Controlled Dropout using Functional Measures in ANN and CNN architectures: An Example using Pupillary Responses to Real and Posed Smiles

Jesse Wright[1][0000−0002−5771−988X]

Australian National University, Canberra ACT 0200, Australia https://www.anu.edu.au/
jesse.wright@anu.edu.au

Abstract. Time series classification (TSC) is a well known data-mining challenge. Recently, there has been investigation into the use of Artificial Neural Networks (ANN's) and Convolutional Neural Networks (CNN's) to address this. In this paper we perform a comprehensive study investigating the use of Functional Measures to enhance each of these approaches. For ANN's, a common hurdle is that of extracting and selecting features of datasets on which to train neural networks. Factors that increase the difficulty of such a task include: having small or unbalanced datasets; working with data that has already been pre-processed by other parties and working with data which has a large set of possible features to extract. This paper analyses existing techniques of weight matrix analysis, functional measure analysis and sensitivity analysis to help solve a classification problem on real-world timeseries data. Contrastingly Convolutional Neural Networks, which take the timeseries itself as input, are known to have high training times, poor explainability and are susceptible to overfitting. We address this problem by introducing a novel technique for controlled dropout between convolutional layers which extends upon existing *Angle*-Based functional measures.
Trials for this experiment are run on a range of network models (including ANN, CNN and SVM classifiers), optimizers and training sets in order to analyse the generalisability of each technique. Our findings indicate that on average $> 30\%$, and in certain cases $> 99\%$, of nodes in fully connected CNN layers can be dropped without impacting accuracy. We also show that our novel dropout technique can reduce bias in basic CNN's and prune unnecessary neurons in more complex models. Due to limitations in the dataset, all ANN techniques were found to yield results of between 45-60% accuracy. Based on the range of tests applied, we conclude that such techniques are limited in their capacity to improve network accuracy on the given dataset.

Keywords: Distinctiveness Analysis · ANN · CNN · Time Series · Observers · smilers' · Physiological Features · Controlled Dropout · Functional Measures

1 Introduction

Time Series Classification (TSC) refers to the task of labelling time series data; and is considered one of the most challenging data mining problems of the last two decades [35] [9].

Due to a growing abundance of Time Series datasets [32], many data mining techniques have been applied to TSC [2]. These include: one-nearest-neighbour dynamic-time-warping [1], elastic ensemble [22], time series forest [7] and the collective of transformation-based ensembles (COTE) [3]. However, such techniques have critical shortcomings [2], such as: COTE having a $O(n^2T^4)$ time complexity where n is the number of time series and T is the length of the time series [11]; and the remaining techniques being a factor of 2-12x less accurate, when compared on datasets such as the UCR Archive [6], [2].

More recently, there has been research into application of Deep Neural Networks (DNNs) to the task of TSC [10]. In particular, CNN architectures have seen widespread usage in classification tasks with a large input size and in more recent years, they have begun to be adopted for the task of time series classification [37].

Whilst such techniques have been proven to yield high accuracies [11], there are several shortcomings to their current usage. This includes high computational costs for training [30]; overfitting [5]; and poor explain-ability of networks [25]. Consequently, there is demand for techniques which minimize these factors. One such technique that has been applied to many other network architectures is Dropout, which reduces the number of active nodes in the network over time and hence reduces the impact of all 3 aforementioned issues [4]. However, many recent works have concluded that traditional dropout techniques often detrimentally impact in the training of CNN's [4]. Hence there is an incentive to find ways of implementing dropout on CNN architectures designed for timeseries' that do not negatively impact performance.

Smiling can be used to convey various human emotions including joy, surprise and nervousness [18]. Smiles can also be posed for dispassionate reasons such as communication or out of social etiquette [8]. Throughout this paper we consider two classes of smiles: real smiles, which arise from happiness; and posed (controlled) smiles.

Existing work classifying real and posed smiles has demonstrated 96.1% [19] prediction accuracy. Three physiological signals from 24 observers reacting to 20 distinct stimuli were used to achieve this.

In this work we use only pupilliary response for the purpose of prediction. Pupilliary response can be affected by a multitude of factors including pain, stress, cerebral activity and visual stimuli [15]. Hence it is likely that pupil size will be subject to change differently based on whether real or posed smiles are being viewed. However, this data is also likely to be noisy given the range of influences on pupil size; substantially increasing the difficulty of the classification task due.

The dataset used [19] comes from existing studies, which present two forms of the data. The first consists of 20 pre-processed features of 10 observers 'average' response to real or posed smiles. Given the large number of features relative to the number of datapoints, this view of the dataset is well suited to a joint approach of feature selection, followed by classification with an ANN [21].

The second view of the dataset consists of 2 timeseries' (one for each pupil) for each of 10 participants, each watching 10 smilers' videos. In order to directly perform classification on this sequential data, we choose to employ deep learning techniques [29]; in particular - we use a basic CNN approach; and then extend this to the recently developed InceptionTime model [11]. With these CNN architectures we are able to rigorously test our novel approach extending DropBlock [14].

Together, these views make the dataset well suited to our study of using Functional Measure for Controlled Dropout - including first layer feature selection.

Networks in this experiment were trained with error-backpropogation [31] using cross-entropy loss [36]. To ensure fair comparison with existing work [13], all connections are forward passing weighted links between consecutive network layers. The networks were trained on pre-processed and labelled feature vectors. Further trials were performed in the ANNs with training sets consisting of additional data points extrapolated from the original training data. Network training was terminated after a fixed number of epochs to reflect resource limitations when these techniques are applied in the real world. Following this, the test set accuracy was recorded. For the ANNs in this study, we have used the Sigmoid Logistic Activation function $y = (1 + e^{-x})^{-1}$ and Leaky Rectified Linear Units (LReLU) [24] with a gradient of 0.01 for negative inputs. Due to the limited size of the dataset for the ANNs, we used leave-one-subject-out (LOSO) cross-validation [16] to improve our analysis of the networks. The results presented in this paper are the aggregated results from the test set in each trial of LOSO cross-validation. Since every datapoint is used in the LOSO cross-validation, we present results for several predetermined network architectures rather than performing feature engineering with data that is also used for analysis. This improves the integrity of our results [20].

For our study with the CNNs, we also need to ensure that all test data is fully free from training data. To do so, we extend upon the Independent Approach introduced in a previous study which used a superset of the data analysed in this paper [19]. To achieve this; all hyperparameter selection is performed with all data relating to 3 chosen individuals, and 8 chosen stimuli removed[1]. Leave-one-subject-and-one-stimulus-out (LOSSO) cross-validation [13] can then be robustly achieved by using test sets consisting only of combinations of the aforementioned persons and videos. This way, no information from the test set is able to influence the results.

2 Network Topologies

2.1 ANN

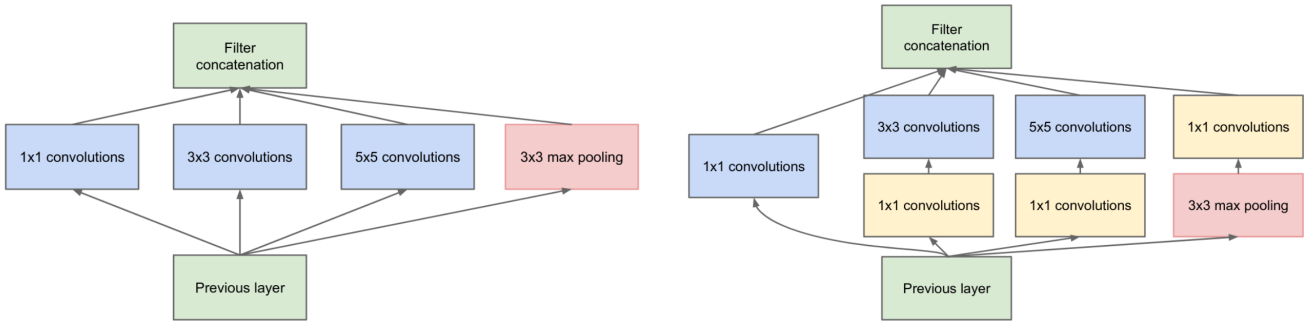
Our techniques were evaluated on a 20-10-2 network topology; that is, twenty inputs, ten hidden neurons, and two output neurons and a 16-10-10-2 network topology. For each topology we ran tests with the Sigmoid Logistic Activation function and separately with the Leaky ReLU activation function.

2.2 Deep Learning Architectures

Basic CNN The base CNN architecture (c.f. Figure 12) consists of 5 1-dimensional convolution layers with sizes 2, 32, 64, 32 and 16 respectively. They are all given the same kernel size which is a calculated hyperparameter. Following this is a dropout layer, with dropout rate given by a calculated hyperparameter. There is then a 1-dimensional pooling kayer with kernel size 2. The channels are then flattened and fed into a 10 node linear layer with ReLU activation which is followed by a 2 node linear layer with Softmax activation to produce the binary prediction.

[1] The labels for the observers are p2, p4 and p9 and labels for the 8 videos are H1, H2, H3, H5, A1, A2, A3, and A4

Inception Time InceptionTime [11] is a network architecture that extends upon traditional CNN's; designed with the intention of creating a generic CNN architecture for deep learning on time series' akin to the role of AlexNet in time series'. It has been found to have a prediction accuracy comparable to that of existing state-of-the-art classifiers such as HIVE-COTE, whilst also providing a more scalable design [11].



(a) Inception module, naïve version (b) Inception module with dimension reductions

Fig. 1. Inception Module [33]

The Inception architecture was originally developed for classification of data from ImageNet [33]. An inception module consists of an ensemble of convolutional and pooling layers whose results are concatenated together to produce a 'filter concatenation' (c.f. Figure 1) which acts as the input to the following layer. A basic Inception network can then be formed by layering several inception module's together; max-pooling layers can also be introduced to reduce the dimensionality of the grid.

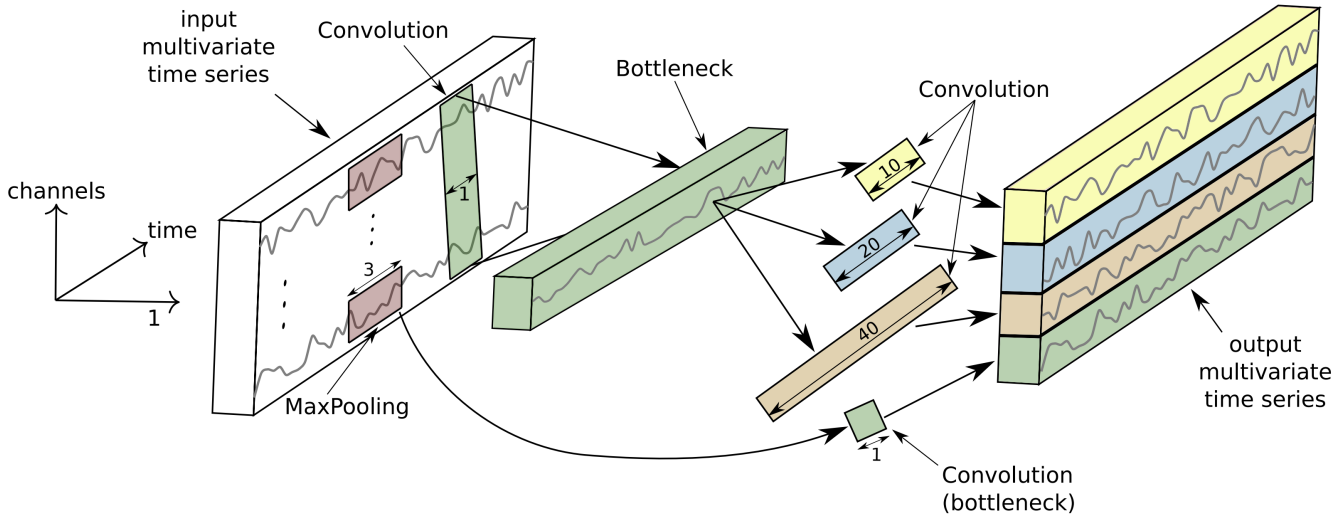


Fig. 2. InceptionTime architecture [11]

The InceptionTime architecture (c.f. Figure 2) consists of an ensemble of 5 deep learning models for TSC. It is created by cascading [23] inception modules; where each module has the same base architecture, and different randomly initialised weights. It also includes filters which have varied lengths so as to enable learning over time sequences of varied lengths.

The InceptionTime Model has 3 hyperparameters determining its architecture. The first is the number of inception modules to cascade, we note that there must be at least 1 inception module in order for the network to learn; and we limit the depth to a size of 4 due to resource constraints. The second is the number of channels to use at a bottleneck; since the data we are working with consists only of 2 channels (left pupil dilation and right pupil

dilation), the choice is limited to single channel bottlenecks or disabling bottlenecks within the network. The final constraint is the size of the (largest) kernel size in the inception module with the following two blocks using kernels of 1/2 and 1/4 of the largest kernel size respectively. Hence, the kernel size parameter should be a strictly positive multiple of 4.

Hyperparameter Selection Hyperparameter selection for the Basic CNN and the InceptionTime model were each performed in 2 stages. First the hyperparameters determining the model architecture were selected using a learning rate of 0.01 over 20000 epochs. Each possible combination of hyperparameters were tested. The loss over the last 50 epochs was then analysed and the combination of parameters with the lowest average, for the lowest 40% of losses, was selected.

Once the model architectures were selected, they were tested with learning rates from 0.005 to 0.02 over 20000 epochs. The learning rate which had the lowest average for the bottom 40% of losses over the last 50 epochs was then selected.

Hyperparameter	Range	Selection
Dropout Rate	0.25, 0.5, 0.75	0.25
Kernel Size	4, 8, 16, 32	4
Learning Rate	0.005, 0.01, 0.015, 0.02	0.01

Fig. 3. Hyperparameter selection for Basic CNN

Hyperparameter	Range	Selection
Number of inception modules	1, 2, 3, 4	4
Kernel Size	4, 8, 16, 32	32
Bottleneck Channels	Yes, No	Yes
Learning Rate	0.005, 0.01, 0.015, 0.02	0.01

Fig. 4. Hyperparameter selection for InceptionTime

Figures 3 and 4 and outline the values for hyperparameters that were tested and the subsequent selections.
All test data to be used in the LOSSO validation was removed prior to performing hyperparameter selection.

3 Analysis Techniques

We analyse the following techniques on ANNs:

Garson measure (1991) [12] which calculates the proportional contribution of an input to a particular output:

$$\mathrm{G}_{ik} = \frac{\sum_{j=1}^{nh} \mathrm{P}_{ij} \cdot w_{jk}}{\sum_{q=1}^{ni} \left(\sum_{j=1}^{nh} \mathrm{P}_{jk} \cdot w_{qj} \right)} \tag{1}$$

where

$$\mathrm{P}_{jk} = \frac{w_{\mathrm{jk}}}{\sum_{r=1}^{nh} w_{rk}} \tag{2}$$

Milne measure (1995) [27] which modifies *Garson* measure to prevent the sign of the contribution being lost. This is done by setting

$$\mathrm{P}_{\mathrm{jk}} = \frac{|w_{jk}|}{\sum_{r=1}^{nh} |w_{rk}|} \tag{3}$$

Wong measure (1995) [34] which distinguishes th magnitude of the contribution from the sign of the contribution:

$$\mathrm{Q}_{ik} = \sum_{r=1}^{nh} (\mathrm{P}_{\mathrm{ir}} \times \mathrm{P}_{rk}) \tag{4}$$

where P_{ik} is as defined for *Milne* measure.

Angle measure [13] which analyses hidden neuron activation's when data from the training set is applied. This enables one to determine the angle between the mutli-dimensional vectors that are formed:

$$\text{angle (i,j)} = \tan^{-1}\left(\sqrt{\frac{\sum_{\text{p}}^{\text{pats}} \text{sact(p,i)}^2 * \sum_{\text{p}}^{\text{pats}} \text{sact(p,j)}^2}{\sum_{\text{p}}^{\text{pats}} (\text{sact(p,i)} * \text{sact(p,j)})^2}} - 1\right)$$

where

$$\text{sact(p,h)} = \text{ activation (p,h)} - 0.5$$

To transform this angle into a measure between with a range of $[0, 1]$ we have crafted the metric

$$L(i,j) = |1 - \frac{4}{\pi}|\text{angle}(i,j)||$$

which can be used in conjunction with a threshold value $t \in [0, 1]$ to create the condition for a neuron/input to dropout when $L(i, j) \leq t$. In our implementation, the choice of dropping i or j is done randomly. An alternative method would be to drop whichever has the largest value of L with respect to all other neurons in the layer.

3.1 Extending Analysis Techniques for CNN architectures

Many recent works have concluded that traditional dropout techniques often have detrimental impact in the training of CNN's [4]. It is hypothesised that this is in large part, due to the conflict between dropout and batch normalization. DropBlock [14] is a technique which addresses this by introducing a more structured form of dropout in which contiguous elements of a feature map are collectively dropped.

We produce a novel extension to this technique, by using *Angle* measure to determine the appropriate section of the feature map to drop. Our selection method uses a predefined hyperparameter w which determines the width of the block to be dropped either side of the selected node. We use $w = 5$ throughout our experiments.

We assign to each node in a channel n (except those less than a distance w from either end of the channel), a value $K_{w,n}(i)$. This corresponds to the average *Angle* measure between the node and all other elements in the channel *excluding* those nodes within a radius of w of the node - this is to reduce the likely-hood of all elements representing the same feature from being dropped together.

$$K_{w,n}(i) = \frac{\sum_{j=1}^{i-w} L(i,j) + \sum_{j=i+w}^{nh} L(i,j)}{nh - 2w}$$

We note that $K_{w,n}(i)$ is only well defined for $w \leq i \leq nh - w$. For $w \leq i \leq nh - w$ we then define the average of this value across each channel as

$$J_w(i) = \frac{1}{c}\sum_{i=1}^{c} K_{w,n}(i)$$

where c is the number of channels. From this, we can determine the position which has the highest similarity to other positions across all channels, this is given by

$$x = \underset{i \in (w, nh-w)}{\arg\max} \{J_w(i)\}$$

Dropout is then performed with the additional parameters *max_blocks* and *threshold*. Whenever dropout takes place, all the $J_w(i)$'s are calculated. If there are any i such that $J_w(i) \geq threshold$ then x is calculated and we dropout all nodes within a radius of w of x. An example of this dropout, in the case where $w = 2$, is given in Figure 5. This is repeated at most *max_blocks* stopping earlier if the $J_w(i)$'s are less than *threshold*.

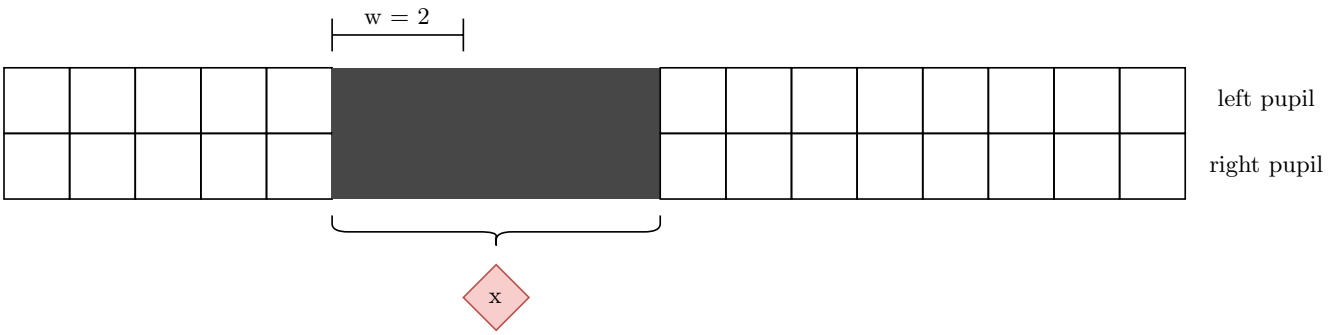


Fig. 5. Block dropout - w is a hyperparameter and x is selected using *Angle* measure

3.2 Implementation

For the ANNs, these techniques were implemented by creating custom dropout layers which set the value of 'de-selected' features / neurons to zero. Consequently, the terms dropout and feature selection are used synonymously throughout this paper.

The *Angle* Measure based DropBlock technique was added to the basic CNN model by replacing the existing dropout layer. For the InceptionTime model we apply this extended DropBlock technique directly, following the convolution layers in each of the inception modules.

4 Dataset: Smilers' and Observers' Physiology

4.1 Time Series Data

The data for this experiment was collected as part of a larger study of physiological responses to smilers' stimuli (video clips of real and posed smiles) [13]. It describes the pupillary sizes of 10 participants in reaction to smilers' stimuli, with one channel for each of the observers left and right pupils. Each participant viewed 20 video clips, 10 of real smiles and 10 of posed smiles. Their pupillary response to each video response was captured at 60Hz for 10 seconds. This was clipped to 600 sample points per video and eye blink points were reconstructed with the cubic spline interpolation technique [13]. The data was then smoothed using average filtering (Hanning window) and the maximum value normalization technique was applied to keep signals in the range 0 to 1. When used in the CNN, instances where data for either pupil in a trial was missing or corrupted, the data for both pupils was discarded. The result is 185 sets of channels for posed smiles and 81 sets of channels for real smiles.

4.2 Feature Data

For each observer, the average pupil size at each time step of watching real smilers', and watching posed smilers' was then taken. The result is 20 sets of time series data consisting of 1 average response to posed smilers' and 1 average response to real smilers' per observer. We have access to the dataset in this format and use it to understand and analyse the nature of the dataset. It is not used for model training and evaluation. Instead, we use features extracted from these timeseries' during previous research [19]. For each of the 20 pre-processed time series' the following 20 features were extracted: mean, maximum, minimum values, interquartile range, variance, sum, skewness, kurtosis, number of peaks, root mean squared error, aac, hjorth mobility, hurst exponent, mean first difference, mean second difference, samp en, ap en and fuzzy en and are visualised in Figure 10.

Figure 9 presents the timeseries for the average responses to posed and real smilers'. There is an observable correlation between the average time series for the real and posed smilers'. This justifies the need for LOSO cross-validation which has also been used in previous work on the data [13]. To confirm this, we performed leave-one-datapoint-out cross-validation on a 20-10-2 network with Sigmoid activation and found the average test result to be 16%, significantly lower than the 50% prediction rate that is achieved by random selection on a 2 class problem with an evenly balanced dataset. This correlation carries over to the feature statistics which are presented in Figure 10.

4.3 Data Preparation

In each iteration of LOSO cross-validation for the ANNs, the feature vector x_{raw} (including test vectors) were normalized (to x_{norm}) according to the equation

$$x_{norm} = \frac{x_{raw} - \min(Tr)}{\max(Tr) - \min(Tr)} \tag{5}$$

where Tr is the training set. This ensures that the features (such as sum, cf. Fig 10) are not weighted unfairly based on their magnitude relative to other features. Note that we do not include the test set statistics in the calculation of the min/max constants for this equation, so as to ensure that data in the test sets have no influence on training performance.

4.4 Data Extrapolation

As can be seen in Figures 9 and 10, there are clear indicators that a signal is from a response to a posed or real smiler when considered with respect to the same observer. For instance in 80% of observers, the values for hurst exponent and variance were larger for posed smiles than real smiles.

Based on this observation, we use four equations to extrapolate additional datapoints for training. Let p_r, p_f denote the feature vector of average pupillary response timeseries for individual p for all real and posed smiles respectively. Then for individuals p and q we can generate the new feature vectors $f^{(1)}_{p,q}$, $f^{(2)}_{p,q}$ for posed responses and $r^{(1)}_{p,q}$, $r^{(2)}_{p,q}$ for real responses according to the equations

$$\begin{aligned} f^{(1)}_{p,q} &= p_f - p_r + q_r \\ r^{(1)}_{p,q} &= p_r - p_f + q_f \\ f^{(2)}_{p,q} &= \frac{p_f + p_r}{2} + \frac{q_f - q_r}{2} \\ r^{(2)}_{p,q} &= \frac{p_f - p_r}{2} + \frac{q_f + q_r}{2} \end{aligned}$$

Importantly we note that these equations preserve the 'order' of each feature between the observation of real smilers' and the observation of posed smilers'. That is, if $p_f \leq p_r$ and $q_f \leq q_r$ then $f^{(i)}_{p,q} \leq r^{(i)}_{p,q}$.

We perform two sets of LOSO cross-validation experiments with this extrapolated data. One using the equations $f^{(1)}, r^{(1)}$ (which we call the *subtract add* dataset) and the other using $f^{(2)}, r^{(2)}$ (which we call the *difference of averages* dataset). In each of these experiments the training set is selected from the original features data, and then for all p, q in the initial training set $f^{(i)}_{p,q}, p^{(i)}_{p,q}$ are added to the expanded training set. We note that these extrapolations are *not* performed on any data in the test set in order to maintain the integrity of our results.

5 Experimental Design

For the ANNs we ran each set of LOSO cross-validation experiments independently on the 3 feature datasets described above (the original features dataset, the *subtract add* extrapolated dataset, and the *difference of averages* extrapolated dataset). Each set of experiments consisted of independently training and evaluating the 4×2 networks and optimizer combinations, and then running the experiments again with *Garson*, *Milne*, *Wang* and *Angle* dropout techniques applied to each of the networks. All of the experiments were performed with a learning rate of 0.01 over 100 epochs. Dropout based on *Garson*, *Milne*, *Wang* or *Angle* measure was enabled on every 10th epoch after the first epoch. To each 'dropout' layer we added the condition that at least 1/3 neurons/inputs in each layer must remain activated, in order to ensure there are enough features to learn from [17]. The thresholds applied to each of the measures were *Garson* - 0.93, *Milne* - 0.90, *Wong* - 0.90 and *Angle* - 0.92. Before each trial the random seeds and network weights were reset. Thus, each technique is being evaluated on a model with the same initial weights which improves the rigour of our experiment.

For the CNNs LOSSO validation was used; however only data from persons p2, p4 and p9; and videos H1, H2, H3, H5, A1, A2, A3, and A4 which remain unseen. The remainder of the data was used for hyperparameter selection and has thus been seen by the network. A dropout threshold of 0.8 was used throughout experiments and trials were run with no extended dropout; and extended dropout with *max_blocks* set to 1, 10 and 100. Each network was trained and evaluated for 5000 epochs, with dropout taking place on epochs 1000, 2000 and 3000 so as to allow time for the network to re-converge after the last round of dropout. The Adam optimizer was used to train the CNNs as resource limitations prevent us from running enough epochs for SGD to converge.

6 Results

We omit the results for training ANNs with the Stochastic Gradient Descent (SGD) optimizer as the loss function did not converge after 100 epochs. For completeness, we note that the results of LOSO cross-validation were between 45% and 60% for all models and techniques when using SGD.

Figure 6 summarises the results of the the experiments run with the Adam optimizer. We can see that in all instances, there is little or no performance increase obtained by use of each of the measures and that the base network itself is approximately random in the results it produces. In all 3 tests, the *Angle* based measure performed marginally worse than all other measures; however, the margin of error is too large to come to any conclusion. In addition, hyper parameters such as threshold can impact comparative performance.

As a baseline for comparison, we also include the results obtained when training an SVM with a linear kernel for 250 epochs on each dataset.

Dataset	Technique	Min. Test Accuracy	Max. Test Accuracy	Avg. Test Accuracy
Original Features	Base	0.45	0.5	0.48
	Garson Measure	0.45	0.5	0.5
	Milne Measure	0.5	0.5	0.5
	Wang Measure	0.5	0.5	0.5
	Angle Measure	0.45	0.5	0.46
	SVM	0.45	0.45	0.45
Subtract Add	Base	0.45	0.55	0.50
	Garson Measure	0.5	0.55	0.53
	Milne Measure	0.5	0.55	0.51
	Wang Measure	0.5	0.55	0.51
	Angle Measure	0.5	0.55	0.51
	SVM	0.6	0.6	0.6
Difference of Averages	Base	0.35	0.55	0.48
	Garson Measure	0.55	0.50	0.51
	Milne Measure	0.5	0.55	0.51
	Wang Measure	0.5	0.55	0.51
	Angle Measure	0.45	0.55	0.5
	SVM	0.55	0.55	0.55

Fig. 6. Summary statistics for testing basic neural networks with various measures

Network Architecture	No Extended Dropout	1 *max_blocks*	10 *max_blocks*	100 *max_blocks*
Basic CNN	33.3	45.8	41.7	41.7
InceptionTime	54.1	41.6	41.6	54.1

Fig. 7. Comparison of performance of CNN's with and without BlockDrop based on Angle Measure

Module	Mean	Median
1	38.1	19.9
2	80.1	88.1
3	57.1	57.1
4	98.8	98.8
Combined	68.5	89.8

Fig. 8. Dropout Statistics for InceptionTime

7 Discussion

We observe from Figure 6 that in the case of the original feature set, all ANN's and the SVM are producing results equatable to that of random selection given the even distribution of classes in the feature dataset. The inability of the SVM to produce meaningful predictions indicates that there are no linear solutions to the problem and the inability of the base ANN's to produce meaningful predictions is also not surprising given the limited size of the dataset.

We reason that we cannot expect that the use of these measures for dropout/feature selection in the ANNs, to yield a performance increase on a network that has not learned any meaningful information about the data set. This is because the measures rely on artefacts of the trained network, namely it's weights and activation, in order to determine which features to select.

In both CNN experiments we have also found minimal performance gain from the use of our functional-measure-based dropout. However, we note that in the case of the DNN, the BlockDrop extension based on Angle Measure resulted in a significantly high level of dropout (in some cases layers reported dropout rates of $> 99\%$ whilst still maintaining the level of prediction corresponding to that of no dropout).

We observe that in our results, there was a bias towards the classification of posed Smiles. This is likely a result of the class imbalance in the timeseries dataset. In the case of the Basic CNN the original 5/1 posed/real ratio without extended dropout was reduced to an 3/1 ratio with a max_block size of 100. This indicates that the dropout may have some influence in reducing the over-fitting of the data.

In comparison to the 96.1% accuracy achieved with an ensemble of KNN, SVM and NN classifiers on the 3 physiological responses [19], the results obtained in this experiment are poor. However, we cannot make a fair comparison between the results of these two papers, due to the fact that the classifiers used had differing levels of information available and used different classification architectures. We do observe that with 24 observers the authors of the given paper were able to achieve an average accuracy of over 75% for neural network classifiers; which indicates that it may be plausible to obtain non-trivial predictions from the subset of observers provided. This creates space for further work on this limited dataset with improved hyperparameter tuning.

8 Future Work

There were a limited number of epochs for which each network was trained. We found in a set of separate series experiments that the network was able to obtain an accuracy of 60% when evaluated with LOSO cross-validation if an individual network with 1 hidden layer and SGD activation was run for over 50000 epochs. At present we are unable to run such an experiment on all optimizers, models and techniques due to resource limitations.

As this work made use of only features that had already been extracted and pre-processed from the time series by another party, another direction of future work would be to investigate whether performance increases could be obtained by extracting more descriptive features from the dataset.

It would also be beneficial to have baseline comparison to feature selection performed using statistical methods on the training set. For instance, training could be performed on just the features min, iqr, skw, samp en, ap en, and fuzzy en, which are the values with the greatest value of F for the ANOVA test [26].

The current technique for data extrapolation maintains a linear relationship to original datapoints. Research indicates that when generating extra training data, it can be useful to add noise [28]. Other techniques for data extrapolation could also be applied, including doing extrapolation of the timeseries' and then calculating the features following this.

A possible new direction for research would be to investigate whether the angle based measure can be used to remove features/neurons that are capturing unwanted relationships. In the case of this experiment, it would be those that cause signals coming from the same entity which have a higher likely hood of being classed together, even if one signal is in response to a real smile and the other is in response to a posed one.

It would also be worthwhile investigating whether an ensemble of the ANNs and CNNs produced in this paper could be used as an ensemble classifier.

We also note that in the case of the InceptionTime model, the calculated hyperparameter for depth was the largest value our resources would allow. It would be worth repeating this hyperparameter selection with a wider range of options for the depth parameter and produce new results with the new value for depth.

9 Appendix

Fig. 9. The average pupil dilation for each observer when observing fake and real smiles.

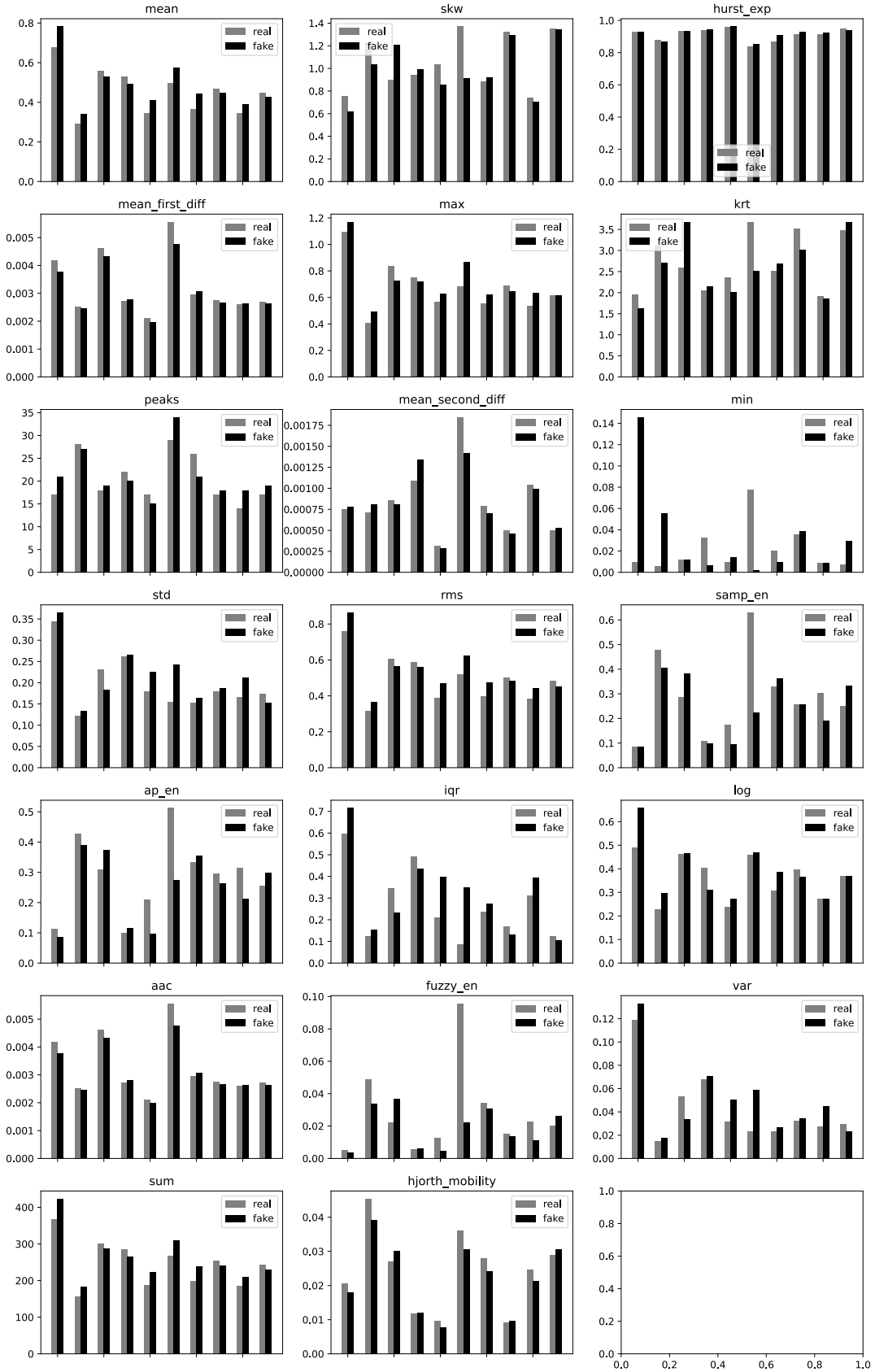


Fig. 10. Feature analysis for average results each observer when observing Fake and Real smiles.

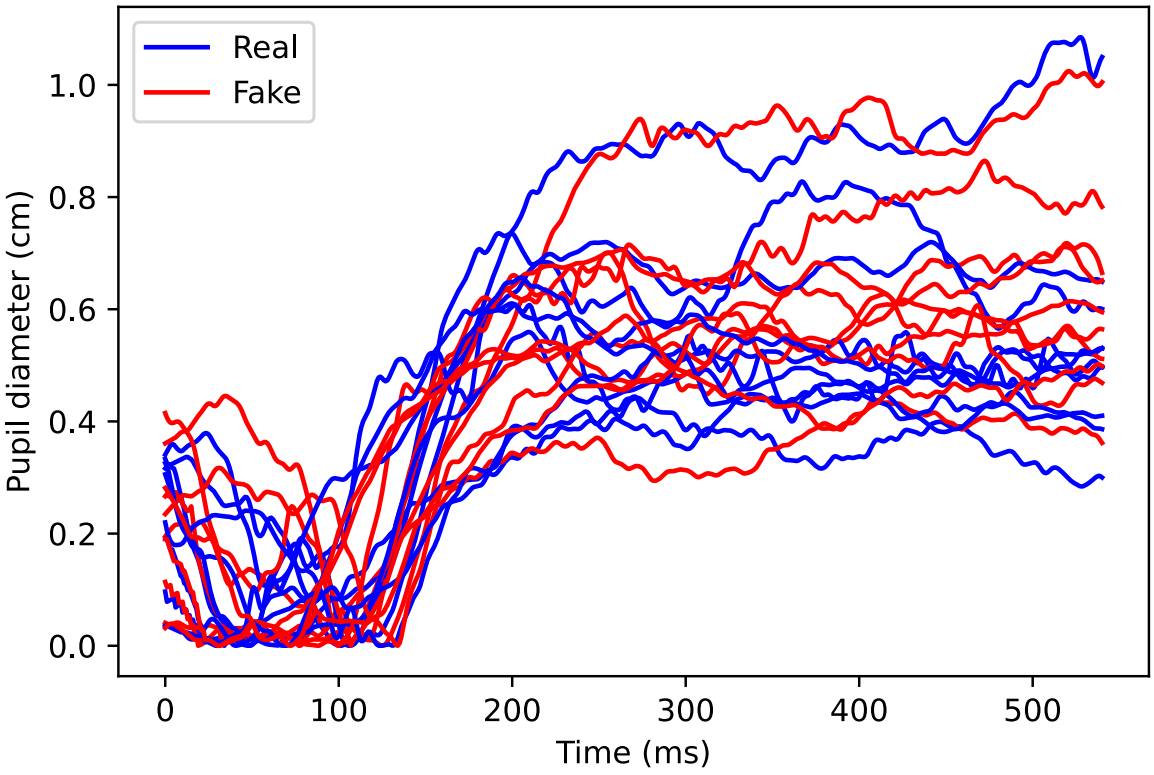


Fig. 11. Timeseries data for Pupillary responses to Real and Fake smilers for all participants

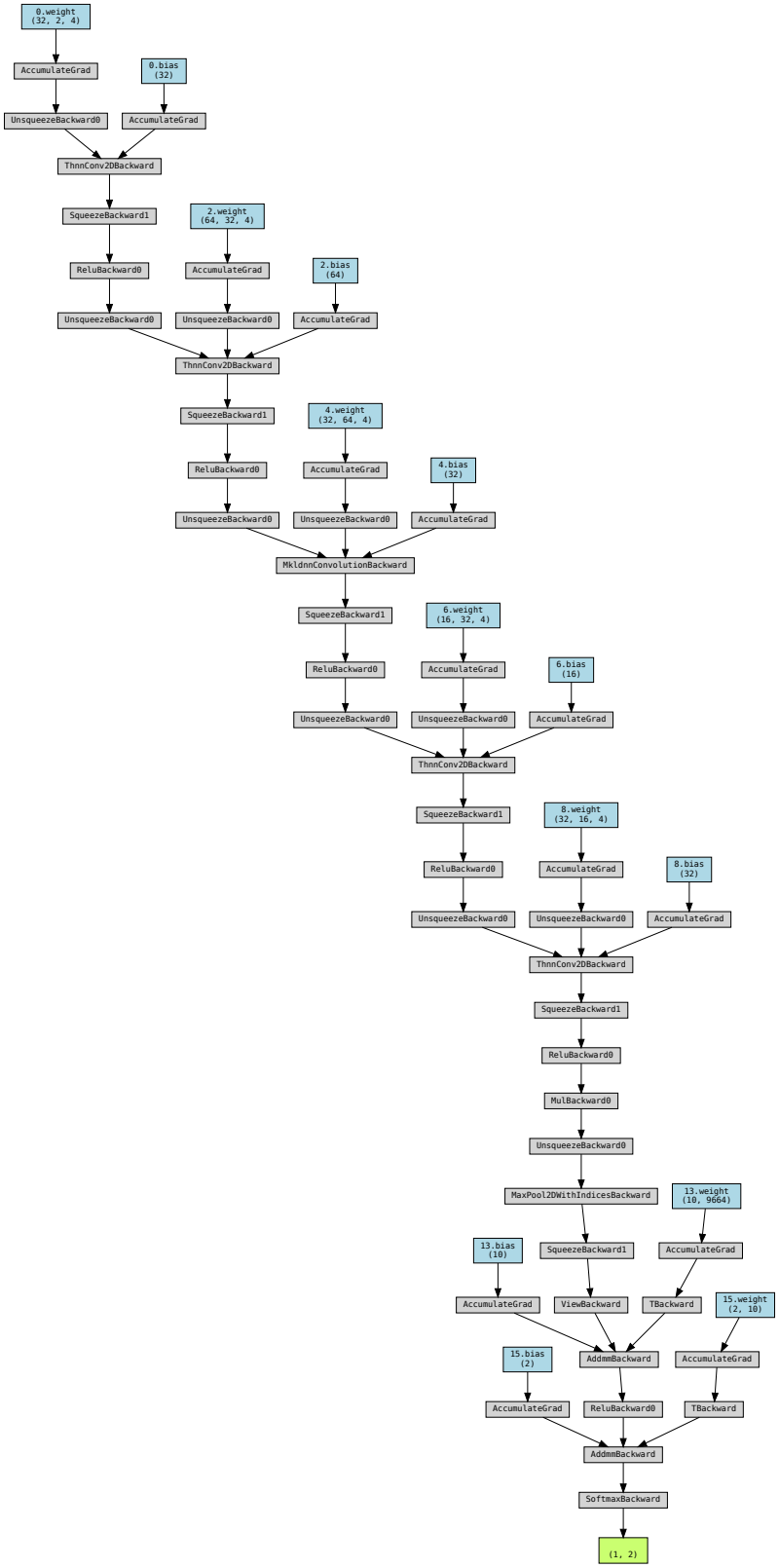


Fig. 12. Chosen CNN architecture

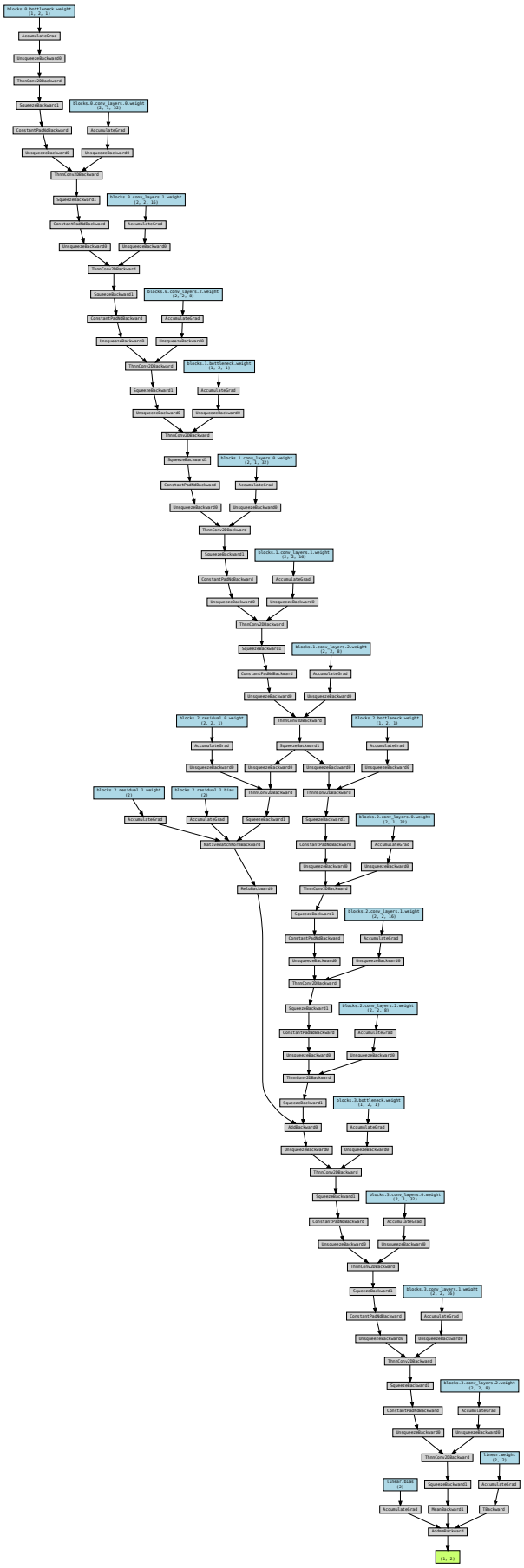


Fig. 13. Chosen InceptionTime architecture

References

1. Bagnall, A., Lines, J.: An experimental evaluation of nearest neighbour time series classification. arXiv preprint arXiv:1406.4757 (2014)
2. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data mining and knowledge discovery **31**(3), 606–660 (2017)
3. Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-series classification with cote: the collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering **27**(9), 2522–2535 (2015)
4. Cai, S., Shu, Y., Chen, G., Ooi, B.C., Wang, W., Zhang, M.: Effective and efficient dropout for deep convolutional neural networks. arXiv preprint arXiv:1904.03392 (2019)
5. Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., Batra, D.: Reducing overfitting in deep networks by decorrelating representations. arXiv preprint arXiv:1511.06068 (2015)
6. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019)
7. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences **239**, 142–153 (2013)
8. Ekman, P.: Strong evidence for universals in facial expressions: a reply to russell's mistaken critique. (1994)
9. Faust, O., Hagiwara, Y., Hong, T.J., Lih, O.S., Acharya, U.R.: Deep learning for healthcare applications based on physiological signals: A review. Computer methods and programs in biomedicine **161**, 1–13 (2018)
10. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019)
11. Fawaz, H.I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (Sep 2020). https://doi.org/10.1007/s10618-020-00710-y, https://doi.org/10.1007/s10618-020-00710-y
12. Garson, D.G.: Interpreting neural network connection weights (1991)
13. Gedeon, T.D.: Data mining of inputs: analysing magnitude and functional measures. International Journal of Neural Systems **8**(02), 209–218 (1997)
14. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. arXiv preprint arXiv:1810.12890 (2018)
15. Gong, P., Ma, H.T., Wang, Y.: Emotion recognition based on the multiple physiological signals. In: 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR). pp. 140–143. IEEE (2016)
16. Gronau, Q.F., Wagenmakers, E.J.: Limitations of bayesian leave-one-out cross-validation for model selection. Computational brain & behavior **2**(1), 1–11 (2019)
17. Hagan, M.T., Demuth, H.B., Beale, M.H., De, J.O.: Neural network design. s. n. (2016)
18. Hoque, M., Morency, L.P., Picard, R.W.: Are you friendly or just polite?–analysis of smiles in spontaneous face-to-face interactions. In: International Conference on Affective Computing and Intelligent Interaction. pp. 135–144. Springer (2011)
19. Hossain, M.Z., Gedeon, T.D.: An independent approach to training classifiers on physiological data: An example using smiles. In: International Conference on Neural Information Processing. pp. 603–613. Springer (2018)
20. Kuhn, M., Johnson, K.: Feature engineering and selection: A practical approach for predictive models. CRC Press (2019)
21. Kupinski, M.A., Giger, M.L.: Feature selection with limited datasets. Medical Physics **26**(10), 2176–2182 (1999)
22. Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery **29**(3), 565–592 (2015)
23. Littmann, E., Ritter, H.: Cascade network architectures. In: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. vol. 2, pp. 398–404 vol.2 (1992). https://doi.org/10.1109/IJCNN.1992.226955
24. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30, p. 3. Citeseer (2013)
25. Maweu, B.M., Dakshit, S., Shamsuddin, R., Prabhakaran, B.: Cefes: A cnn explainable framework for ecg signals. Artificial Intelligence in Medicine **115**, 102059 (2021)
26. Miller Jr, R.G.: Beyond ANOVA: basics of applied statistics. CRC press (1997)
27. Milne, L.: Feature selection using neural networks with contribution measures. In: AI-CONFERENCE-. pp. 571–571. Citeseer (1995)
28. Murphey, Y.L., Guo, H., Feldkamp, L.A.: Neural learning from unbalanced data. Applied Intelligence **21**(2), 117–128 (2004)
29. Nikolenko, S., Kadurin, A., Arkhangelskaya, E.: Deep learning. SPb.: Peter (2018)
30. Pei, Z., Li, C., Qin, X., Chen, X., Wei, G.: Iteration time prediction for cnn in multi-gpu platform: Modeling and analysis. IEEE Access **7**, 64788–64797 (2019). https://doi.org/10.1109/ACCESS.2019.2916550
31. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (Oct 1986). https://doi.org/10.1038/323533a0, https://doi.org/10.1038/323533a0
32. Silva, D.F., Giusti, R., Keogh, E., Batista, G.E.: Speeding up similarity search under dynamic time warping by pruning unpromising alignments. Data Mining and Knowledge Discovery **32**(4), 988–1016 (2018)

33. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
34. Wong, P.M., Gedeon, T.D., Taggart, I.J.: An improved technique in porosity prediction: a neural network approach. IEEE Transactions on Geoscience and Remote Sensing **33**(4), 971–980 (1995)
35. Ye, L., Keogh, E.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data mining and knowledge discovery **22**(1), 149–182 (2011)
36. Zhang, Z., Sabuncu, M.R.: Generalized cross entropy loss for training deep neural networks with noisy labels. arXiv preprint arXiv:1805.07836 (2018)
37. Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics **28**(1), 162–169 (2017)