Methods for Improved EEG Classification with Neural Networks

Xiang Wang, Cherelle Connor, Tai Nguyen, Liangyu Tao

Background

- Neural prosthetics allow people with motor disability to have a chance at a normal life.
- EEG shows great potential on decoding movements due to being noninvasive, cheap and fast.
- Previous research has shown that Neural Networks can achieve high accuracy in classifying actions involving highly segregated limb extremities
- Can this high accuracy generalize to more difficult datasets?

Related Work

Paper 1. Upper limb movements can be decoded from the time-domain of low-frequency EEG (Ofner, 2017)

Aim: To determine if executed and imagined movements from the same limb can be extracted from low-frequency time-domain signals (<3 Hz).

Method: Used linear discriminative analysis and sLORETA to identify between 6 movement classes and 1 rest class

Results: Obtained significant accuracies of 55% (movement vs movement) and 87% (movement vs rest) for executed movements, and 27% and 73%, respectively, for imagined movements

Related Work

Paper 2. Cascade and Parallel Convolutional Recurrent Neural Networks on EEG-Based Intention Recognition for Brain Computer Interface (Zhang et al., 2018)

Aim: To detect human movement intentions through learning the effective compositional spatio-temporal dynamics from raw EEG streaming signals without preprocessing.

Method: Converting EEG sequences to 2 D Meshes and build a cascade and a parallel LSTM-CNN

Results: Both models achieve and accuracy near 98.3%; real world evaluation of with BCI resulted in a recognition accuracy of 93%

Aim

1. Investigate whether the network architecture can be used to decode a more difficult dataset (similar set of actions)
2. Gain insight into the electrophysiological features associated with movement execution that allows neural networks to achieve high accuracy.

Outline of proposal

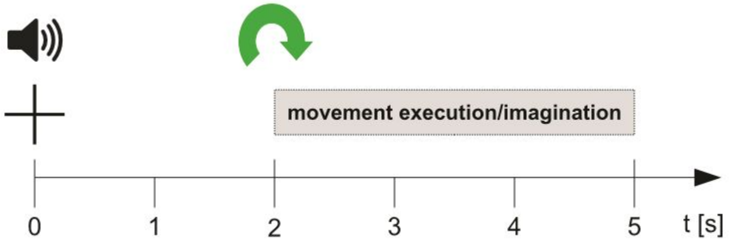
- Dataset
- Data Preprocessing
- Network Architecture
- Results
    - Pairwise Classification
    - Model Interpretation
- Conclusions
- Limitations/Future Work

Data

15 participants

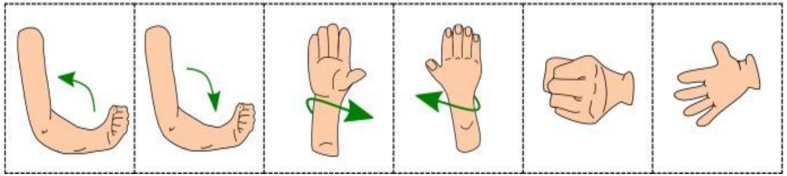
2 sessions

EEG data recorded on 61 channels from corresponding motor execution or motor imagination



a

b

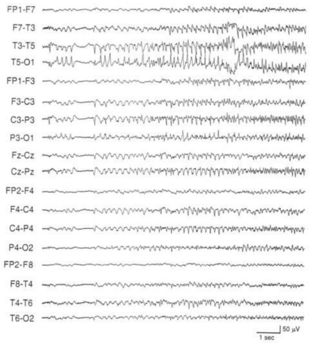


Ofner et. al (2017)

Data Processing

Raw eeg data → Signal Processing → Artifact Detection → Movement Onset

Raw eeg data



Signal Processing

1. Downsample from 512 → 128 Hz
2. Linear baseline subtraction

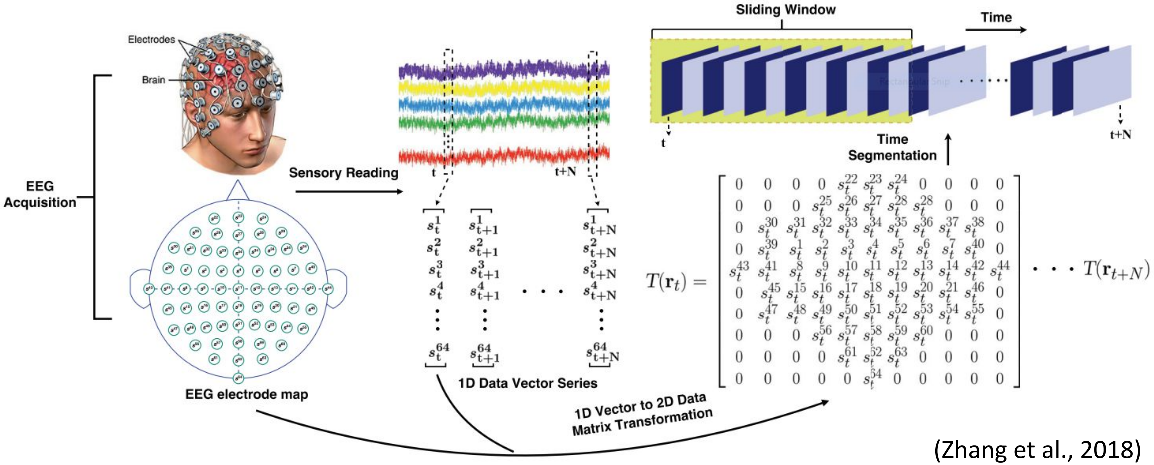
Artifact Detection

1. Detect artifact based on EOG channels
2. Kurtosis > 5 std
3. Joint Probability > 5 std
4. Absolute value > 200uV

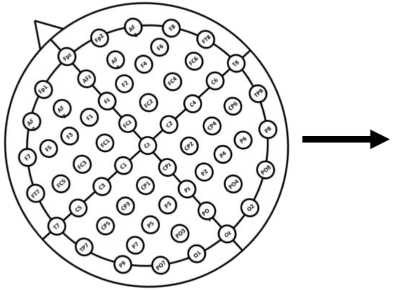
Movement Onset

1. PCA on 3D elbow, 3D wrist, and 14D hand data
2. Inflection point detection
3. Align and window to +/- 1s from movement onset

Preprocessing for Neural Network



(Zhang et al., 2018)

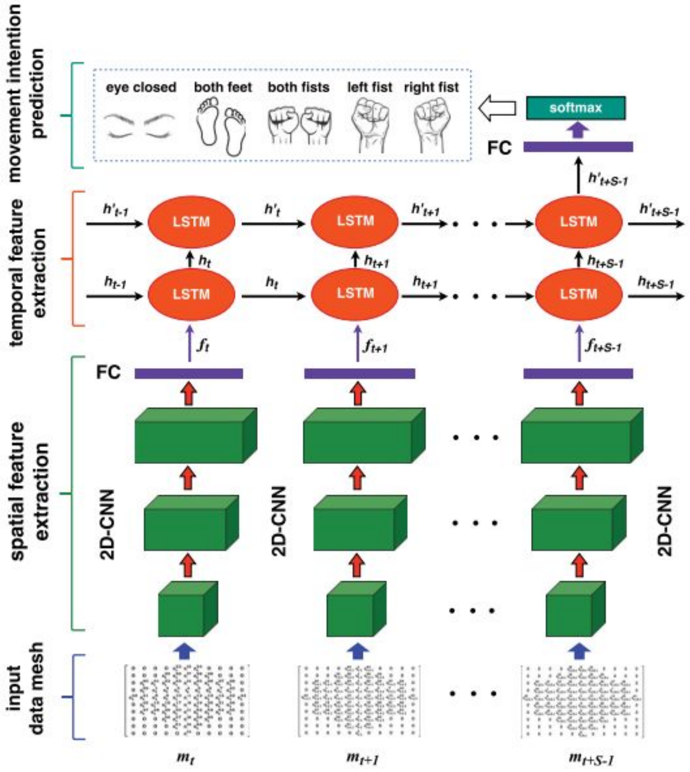


0	0	0	0	F4	FFC6h	FC6	FTT8h	0
0	0	0	F2	FFC4h	FC4	FCC6h	C6	TTP8h
0	0	Fz	FFC2h	FC2	FCC4h	C4	CCP6h	CP6
0	F1	FFC1h	FCz	FCC2h	C2	CCP4h	CP4	CPP6h
F3	FFC3h	FC1	FCC1h	Cz	CCP2h	CP2	CPP4h	P4
FFC5h	FC3	FCC3h	C1	CCP1h	CPz	CPP2h	P2	0
FC5	FCC5h	C3	CCP3h	CP1	CPP1h	Pz	PPO2h	0
FTT7h	C5	CCP5h	CP3	CPP3h	P1	PPO1h	0	0
0	TTP7h	CP5	CPP5h	P3	0	0	0	0

Following Zhang et al., we preprocessed by:

1. Convert 1D channel information to 2D data matrix
2. Bin matrices into 62.5 ms bins with 31.25 ms overlap

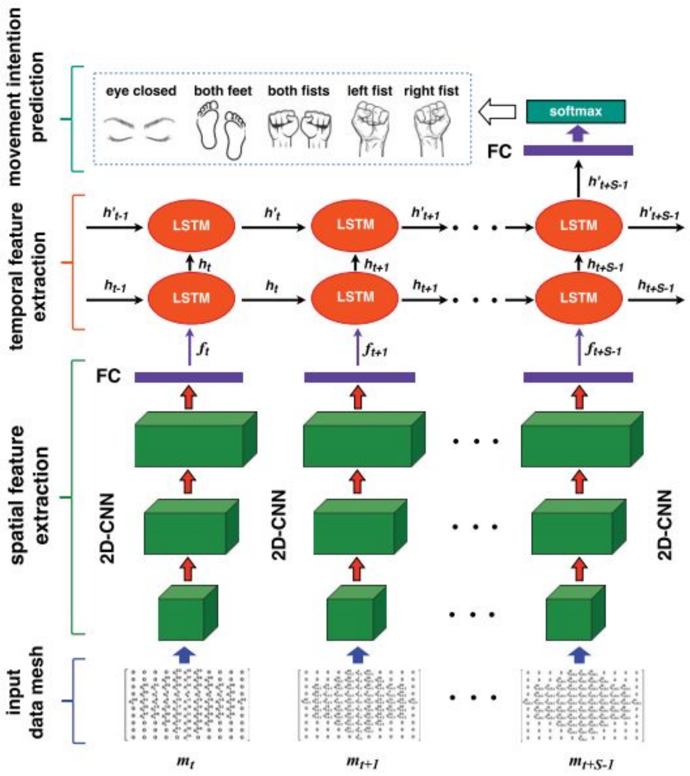
Network Architecture



(Zhang et al., 2018)

- Using keras's TimeDistributed class in order to create S (number of frames in a segment) parallel CNN stacks (8) automatically with input dimension of 8 x 9 x 9 x 1
- 1st CNN layer has kernel size = 3x3 and 32 filters
- 2nd CNN layer has kernel size = 3x3 and 64 filters
- 3rd CNN layer has kernel size = 3x3 and 128 filters
- The output of the S CNN stacks is: 8 x 9 x 9 x 128, which is flatten into 8 x 10368
- Then, it is fed into a fully connected layer, condensing into 8 x 1024
- Each 1024-length vector is fed into 1 LSTM unit
- 2 LSTM layers, bottom layer is treated as external input of the upper layer
- The output of the last unit in the 2nd layer is fed into a fully connected layer of size 1 x 64
- The output of this layer is fed into another fully connected layer of size 1 x 8 and softmax activation function is applied into this last layer for classification.
- Learning rate is 1e-4
- Optimizer is Adam
- Cost function is Categorical Entropy
- Batch size is 16. Number of epoch is 20.

Network Architecture

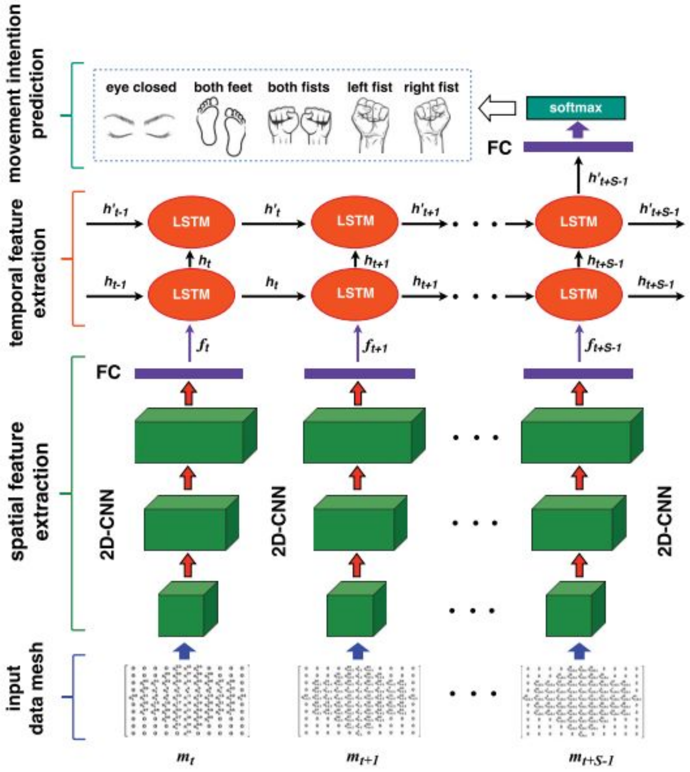


(Zhang et al., 2018)

- `model = Sequential()`
- `model.add(TimeDistributed(Conv2D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=inputs.shape[1:])))`
- `model.add(TimeDistributed(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu')))`
- `model.add(TimeDistributed(Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')))`
- `model.add(TimeDistributed(Flatten()))`
- `model.add(TimeDistributed(Dense(1024, activation='relu')))`
- `model.add(TimeDistributed(Dropout(0.5)))`
- `model.add(LSTM(S, return_sequences=True))`
- `model.add(LSTM(S))`
- `model.add(Dense(64, activation='relu'))`
- `model.add(Dropout(0.5))`
- `model.add(Dense(nClasses, activation='softmax'))`
- `opt = keras.optimizers.Adam(learning_rate=1e-4)`
- `model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])`

- `targetsOneHot = to_categorical(y_train-1)`
- `history = model.fit(X_train, targetsOneHot, epochs=50, batch_size=16, verbose=1, validation_split=0.2, callbacks=[checkpoint,early])`
- `# evaluate model`
- `targetsOneHot = to_categorical(y_test-1)`
- `_, accuracy = model.evaluate(X_test, targetsOneHot, batch_size=16, verbose=1)`
- `display(accuracy)`

Network Architecture

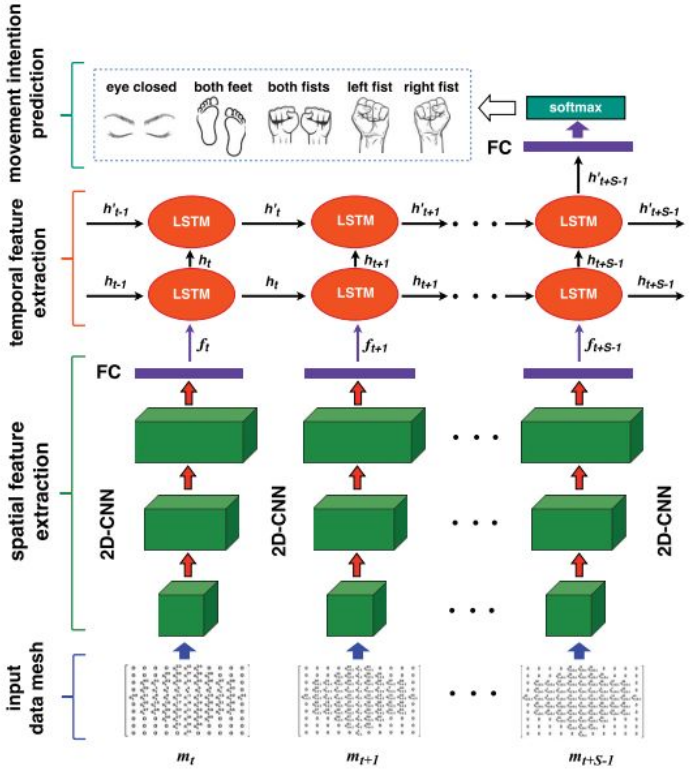


(Zhang et al., 2018)

Layer (type)	Output Shape	Param #
time_distributed_12 (TimeDis	(None, 8, 9, 9, 32)	320
time_distributed_13 (TimeDis	(None, 8, 9, 9, 64)	18496
time_distributed_14 (TimeDis	(None, 8, 9, 9, 128)	73856
time_distributed_15 (TimeDis	(None, 8, 10368)	0
time_distributed_16 (TimeDis	(None, 8, 1024)	10617856
time_distributed_17 (TimeDis	(None, 8, 1024)	0
lstm_2 (LSTM)	(None, 8, 8)	33056
lstm_3 (LSTM)	(None, 8)	544
dense_5 (Dense)	(None, 64)	576
dropout_4 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 2)	130

Total params: 10,744,834
Trainable params: 10,744,834
Non-trainable params: 0

Network Architecture



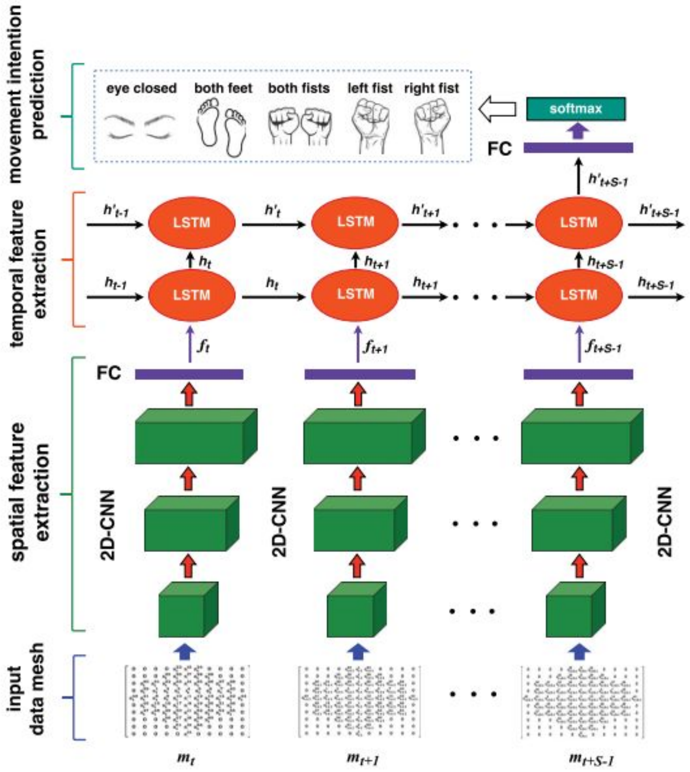
(Zhang et al., 2018)

Sample Run with class 4, 5 and 6

```
Epoch 00008: val_accuracy improved from 0.84980 to 0.85938, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 9/50
90598/90598 [==============================] - 517s 6ms/step - loss: 0.1178 - accuracy: 0.9603 - val_loss: 0.4176 -
val_accuracy: 0.8645

Epoch 00009: val_accuracy improved from 0.85938 to 0.86446, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 10/50
90598/90598 [==============================] - 519s 6ms/step - loss: 0.1040 - accuracy: 0.9653 - val_loss: 0.4128 -
val_accuracy: 0.8698

Epoch 00010: val_accuracy improved from 0.86446 to 0.86980, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 11/50
90598/90598 [==============================] - 510s 6ms/step - loss: 0.0913 - accuracy: 0.9689 - val_loss: 0.4214 -
val_accuracy: 0.8729

Epoch 00011: val_accuracy improved from 0.86980 to 0.87289, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 12/50
90598/90598 [==============================] - 504s 6ms/step - loss: 0.0864 - accuracy: 0.9714 - val_loss: 0.4060 -
val_accuracy: 0.8784

Epoch 00012: val_accuracy improved from 0.87289 to 0.87841, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 13/50
90598/90598 [==============================] - 493s 5ms/step - loss: 0.0779 - accuracy: 0.9744 - val_loss: 0.4124 -
val_accuracy: 0.8792

Epoch 00013: val_accuracy improved from 0.87841 to 0.87921, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 14/50
90598/90598 [==============================] - 491s 5ms/step - loss: 0.0730 - accuracy: 0.9757 - val_loss: 0.4167 -
val_accuracy: 0.8807

Epoch 00014: val_accuracy improved from 0.87921 to 0.88071, saving model to CascadeModel_S8_C4_5_6.h5
Epoch 15/50
84432/90598 [==========================>...] - ETA: 31s - loss: 0.0684 - accuracy: 0.9774
```

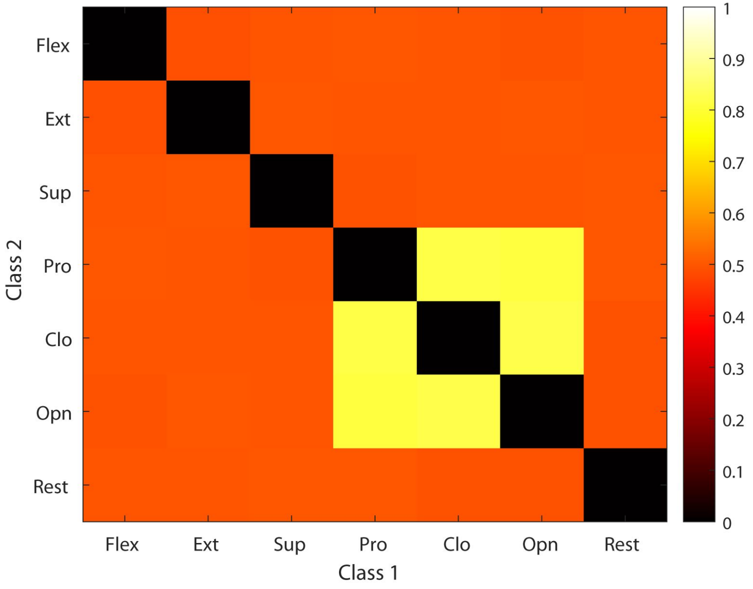
Disclaimer

- The following results were run using 1D CNN layers
- Rerunning with 2D CNN layers improves classification performance and epochs to convergence



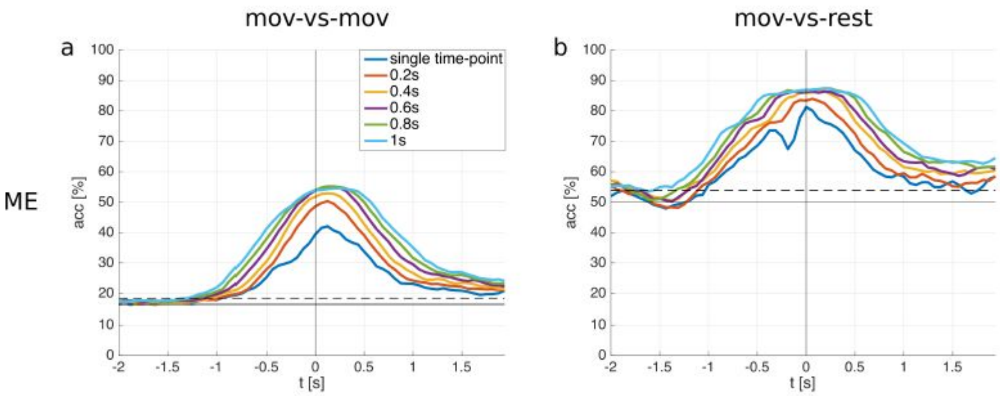
(Zhang et al., 2018)

Pairwise classification shows high accuracy for pronation, hand close, and hand open



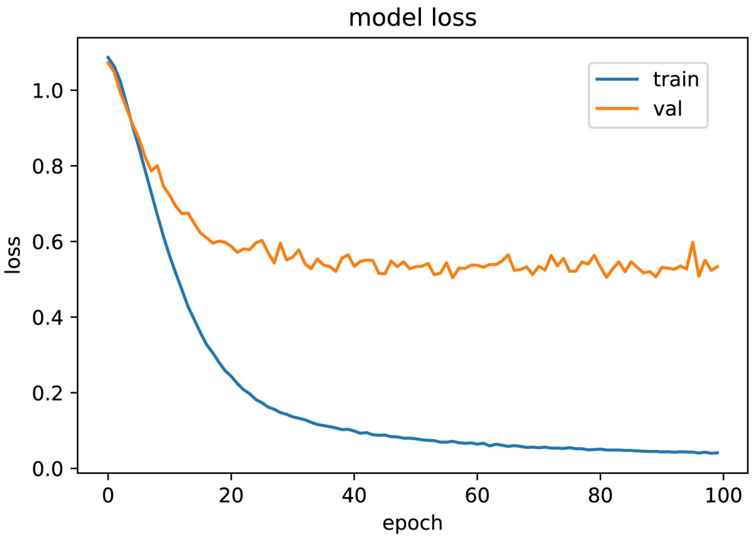
- Plot shows training accuracy
- Stopped after 10 epochs, batch size of 64
- High classification for pronation, hand open/close
- Chance level for other classes

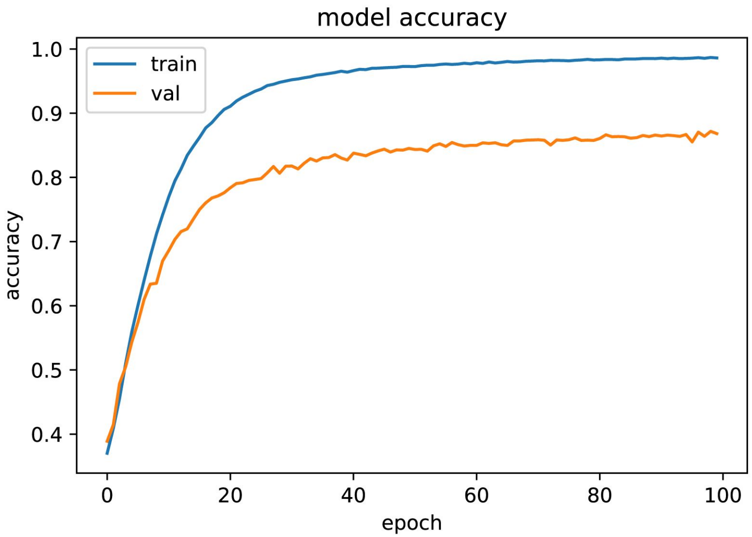
Comparison with a LDA classifier



(Zhang et al., 2018)

Classification for pronation, hand close, and hand open movements reach above 80% accuracy





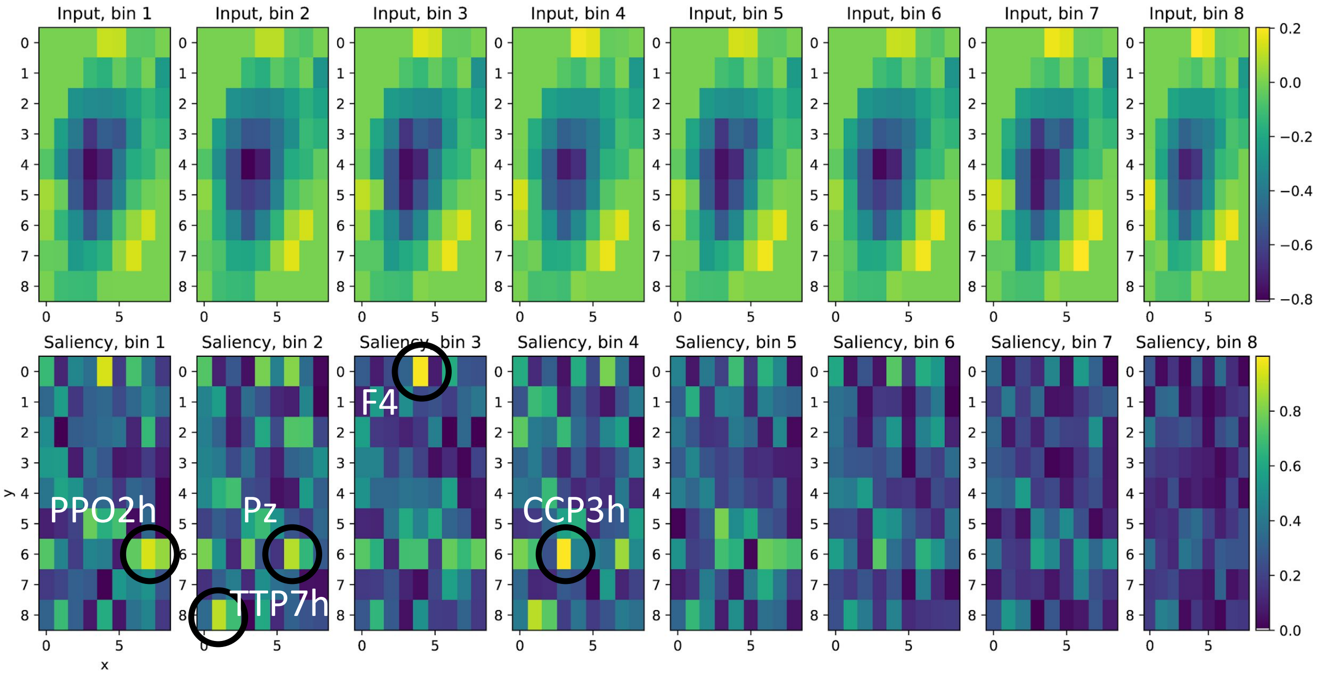
Training parameters:

- 64/16/20 (training/validation/testing)
- Reaches ~88% validation and testing accuracy by 150 iterations

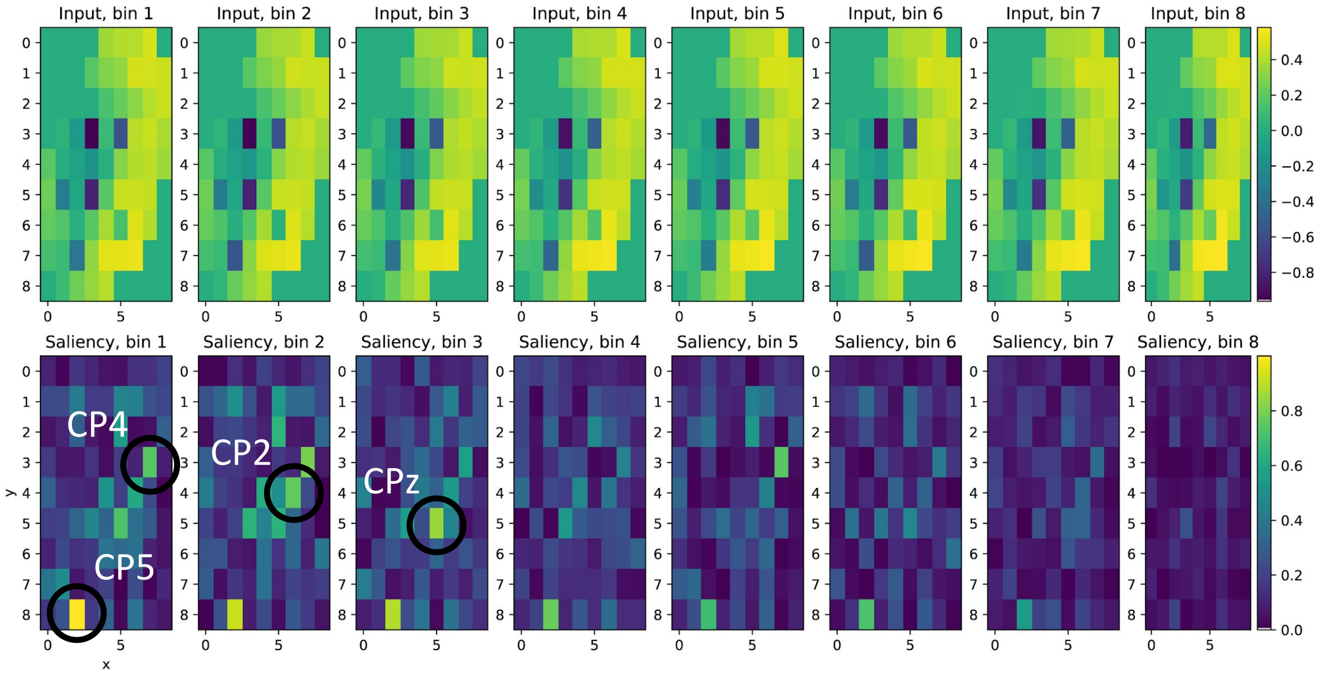
Saliency maps for “average” forearm pronation

X = Time points in average 8 frame bin

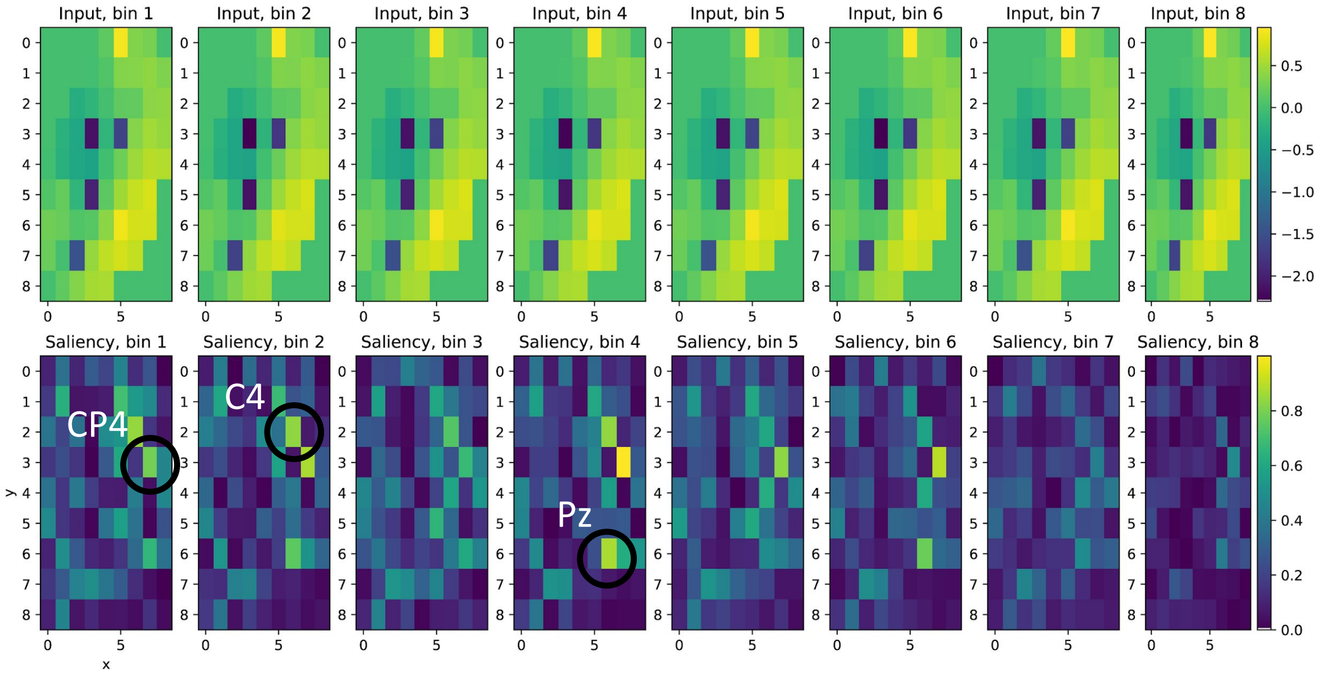
Saliency Maps $(\frac{dOutput}{dx})$



Saliency maps for “average” hand close

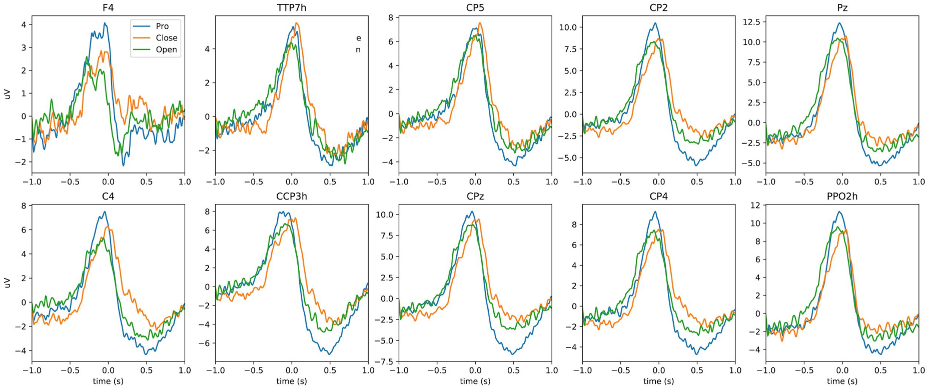


Saliency maps for “average” hand open

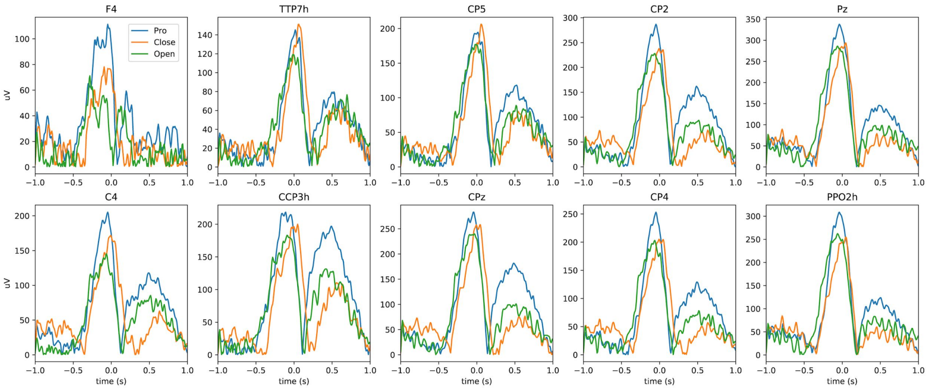


Movement Related Cortical Potential (MRCP) for most influential channels

MRCP Mean

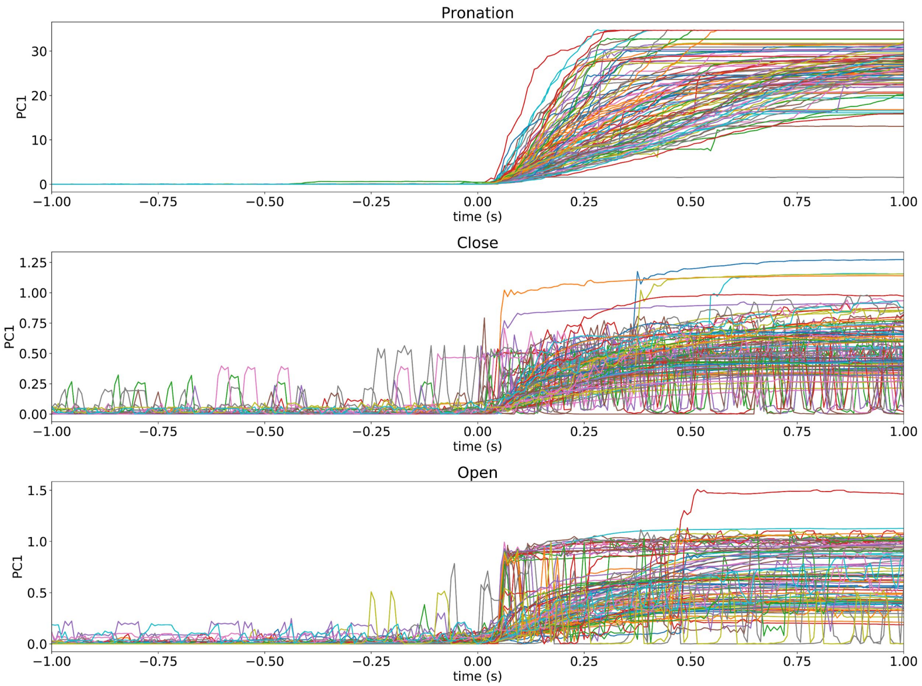


MRCP SEM



Extremely low SNR!

Movement vectors for actions show high variability in movement capabilities



Sample of 150 trials for each class aligned by movement onset

Conclusions

1. Developed a neural network model that offers a more robust method for the analysis for EEG signals
   a. Capitalize on the spatio-temporal information previously overlooked in the dataset
2. Improved classification accuracy data as compared to Ofner et. al.
   a. High classification of 3 movement classes: wrist pronation, hand open, and hand close
3. Gained some insight into significance of eeg channels on neural network

Improvements/Future directions

1. Data
    a. Classification based on the Motor Imagery dataset
2. Data Processing
    a. Systematically remove preprocessing steps to understand what is important for network performance
    b. Exploration of classification based on frequently studied frequency bands (mu, beta, delta, and low frequency)
3. Model architecture
    a. Implement a CNN-LSTM parallel network
    b. Hyperparameterization
4. Interpretation
    a. Use bins that maximize the confidence of each class (not the average bin for each class) to compute saliency maps
    b. Compute gradient maps to interpret individual layers
    c. Frequency information

References

1. Ofner, P., et al., Upper limb movements can be decoded from the time-domain of low-frequency EEG. PLOS ONE, 2017. 12(8): p. E0182578.
2. Zhang, D., et al., Cascade and Parallel Convolutional Recurrent Neural Networks on EEG-based Intention Recognition for Brain Computer Interface. 2017.