A ResNet Autoencoder Approach for Time Series Classification of Cognitive State

Paul Terwilliger, Jack Sarle, Shannon Walker
Booz Allen Hamilton
Norfolk, VA
paul.terwil@gmail.com, sarle_john@bah.com, walker_shannon@bah.com

Angela Harrivel, Ph.D.
NASA Langley Research Center
Hampton, VA
angela.r.harrivel@nasa.gov

ABSTRACT

Time series classification (TSC) is the problem of predicting class labels at discrete intervals along a time series. Although there are many approaches to TSC, Convolutional Neural Network (CNN) models have been rising in popularity. If a CNN trains on a fixed-length window of time of length n time-steps and there is overlap between windows of time, the problem of overfitting is exacerbated. Traditional transformations to fix this problem include transforming the data into the frequency domain or using exponential smoothing. In this paper, we present a convolutional autoencoder approach, wherein the fixed-length sliding window of time is transformed using a deep convolutional autoencoder. A separate model is then trained on an encoded representation.

For our work, we are given time series data from the Crew Systems and Aviation Operations Branch at NASA Langley Research Center: biometric data of pilot test subjects as they are flying a simulation. At each time step, the test subjects are classified as being in one of four one-hot encoded cognitive states: no event, channelized attention, diverted attention, and startled/surprised. Preliminary results suggest that our convolutional autoencoder approach allows us to predict at a lower log-loss than other approaches, particularly in the case of grossly overlapping windows of time, thereby mitigating the effect of overfitting.

ABOUT THE AUTHORS

Paul Terwilliger is a Data Scientist with Booz Allen Hamilton and Chess.com. He holds a B.S. in Physics from the University of Pennsylvania. His research interests include building chess-related machine learning models and testing novel data science ideas.

Jack Sarle is a Staff Scientist with Booz Allen Hamilton. He holds a B.S. in Computer Science Engineering from North Carolina State University. His research interests include data science and DevOps automation techniques.

Shannon Walker is a Lead Associate with Booz Allen Hamilton. He holds a B.S. in Aerospace Engineering from Virginia Tech and an M.S. in Applied Physics from the Air Force Institute of Technology. His research interests include astrophysics, earth sciences, and data science.

Dr. Angela Harrivel is a Biomedical Engineer and leads the Human Performance and Monitoring Team in the Crew Systems and Aviation Operations branch at NASA Langley. She holds a M.S. in Physics from John Carroll University and a Ph.D. in Biomedical Engineering from the University of Michigan. She is an expert in functional neuroimaging and served as the Technical Lead for the Commercial Aviation Safety Team (CAST) research safety enhancement entitled “Training for Attention Management.”

A ResNet Autoencoder Approach for Time Series Classification of Cognitive State

Paul Terwilliger, Jack Sarle, Shannon Walker
Booz Allen Hamilton
Norfolk, VA
terwilliger_paul@bah.com, sarle_john@bah.com, walker_shannon@bah.com

Angela Harrivel, Ph.D.
NASA Langley Research Center
Hampton, VA
angela.r.harrivel@nasa.gov

1. INTRODUCTION

The largest category of loss of control – inflight fatalities stem from a loss of airplane state awareness due to ineffective attention management on the part of pilots who may be distracted or in other cognitive states that may lead to errors and human performance decrement during the flight (Harrivel et al., 2017). In effort to prevent future flight-related accidents, incidents and fatalities, we work with the Crew State Monitoring (CSM) team of NASA Langley Research Center's (LaRC) Crew Systems and Aviation Operations Branch (CSAOB). Together, we aim to predict losses of airplane state awareness in order to intervene before situations of increased risk arise. Improvements in class label prediction accuracy support the future use of biometric time series data to provide information regarding the state of the pilot to aid time-critical decision making.

The CSM team researches real-time changes in aircrew psychophysiological attentional states (crew state) through biophysical sensors and machine learning algorithms, with their primary platform being the CSM psychophysiological monitoring system (CSM system). We work with the CSM team to classify different cognitive states that pilot test subjects experience as they fly aircraft simulators. For this task, we are supplied with Scenarios for Human Attention Restoration using Psychophysiology (SHARP)-1 data by the CSM team with a time series dataset.

1.1 Data

All the SHARP-1 data for this research come from biological sensors applied to 18 test subjects. The data are provided in sets of four for each test subject. Three datasets contain the labeled benchmark data for each of the non-nominal or event classes during the performance of psychological tasks (Harrivel et al., 2016): channelized attention (CA), diverted attention (DA), and startled/surprised (SS). Each of the benchmark data files also contain labeled nominal or no event (NE) data and are about six minutes in length. The last file contains the line-oriented flight training (LOFT) experiment file for each of the test subjects, during which scenario events induced the states of interest (Stephens et al., 2017). This file contained about one hour's worth of data and contains labeled instances of each of the four classes (SS, CA, DA, and NE).

The data consist of 25 dimensions. The first two are a time stamp and an event label. Three of the features come from electrocardiogram (ECG), respiratory (R), and galvanic skin response (GSR) sensors. The remaining 20 come from electroencephalogram (EEG) sensors.

There are approximately 2.3×10^7 data points and the dataset is heavily class imbalanced. The nominal class comprises about 83% of the data. The three non-nominal classes are as follows: channelized attention comprises about 14%, diverted attention comprises about 2%, and startled/surprise comprises 1% of the data.

1.2 Data preprocessing

We find that, when predicting a cognitive state based on a time window, the global time stamp is a high predictor for the cognitive state, regardless of the rest of the data in a time window. Therefore, we discard the global time stamp column.

As a baseline, we try predicting cognitive states using only a single frame of time using a variety of machine learning techniques: signal smoothing to preprocess, then Light Gradient Boosting Method (LightGBM) (Ke and Meng, 2017) or a shallow neural network. We do not find success. In fact, it is common (Fawaz et al., 2019) in TSC to take some past time frames and (if possible) some future time frames into account when predicting. Given a time frame to predict on, we choose to take N past and N future time frames into account, creating a window of time of size 2N+1. Throughout this paper, we refer to this set of time frames around and including the sample as a time window.

When considering two different time windows, it is possible for the two time windows to contain one (often many) of the same time frames. Throughout this paper, we call this overlap. Two time windows overlap if they contain at least one shared time frame, and overlap increases the more shared time frames they contain.

In order to simulate real-life experiences as best as possible, we only predict on future states. We create train/test splits for each test subject such that all test time windows are temporally after all train time windows. Although train/test time windows may overlap within their respective sets, there is no overlap between a train and a test time window. Time windows can be shuffled within their respective sets.

Due to the data imbalance, we consider resampling strategies for imbalanced datasets. Resampling is a widely adopted technique for dealing with highly unbalanced datasets. We find sufficient success when we randomly undersample our dataset until our classes are balanced, so we choose to undersample our data (Figure 1).

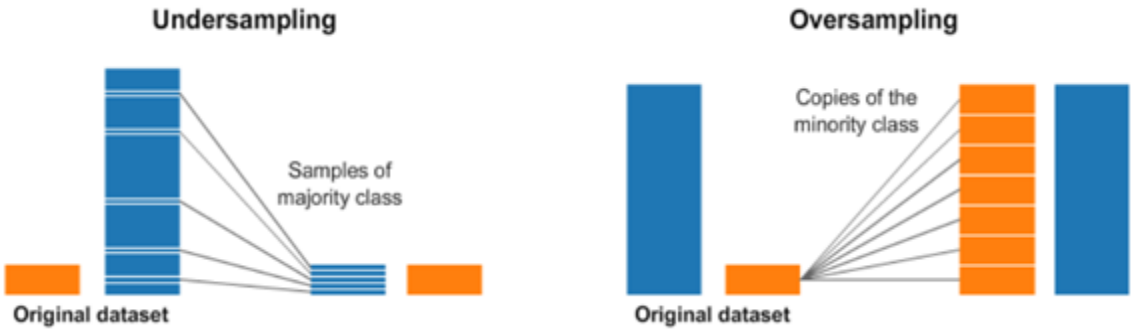

Figure 1. (Left) In undersampling, the simplest technique involves removing random records from the majority class, which can cause loss of information. (Right) The simplest implementation of oversampling is to duplicate random records from the minority class, which can cause overfitting on the duplicated records (Badur, 2019).

After undersampling, due to our objective of detecting a loss of airplane state awareness, we aggregate our four classes into two classes: Event or No Event. Throughout the rest of the paper, we only predict on these two classes.

1.3 Problem Statement

We find that using a naïve approach of classifying each time window without any processing as an input to a Convolutional Neural Net (CNN) model leads to extreme overfitting. This is because of two interacting problems:

1. Overlapping time windows share a number of time frames. The model will see the same time frame many times before finishing even the first epoch.
2. Due to the nature of our data, adjacent time windows often have the same classification. Therefore, a convolutional kernel will slide over the same time window with the same classification many times, and it will learn to correlate the two, which hinders generalization.

We want to take large time windows of N=128 in order to properly account for past and future events. With such a high number of time frames in each time window, assuming that there is no undersampling, the model will see each time stamp 2N+1 times (ignoring boundary behaviors), causing overfitting to happen 2N+1 times faster. This

causes overfitting to occur before the first epoch even finishes. Our problem is to build a model that classifies on time windows while avoiding this overfitting problem.

There are four methods of resolving the overfitting problem that we consider and discard:

1. Resolving the overfitting problem by gathering more data is costly and therefore impractical. Gathering more data relies upon collecting benchmark and LOFT data from several human test subjects who may or may not be compensated for their time. Much of the data must be discarded because data is fraught with noise as it is collected through biophysical sensors.
2. Resolving the overfitting problem by undersampling data such that time windows do not overlap is not practical due to the already small amount of data we are given. Data sparsity is further exacerbated by the data imbalance present in our dataset – the amount of startle/surprise data is very sparse at 1%. After undersampling, we have a dataset that contains about one one-hundredth of the original data.
3. Resolving the overfitting problem by removing the use of time windows and using only present time stamps significantly reduces predictive power. In order to predict with a high accuracy, we would like to take past and future data into account.
4. Resolving the overfitting problem by instead training on hand-extracted features is time-intensive, prone to human error, and costly.

The problem we face is: how do we train on overlapping time windows without overfitting? There is evidence that sparse representations of datasets using sparse autoencoders improve performance on classification tasks (Makhzani and Frey, 2014). We are inspired by this and we find that training a neural network on a latent representation of a time window using a Residual Network (ResNet) (He et al., 2015) autoencoder sufficiently reduces the overfitting problem.

2. RELATED WORK

2.1 Cognitive state prediction

For a thorough discussion of the use of psychophysiological feedback toward both human and system adaptation for the optimization of performance, or Biocybernetic Adaptation, the reader may refer to Stephens, Dehais and Pope, 2018. For the application of cognitive state prediction to commercial aviation pilot training for attention management, please see Harrivel and Liles, 2016, Stephens and Prinzel, 2017, and Harrivel and Stephens, 2017.

2.2 Autoencoders as an unsupervised method

Autoencoders are a type of autoassociative neural network that aim to learn efficient data codings in an unsupervised manner (Kramer, 1991). When an autoencoder is properly trained, it achieves a nonlinear dimensionality reduction where it learns a representation of the dataset while ignoring signal noise.

In our autoencoders, we aim to predict the same input to its output. Given enough nodes in each layer, there is a trivial solution where the network can learn the identity function to perfectly map input to outputs. In our autoencoders, we use a "bottleneck" hidden layer – a hidden layer with few enough nodes that the identity function is impossible to learn – as depicted in Figure 2 below. This sparsity forces the model to respond to the unique statistical features of the data. With our ResNet approach, we constrain the hidden layer using a combination of a limited number of filters and a limited dimension size.

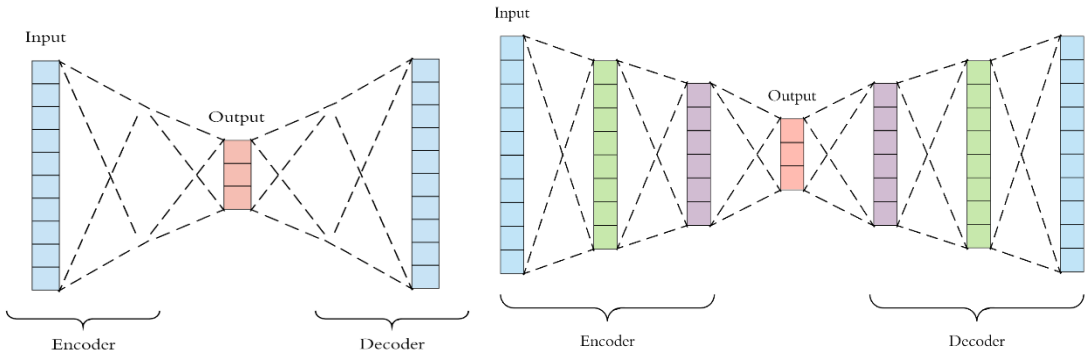

Figure 2. Bottleneck autoencoder structures. (Left) The method uses an abrupt change in model size (Stewart, 2019). (Right) Because of our CNN structure, we gradually decrease and increase model size to account for local pooling (Dertat, 2017).

2.3 LSTM Autoencoders in Time Series projects

In 2015, Long Short-Term Memory (LSTM) autoencoders were used to find latent representations of a time series video task (Srivastava et al., 2015). However, we encounter difficulty using LSTMs in three key areas:

1. LSTMs are designed mainly to predict an output for each time stamp in the time window, where we want to predict only once (Längkvist et al., 2014)
2. Despite the success of LSTMs over Recurrent Neural Networks (RNN), LSTMs still suffer from the vanishing gradient problem due to training on long time series (Nugaliyadde et al., 2019) (Pascanu et al., 2012)
3. LSTMs are difficult to train and parallelize (Pascanu et al., 2013)

Therefore, we choose to use a CNN model, which addresses all three problems (Fawaz et al., 2019).

2.4 Convolutional Neural Networks in Time Series projects

In a study in May 2019, CNNs were found to be the most widely applied architecture to the TSC problem by Fawaz and colleagues. We use a couple of common improvements on the CNN architecture. By implementing skip-connections, we build a ResNet which helps alleviate the vanishing/exploding gradient problem. By implementing squeeze-excitation (Hu et al., 2017) on each CNN layer, we recalibrate channel-wise features by explicitly modeling interdependencies between channels.

We aim to use the benefits of both bottleneck autoencoders and ResNets to build a hybrid classifier.

3. A RESNET AUTOENCODER-CLASSIFIER MODEL TO ALLEVIATE OVERFITTING

The bottleneck layer of our autoencoder learns a latent representation that encodes unique statistical features while ignoring noise. We hypothesize that, by making the bottleneck of the autoencoder sparse enough, we force the latent representation to contain the core statistical features present in each time window while not only ignoring statistical noise, but also ignoring whatever causes the network to overfit.

We build a ResNet autoencoder with a bottleneck layer (referred to just as a ResNet autoencoder or just autoencoder) by having a sparse hidden layer near the center of the CNN that reduces dimension size through local average pooling and by reducing filter size. We then reconstruct the original time series by gradually increasing the dimension size via reversing the local average pooling and increasing filter size. Local average pooling is used instead of local max pooling because of the reconstruction step. After training the ResNet autoencoder until it hits early stopping on a train/test split using a Mean Square Error loss function, we transform the data through the sparse ResNet autoencoder into a latent representation via the bottleneck layer. The latent representation is then flattened,

paired with original one-hot encoded labels, and fed into a shallow dense neural network for learning classification. The exact model architecture can be seen in Appendix A.

In order to show the improvement of our ResNet Autoencoder-Classifier method, we use two other models as a baseline. To show the reduction in overfitting, we use a ResNet Classifier as a baseline (structure in Appendix B). To show the increased prediction power of using an entire time window, we train a Dense Classifier (structure in Appendix C). The Dense Network is a fully-connected neural network that is only trained on the present step - it is not given an entire time window.

4. EXPERIMENTS

We evaluate our method on the SHARP-1 data provided by the CSM team. The models are trained on 883,616 samples with four classes. Roughly 25% of the data is held for validation, and another 25% of the data is held for testing. Test time windows are in the future of train time windows and have no overlap. We obtain a final result on each model and compare log-loss values (Figure 3) and confusion matrices (Figure 4).

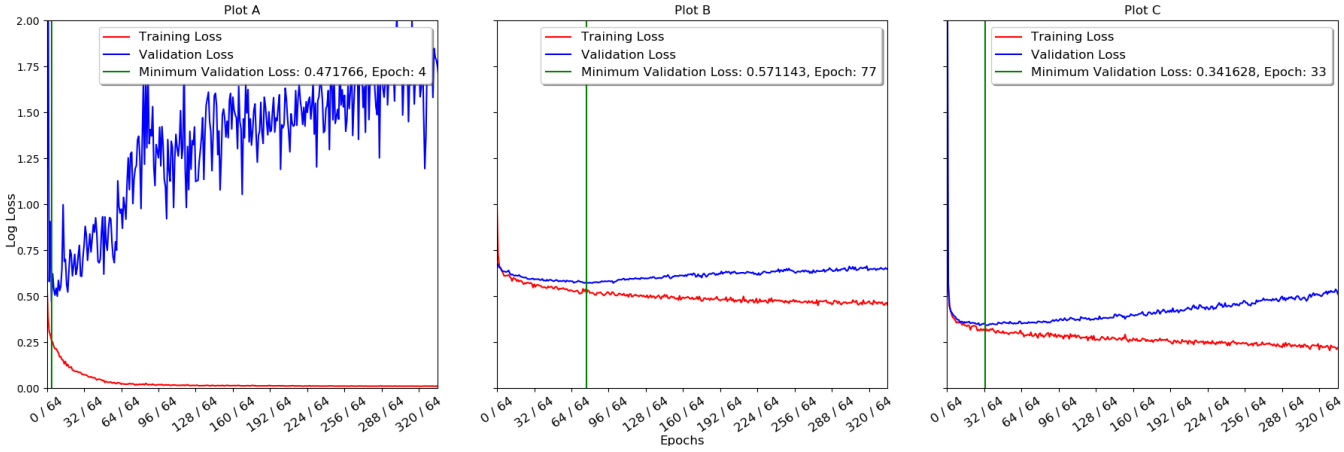

Figure 3. (A) ResNet Classifier. (B) Dense Classifier. (C) ResNet Autoencoder-Classifier. Since overfitting occurs before a single epoch is finished, we report train/test loss values every 1/64th of an epoch. We report the minimum validation loss to show where overfitting begins as epochs increase (at the green line) and what the value of loss is at that point.

With only 4/64th of an epoch finished on the ResNet Classifier (Plot A, left), the ResNet Classifier begins overfitting very early, as expected. We remove the overfitting behavior in the Dense Classifier (Plot B, middle) by training on a single time frame instead of the entire time window. However, we see that the ResNet Classifier has a better validation loss (0.47) than the Dense Classifier (0.57), which shows the predictive power that is lost by removing past and future time frames from the prediction. The ResNet Autoencoder-Classifier (Plot C, right) is able to learn for eight times more epochs than the ResNet model achieving a loss of 0.34, better than both baselines.

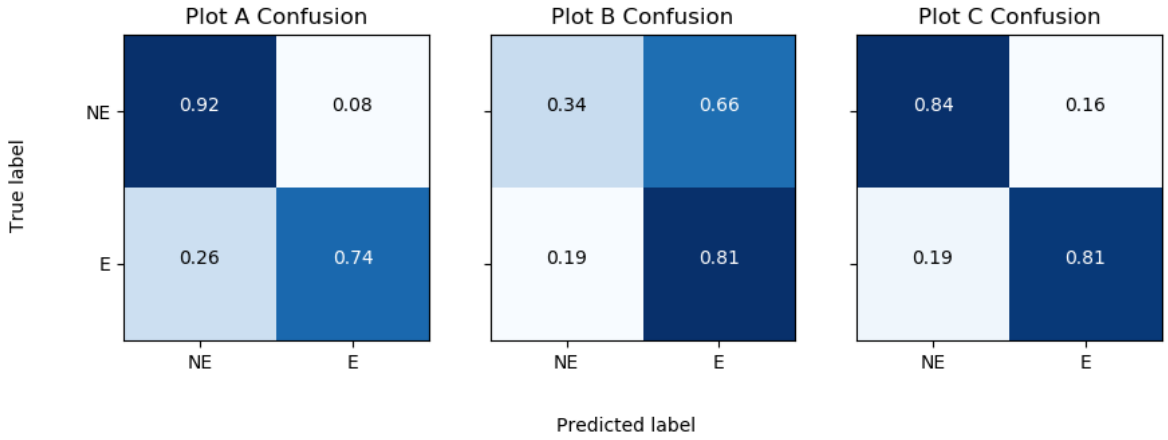

Figure 4. (A) ResNet Classifier. (B) Dense Classifier. (C) ResNet Autoencoder-Classifier. Confusion matrices have true labels on the y-axis and predicted labels on the x-axis.

Confusion matrices are generated to analyze true positives, false positive, and false negatives. We find that our ResNet Autoencoder-Classifier model (Plot C, right) produces a low rate of false positives and false negatives, showing that we can accurately predict whether there is an Event or No Event. Our ResNet Classifier (Plot A, left) shows similarly strong results, but we consider the unstable behavior of the loss plot (Figure 3, Plot A) and the higher loss value to mean that the ResNet Classifier performs worse than the ResNet Autoencoder-Classifier. The Dense Classifier (Plot B, middle) performs the worse, adding credence to our hypothesis that adding temporal features from the past and future time stamps is important when making predictions.

5. SUMMARY

Our ResNet Autoencoder-Classifier for encoding time series datasets is shown to be more successful than the two baselines, mitigating overfitting on overlapping time windows. With the success we have achieved in predicting cognitive states, we are hopeful that we can improve on this method.

6. ACKNOWLEDGEMENTS

Special thanks goes to Jack Sarle for his work in engineering the models. Jack was given theoretical designs and from them he coded the models, performed hyperparameter tuning, adjusted the models to incorporate newer methods, gathered statistics, and performed debugging. Without his work we would not have the results in this paper.

References:

Badur, W. (2019). Having and Imbalanced Dataset? Here Is How You Can Fix It. Towards Data Science. Retrieved from https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb

Dertat, A. (2017). Applied Deep Learning - Part 3: Autoencoders. Towards Data Science. Retrieved from https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. (2019). Deep Learning for Time Series Classification: a Review. Data Mining and Knowledge Discovery, 33, 917–963. https://doi.org/10.1007/s10618-019-00619-1

Funk, S. (2015). RMSprop Loses to SMORMS3 - Beware the Epsilon! Sifter.org. Retrieved from https://sifter.org/~simon/journal/20150420.html

Harrivel, A., Liles, C., Stephens, C., Ellis, K., Prinzel, L., Pope A. (2016). Psychophysiological Sensing and State Classification for Attention Management in Commercial Aviation. AIAA Science and Technology Forum and Exposition 2016. Oral Session: SEN-02, Novel Sensor Systems and Sensing Techniques II, January 6, 2016.

Harrivel, A., Stephens, C., Milletich, R., Heinich, C., Last, M., Napoli, N., Pope, A. (2017). Prediction of Cognitive States During Flight Simulation using Multimodal Psychophysiological Sensing. AIAA SciTech 2017, Applications of Sensor and Information Fusion, January 11, 2017, Grapevine, Texas.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv:1512.03385

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2017). Squeeze-and-Excitation Networks. arXiv:1709.01507

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., . . . Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems, 30. Retrieved from https://papers.nips.cc/book/advances-in-neural-information-processing-systems-30-2017

Längkvist, M., Karlsson, L., Loutfi, A. (2014). A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling. Pattern Recognition Letters, 42, 11–24.

Makhzani, A., & Frey, B. (2014). k-Sparse Autoencoders. arXiv:1312.5663v2

Mark A. Kramer, M. (1991). Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. AIChE Journal, volume 37, issue 2. https://doi.org/10.1002/aic.690370209

Nugaliyadde, A., Wong, K., Sohel, F., & Xie, H. (2019). Language Modeling through Long Term Memory Network. arXiv:1904.08936

Pascanu, R., Mikolov, T., & Bengio, Y. (2012). Understanding the Exploding Gradient Problem. ArXiv 1211.5063

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the Difficulty of Training Recurrent Neural Networks. Proceedings of the 30th International Conference on Machine Learning, 28, III-1310 – III-1318.

Ramachandran, P., Zoph, B., & Le, Q. (2017). Searching for Activation Functions. arXiv:1710.05941

Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised Learning of Video Representations using LSTMs. arXiv:1502.04681

Stephens, C., Dehais, F., Roy, R., Harrivel, A., Last, M., Kennedy, K., Pope, A. (2018). Biocybernetic Adaptation Strategies: Machine Awareness of Human Engagement for Improved Operational Performance. Schmorrow D., Fidopiastis C. (eds) Augmented Cognition: Intelligent Technologies. AC 2018. Lecture Notes in Computer Science, vol 10915.

Stephens, C., Prinzel, L., Harrivel, A., Comstock, J., Abraham, N., Pope, A., Kiggins, D. (2017). Crew State Monitoring and Line-Oriented Flight Training for Attention Management. 19th International Symposium on Aviation Psychology (ISAP 2017). Retrieved from https://ntrs.nasa.gov/search.jsp?R=20170005473

Stewart, M. (2019). Comprehensive Introduction to Autoencoders. Towards Data Science. Retrieved from https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

APPENDIX A: STRUCTURE OF THE RESNET AUTOENCODER-CLASSIFIER

The autoencoder ResNet classifier uses two models. The first model (Appendix A.1) creates a latent representation of a time window using an autoencoder method. The second model (Appendix A.2) takes, as its input, the latent representation and produces a classification.

Throughout our network we use a Swish activation function (Ramachandran and Zoph, 2017), where Swish(x) = x * sigmoid(x). In our case, Swish activation functions give better validation loss values than ReLU.

Throughout our networks, we use the SMORMS3 optimizer (Funk, 2015). We find that the SMORMS3 optimizer trains faster than other optimizers we have tried, including Adam, RMSProp, and Stochastic Gradient Descent.

A.1 The Autoencoder for the ResNet Autoencoder-Classifier

In our ResNet Blocks, we use 1-dimensional convolutional layers (across time) with the following structure:

- Convolutions have 64 filters
- Kernel size is 9
- Stride is 1
- Layers are padded with zeros such that the output shape is the same as the input shape (padding=same)
- L2 Regularization is 10^{-5}

The result of each 1-dimensional convolution has a squeeze-excitation operation added to the end of it:

- Global Average pooling is applied on our input, producing V_0
- V_0 is run through a fully connected layer with 4 nodes and a Swish activation, producing V_1
- V_1 is run through a fully connected layer with 64 nodes and a sigmoid activation, producing V_2
- Each filter in the input layer is multiplied by each node in V_2

We then apply two more operations: we batch-normalize the result and then apply a Swish activation function.

Each ResNet Block is a 2-layer Convolutional Neural Net where each convolutional layer has a squeeze-excitation operation applied (defined above), batch normalization, and a Swish activation function in that order, producing S. The input is then summed with S, producing a skip-connection.

The ResNet Autoencoder has the following structure:

1. A convolution is applied with a kernel size of 9, stride of 1, L2 regularization of 10^{-5}, a Swish activation function, and zero-padding to keep the shape the same. This convolution is to bring the number of filters up to 64, so that ResNet Blocks may be applied.
2. 5 Residual Blocks are applied
3. 1-dimensional average pooling with size of 2 is used to bring the width down to 128
4. 5 Residual Blocks are applied
5. 1-dimensional average pooling with size of 2 is used to bring the width down to 64
6. 5 Residual Blocks are applied
7. 1-dimensional average pooling with size of 2 is used to bring the width down to 32
8. 5 Residual Blocks are applied
9. 1-dimensional average pooling with size of 2 is used to bring the width down to 16
10. 5 Residual Blocks are applied
11. 1-dimensional average pooling with size of 2 is used to bring the width down to 8
12. 5 Residual Blocks are applied
13. A convolution is applied with 64 filters, a kernel size of 9, stride of 1, L2 regularization of 10^{-5}, a Swish activation function, and zero-padding to keep the shape the same. This is our "squeeze" layer.
14. A convolution is applied with 64 filters, a kernel size of 9, stride of 1, L2 regularization of 10^{-5}, a Swish activation function, and zero-padding to keep the shape the same.

15. We undo 1-dimensional average pooling by performing a version of upsampling where each upsampled value equals the original value. This brings the width up to 16
16. 5 Residual Blocks are applied
17. We undo 1-dimensional average pooling by performing a version of upsampling where each upsampled value equals the original value. This brings the width up to 32
18. 5 Residual Blocks are applied
19. We undo 1-dimensional average pooling by performing a version of upsampling where each upsampled value equals the original value. This brings the width up to 64
20. 5 Residual Blocks are applied
21. We undo 1-dimensional average pooling by performing a version of upsampling where each upsampled value equals the original value. This brings the width up to 128
22. 5 Residual Blocks are applied
23. We undo 1-dimensional average pooling by performing a version of upsampling where each upsampled value equals the original value. This brings the width up to 256
24. 5 Residual Blocks are applied
25. A convolution is applied with the original number of filters, a kernel size of 9, stride of 1, L2 regularization of 10^{-5}, a Swish activation function, and zero-padding to keep the shape the same.

Training is done with a loss of mean-squared error.

A.2 The Classifier for the ResNet Autoencoder-Classifier

For our classifier, we use a 3-layer fully-connected neural network. Inputs from the squeeze layer are of shape (8, 64) due to the size of the squeeze layer, and then they are flattened as input to the fully-connected neural network. The first two layers (the hidden layers) use 512 nodes, a Swish activation function, and a L2 regularization term of 10^{-5}. The last layer (the classification layer) has 2 nodes and uses a softmax activation function. We use a sparse-categorical-crossentropy loss for training.

APPENDIX B: STRUCTURE OF THE RESNET CLASSIFIER

ResNet Blocks for the ResNet Classifier are identical to the ResNet blocks in the ResNet Autoencoder Classifier (A.1).

1. A convolution is applied with 64 filters, a kernel size of 9, stride of 1, L2 regularization of 10^{-5}, a Swish activation function, and zero-padding to keep the shape the same.
2. 5 Residual Blocks are applied.
3. A convolution is applied with 1 filter, a kernel size of 1, stride of 1, L2 regularization of 10^{-5}, a Swish activation function, and zero-padding to keep the shape the same.
4. A fully-connected dense layer is applied (on flattened inputs) with 2 nodes and a softmax activation function. We use a sparse-categorical-crossentropy loss for training.

Sparse-categorical-crossentropy is used as a loss function for training.

APPENDIX C: STRUCTURE OF THE DENSE CLASSIFIER

For our classifier, we use a 3-layer fully-connected neural network. Inputs are of a single time frame. The first two layers (the hidden layers) use 1024 nodes, a Swish activation function, and a L2 regularization term of 10^{-4}. The last layer (the classification layer) has 2 nodes and uses a softmax activation function. We use a sparse-categorical-crossentropy loss for training.