Learning a Latent Space for EEGs with Computational Graphs

Radhakrishnan Thiyagarajan, Masters Candidate
Dr. Sam Keene, Thesis Advisor

Department of Electrical & Computer Engineering
The Cooper Union for the Advancement of Science & Art

April 4, 2018

Overview

- Diverse biological signals are stored in unstructured forms
  - Examples: EEGs, EKGs, MEGs, X-Rays, MRIs, etc.
- Difficult to perform cohort retrieval or comparisons
- Cohort retrieval - task of efficiently finding a group of observations that share defining characteristics
- Picone et al. 2015 use HMMs to detect and classify signals given time-domain EEG data
  - Can you infer similarity from this?
- Clustering similar signals can reveal deeper information and knowledge about signals

Solution

Optimize a Deep Neural Network so that it can translate directly from signal to embedding such that clusters of similar signals form in the space

Machine Learning

- Supervised: Mapping from a set of input-output pairs
- Unsupervised: Underlying structure from a set of inputs
  - Examples: dimensionality reduction, cluster analysis
- Model as a function

$$\mathbf{y} = f(\mathbf{X})$$

Machine Learning

- Supervised: Mapping from a set of input-output pairs
- Unsupervised: Underlying structure from a set of inputs
  - Examples: dimensionality reduction, cluster analysis
- Model as a function

$$\mathbf{y} = f(\mathbf{X} \mid \boldsymbol{\theta})$$

Machine Learning

- Supervised: Mapping from a set of input-output pairs
- Unsupervised: Underlying structure from a set of inputs
  - Examples: dimensionality reduction, cluster analysis
- Model as a function

$$\mathbf{y} = f(\mathbf{X} \mid \boldsymbol{\theta})$$

- Training

$$J(\mathbf{y}, f(\mathbf{X} \mid \boldsymbol{\theta}))$$

Machine Learning

- Supervised: Mapping from a set of input-output pairs
- Unsupervised: Underlying structure from a set of inputs
  - Examples: dimensionality reduction, cluster analysis
- Model as a function

$$\mathbf{y} = f(\mathbf{X} \mid \boldsymbol{\theta})$$

- Training

$$J_T(\mathbf{y}, f(\mathbf{X} \mid \boldsymbol{\theta})) = J(\mathbf{y}, f(\mathbf{X} \mid \boldsymbol{\theta})) + \lambda \; P(\boldsymbol{\theta})$$

Machine Learning

- Supervised: Mapping from a set of input-output pairs
- Unsupervised: Underlying structure from a set of inputs
  - Examples: dimensionality reduction, cluster analysis
- Model as a function

$$\mathbf{y} = f(\mathbf{X} \mid \boldsymbol{\theta})$$

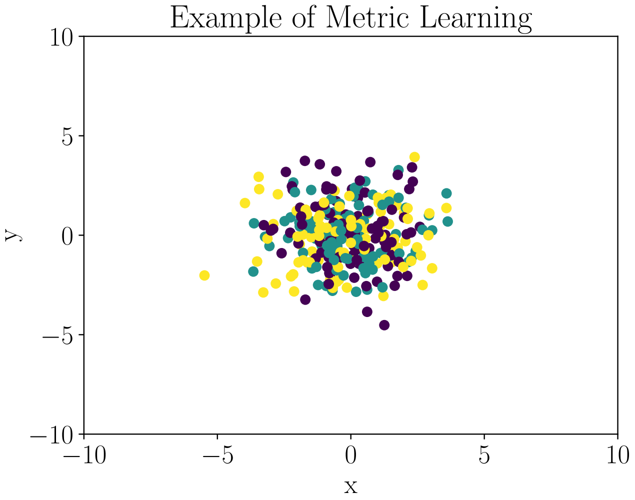
- Training

$$J_T(\mathbf{y}, f(\mathbf{X} \mid \boldsymbol{\theta})) = J(\mathbf{y}, f(\mathbf{X} \mid \boldsymbol{\theta})) + \lambda\ P(\boldsymbol{\theta})$$

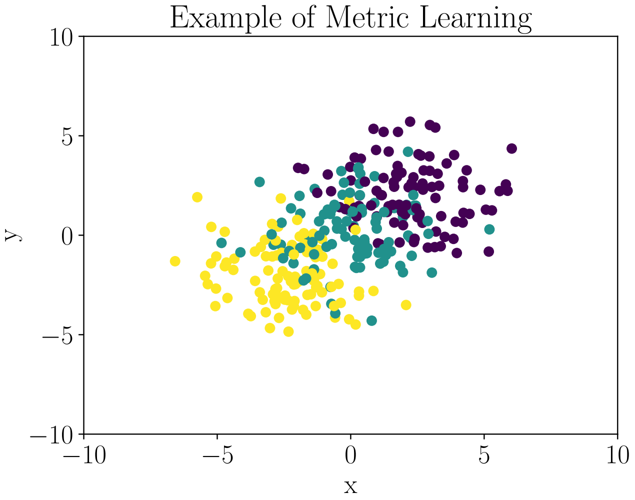
- Gradient Descent

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\ \nabla_{\boldsymbol{\theta}} J_T(\mathbf{y}, f(\mathbf{X} \mid \boldsymbol{\theta}))$$

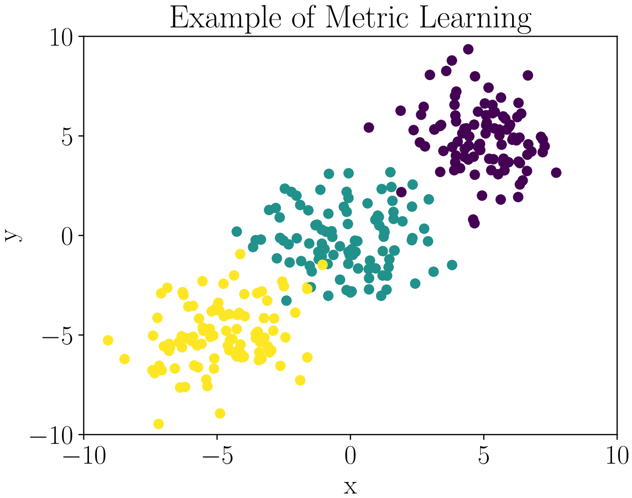
Metric Learning



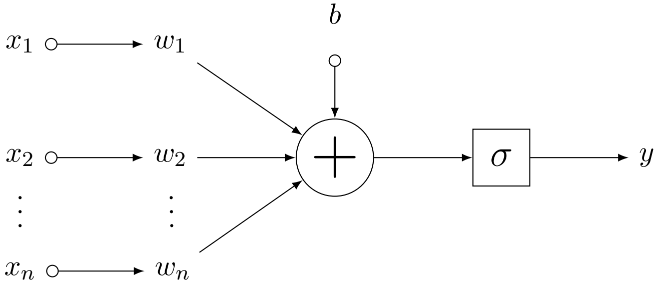
Metric Learning



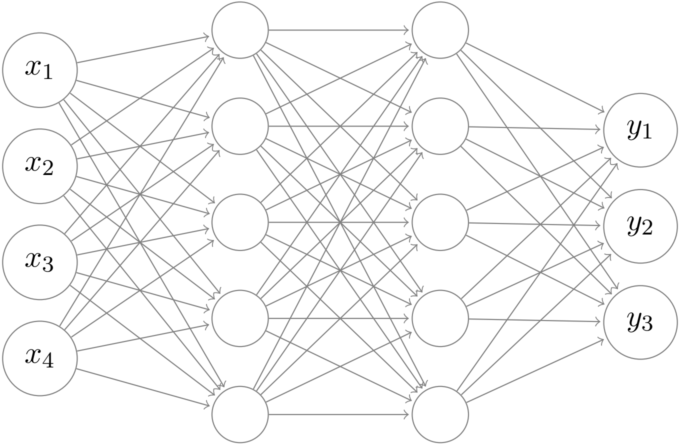
Metric Learning



Computational Graphs & Neural Networks



Fully Connected Layers



Convolutional Layers

0	0 ×0	0 ×0	0 ×1	0	0	0
0	0 ×0	21 ×1	0 ×0	0	0	0
0	85 ×1	71 ×0	0 ×0	0	0	0
0	250	231	127	63	3	0
0	250	252	250	209	56	0
0	250	252	250	250	83	0
0	0	0	0	0	0	0

Image

$*$

0	0	1
0	1	0
1	0	0

Kernel

$=$

0	106	71	0	0
106	321	231	127	63
321	481	379	313	212
481	629	565	462	306
502	502	459	306	83

Feature Map

Contrastive Loss

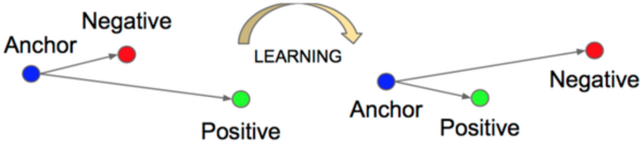
- Hadsell et al. 2006 minimize the distance between a pair of examples with same class label and penalizes the the negative pair distance
- Illustration of contrastive learning:

- Mathematically,

$$J = \frac{1}{m} \sum_{(i,j)}^{m/2} y_{i,j} D_{i,j}^2 + (1 - y_{i,j})[\alpha - D_{i,j}]_+^2$$

Triplet Loss

- Schroff et al. 2015 minimize distance between similar inputs and maximize distances between dissimilar inputs
- How do we know whether a signal is similar? With labels!
- Anchor, an instance of class a; positive, an instance of class a; negative, an instance of class b



- Mathematically,

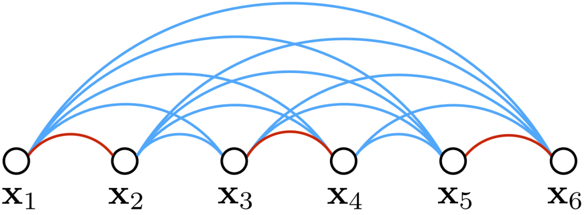
$$J = \sum_{(a,p,n)}^{N} D_{a,p}^2 - D_{a,n}^2 + \alpha$$

Lifted Structure Embedding

- Song et al. 2015 attempt to learn an embedding by looking at all possible pairs of related pairs in a minibatch
- Worked very well but more complicated than Triplet loss

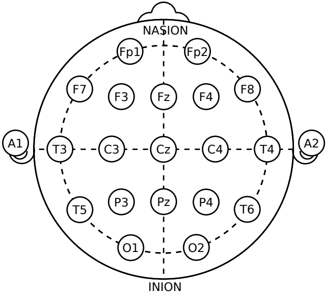
$$\tilde{J}_{i,j} = log\left(\sum_{(i,k)\in\mathcal{N}} \exp\{\alpha - D_{i,k}\} + \sum_{(j,l)\in\mathcal{N}} \exp\{\alpha - D_{j,l}\} \right) + D_{i,j}$$

$$J = \frac{1}{2|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max\left(0, \tilde{J}_{i,j}\right)^2$$

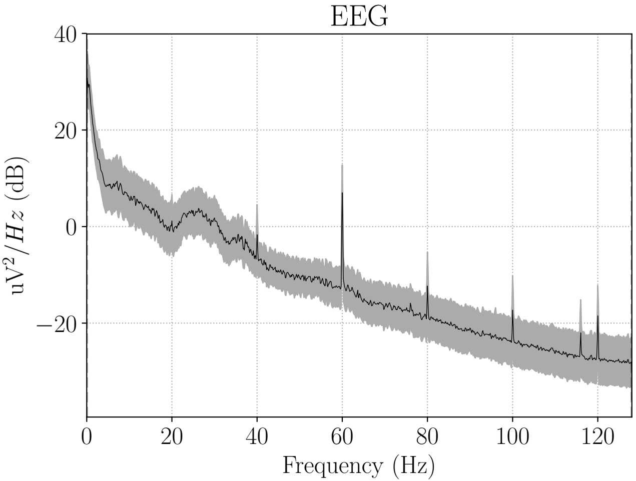


Electroencephalography (EEG)

- Method of measuring electrical activity in the brain
- Helps diagnose variety of diseases
- Has standard, 10-20 placement
- Montages, differences in voltages, are used in medicine
- Frequency, phase, amplitude, location all are import sources of information an an EEG



PSD of Sample from TUH EEG Corpus

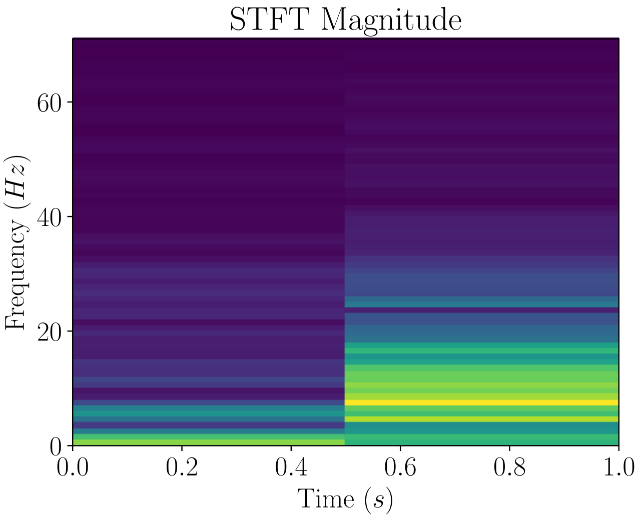


Data Rectification

- Notch filtered at 60 Hz and 120 Hz
- Bandpass filtered with 1 - 70 Hz passband
- STFT with window of 140 samples & stride of 2 samples
- Resulted in a 71×125 matrix for each second of the EEG
- Omitted locations due to classification inconsistencies
- Split into mutually exclusive training and validation set of 85% and 15% respectively

Code	Description	
BCKG	Background noise	Noise-Like
ARTF	Artifacts	Noise-Like
EYBL	Eyeball movement	Noise-Like
SPSW	Spikes & sharp waves	Seizure-Like
PLED	Periodic lateralized epileptiform discharges	Seizure-Like
GPED	Generalized periodic epileptiform discharges	Seizure-Like

Resulting Matrix



Experimental Design Choices

- Is deep learning appropriate for this problem?
  - Dealing with unstructured data
- Which technique can we use to train the network?
  - Triplet Loss because it's simple yet effective
  - Relatively easy to mine for triplets
- What type of network do we choose?
  - Convolutional Neural Network
  - CNNs tend to do well on images
- How do we test the results?
  - Classification using k-Nearest Neighbors
  - Visualize latent space using t-SNE in 2D
  - Compare to baseline classifier

Intial Experiment

- Designed simple CNN and implemented in TensorFlow
- Initially converged to zero due to small values and stalling triplet selection
- Amplified inputs to prevent both mistakes and speed up learning

Layer	Input	Kernel
conv1	$71 \times 125 \times 1$	4×4
pool1	$71 \times 125 \times 32$	3×3
conv2	$35 \times 62 \times 32$	5×5
pool2	$35 \times 62 \times 64$	2×2
fc1	$17 \times 30 \times 64$	N/A
fc2	256	N/A
output	128	N/A

Intial Experiment

Hyperparameter Optimization

- Optimized hyperparameters based on manual gridsearch
  - $\eta = 10^{-3}, \lambda_{L_2} = 10^{-4}$
  - $d = 128, \alpha = 1.0$
- Trained for 60k steps

Measuring Performance

- Used k-NN with $k = 5$ to classify signals and calculate accuracies
- Resulted in 80% accuracy

Error Organizing Data

- Different classes were split, but sessions were not

Layer	Input	Kernel
conv1	$71 \times 125 \times 1$	4×4
pool1	$71 \times 125 \times 32$	3×3
conv2	$35 \times 62 \times 32$	5×5
pool2	$35 \times 62 \times 64$	2×2
fc1	$17 \times 30 \times 64$	N/A
fc2	256	N/A
output	128	N/A

Deeper Convolutional Network

- Designed network with 14 layers
- Convolutions followed by maxpool layers and fully connected layers
- Results in a 64 dimension vector representing the signal in embedding space
- Utilized same triplet loss to optimize network

Layer	Input	Kernel
conv1	$71 \times 125 \times 1$	5×5
maxpool1	$71 \times 125 \times 32$	5×5
conv2	$34 \times 61 \times 32$	3×3
maxpool2	$34 \times 61 \times 64$	3×3
conv3	$16 \times 30 \times 64$	2×2
maxpool3	$16 \times 30 \times 128$	2×2
conv4	$8 \times 15 \times 128$	1×1
maxpool4	$8 \times 15 \times 256$	2×2
conv5	$4 \times 7 \times 256$	4×4
maxpool5	$4 \times 7 \times 1024$	4×4
flatten	$1 \times 2 \times 1024$	N/A
fc1	2048	N/A
fc2	1024	N/A
fc3	512	N/A
fc4	256	N/A
output	64	

Intial Experiment

Hyperparameter Selection

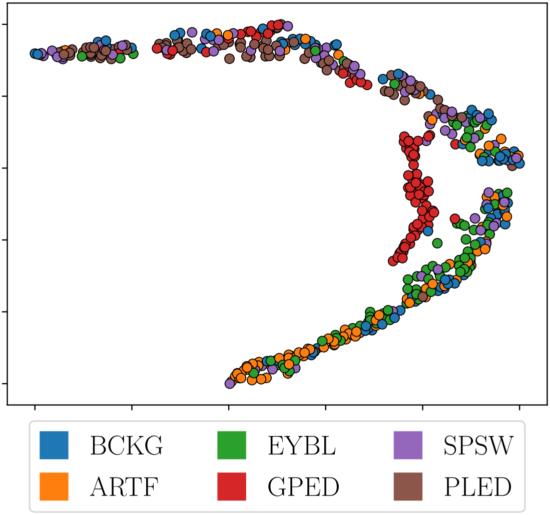
- Optimized hyperparameters based on manual gridsearch
  - $\eta = 10^{-5}, \lambda_{L_2} = 10^{-3}$
  - $d = 64, \alpha = 0.5$
- Trained for 105k steps

Measuring Performance

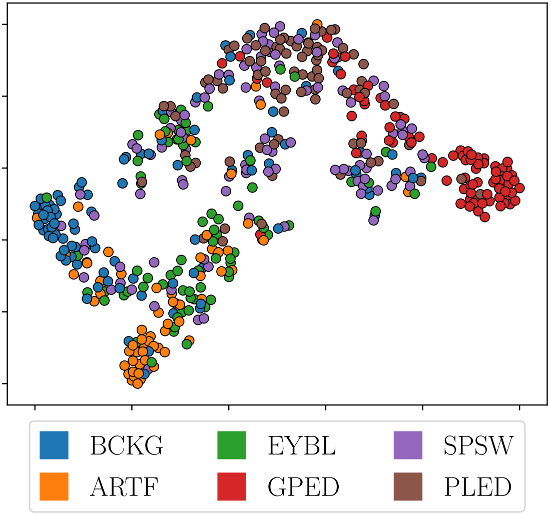
- Same procedure with k=31
- Resulted in 60.4% 6-class and 90.4% 2-class accuracy
- Constructed t-SNE reduced plots

Layer	Input	Kernel
conv1	$71 \times 125 \times 1$	5×5
maxpool1	$71 \times 125 \times 32$	5×5
conv2	$34 \times 61 \times 32$	3×3
maxpool2	$34 \times 61 \times 64$	3×3
conv3	$16 \times 30 \times 64$	2×2
maxpool3	$16 \times 30 \times 128$	2×2
conv4	$8 \times 15 \times 128$	1×1
maxpool4	$8 \times 15 \times 256$	2×2
conv5	$4 \times 7 \times 256$	4×4
maxpool5	$4 \times 7 \times 1024$	4×4
flatten	$1 \times 2 \times 1024$	N/A
fc1	2048	N/A
fc2	1024	N/A
fc3	512	N/A
fc4	256	N/A
output	64	

t-SNE Plot at 5k Iterations



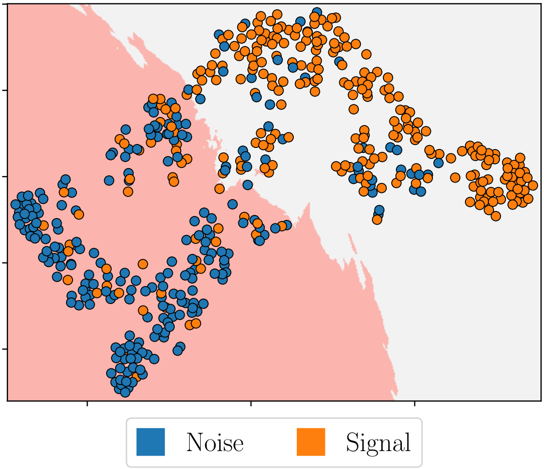
t-SNE Plot at 105k Iterations



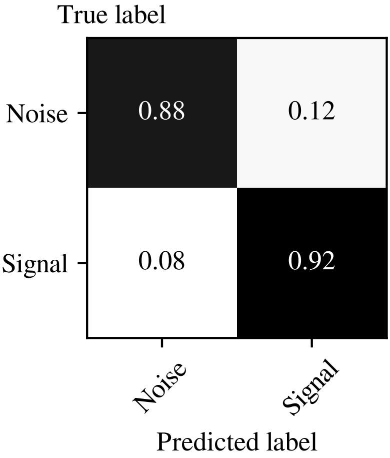
Confusion Matrix for Six-Class Classification

True label \ Predicted label	BCKG	ARTF	EYBL	GPED	SPSW	PLED
BCKG	0.61	0.12	0.09	0.01	0.08	0.08
ARTF	0.16	0.53	0.26	0.01	0.04	0.00
EYBL	0.11	0.22	0.54	0.01	0.11	0.01
GPED	0.00	0.00	0.00	0.94	0.06	0.00
SPSW	0.04	0.03	0.11	0.09	0.32	0.40
PLED	0.01	0.00	0.01	0.18	0.10	0.69

t-SNE Plot with Binary Decision Boundary

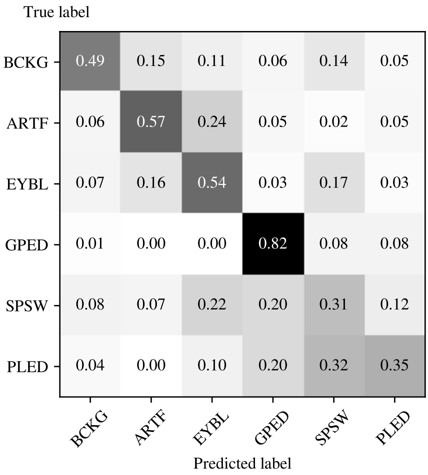


Confusion Matrix for Binary Classification



DCNN with Softmax Loss

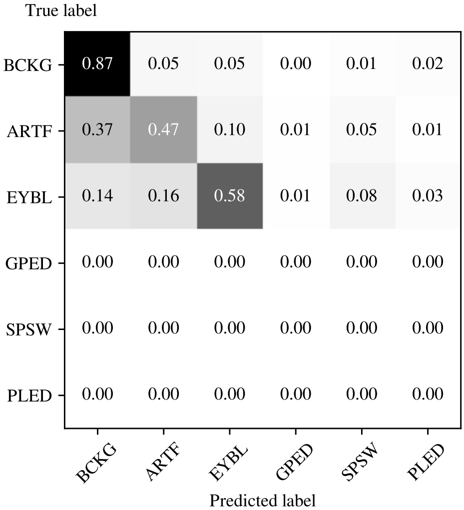
- Created a generic classifier
- Utilized same network
- Applied cross-entropy loss
- Resulted in classification accuracy of 50.2%
- Surprising results

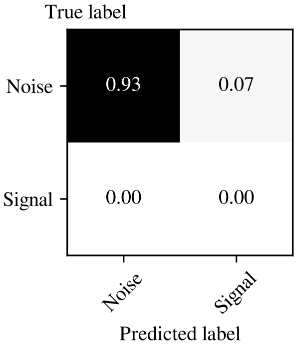


Error Analysis

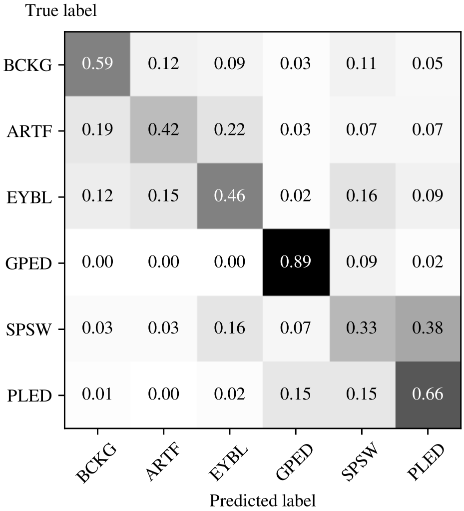
- Analyze where most error occurs
- Split dataset into three sectors:
  - Type A: Sessions without seizure-like signals
  - Type B: Sessions with seizure-like signals
  - Type C: Sessions with seizure-like signals considering only seizure-like signals
- Try to identify reasons why these errors occur
- Splitting them helps identify how the changes in sessions changes results

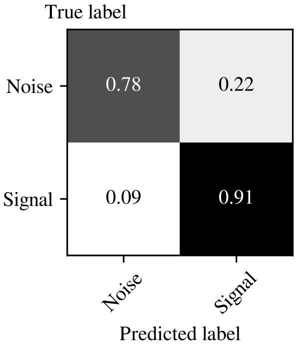
Error Analysis: Type A (64.6% and 93.0%)



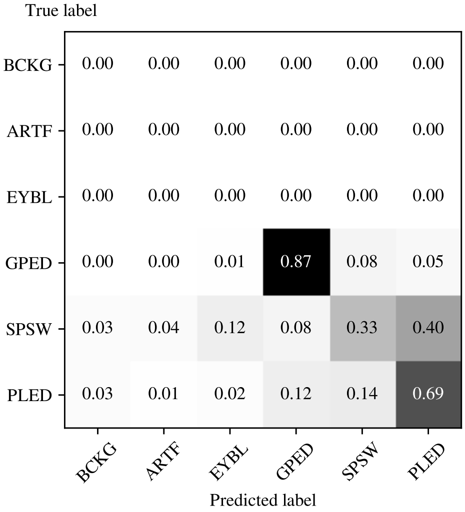


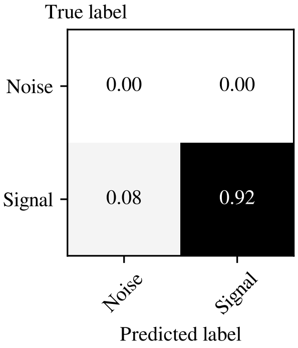
Error Analysis: Type B (56.0% and 85.0%)





Error Analysis: Type C (63.0% and 91.8%)





Possible Sources of Error

- Misclassified signals
- Very similar signals
- Loss of information due to:
  - Notch filter
  - Bandpass filter
  - Magnitude of STFT
  - Location on scalp
- **However,** accuracy is still high for a signal with low SNR

Conclusions and Future Work

- Demonstrated an end-to-end system to learn latent spaces for EEG signals
- 60.4% six-class & 90.4% binary classification accuracies
- Does better than generic DCNN classifier and provides more information on similarity
- Experiment swapping triplet loss with loss functions from Structured Feature Embeddings written Song et al. 2015 respectively
- Do an in-depth analysis between features produced by baseline and those produced by experimental network
- Attempt to incorporate physicians notes in order to enrich embeddings produced using adaptive density discrimination
- Extend this method to other types of medical signals and enrich understanding of different pathologies

Thank you to...

- Professor Sam Keene
- Chris Curro
- ECE Faculty
- My friends
- My parents