Interfaces - Principles and Practice," 2012.

[5] F. Lotte, F. Larrue, and C. Mühl, "Flaws in current human training protocols for spontaneous Brain-Computer Interfaces: lessons learned from instructional design," *Frontiers in Human Neuroscience*, 2013.

[6] P. Wang, J. Lu, B. Zhang, and Z. Tang, "A Review on Transfer Learning for Brain-Computer Interface Classification," *5th International Conference on Information Science and Technology (IC1ST), China*, 2015.

[7] S. Sur and V. K. Sinha, "Event-related potential: An overview," *Industrial Psychiatry Journal*, 2009.

[8] F. Lotte, L. Bougrain, and M. Clerc, "Electroencephalography (EEG) based Brain-Computer Interfaces," *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2015. [Online]. Available: https://hal.inria.fr/hal-01167515

[9] F. H. Duffy, G. B. McAnulty, M. C. McCreary, G. J. Cuchural, and A. L. Komaroff, "EEG spectral coherence data distinguish chronic fatigue syndrome patients from healthy controls and depressed patients-a case control study," *BMC Neurology*, 2011.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25*, 2012.

[11] N. Jaitly and G. E. Hinton, "Vocal Tract Length Perturbation (VTLP) improves speech recognition," *International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[12] X. Cui, V. Goel, and B. Kingsbury, "Data Augmentation for Deep Neural Network Acoustic Modeling," *IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 2014.

[13] J. Schlüter and T. Grill, "Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks," *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, 2015.

[14] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis," *Proceeding ICDAR '03 Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003.

[15] I. Winkler, S. Haufe, and M. Tangermann, "Automatic Classification of Artifactual ICA Components for Artifact Removal in EEG Signals," *Behavioral and Brain Functions*, 2011.

[16] B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Muller, "Single-trial analysis and classification of ERP components — A tutorial," *NeuroImage*, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811910009067