# Decoding EEG Waves for Visual Attention to Faces and Scenes

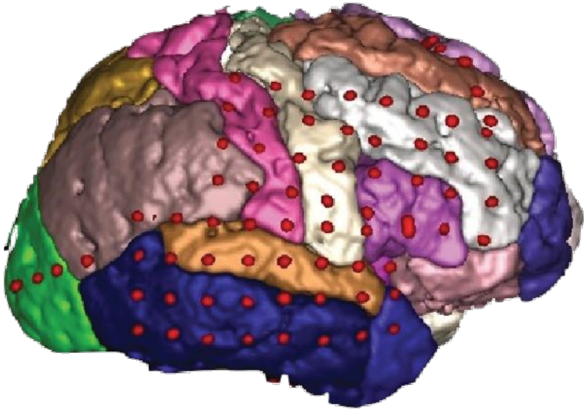
Taylor Berger and Chen Yi Yao

Mentors: Xiaopeng Zhao, Soheil Borhani

# Brain Computer Interface

- **Applications:**
  - Medical Devices (e.g. Prosthetics, Wheelchairs)
  - Educational and Self-regulation
  - Games and Entertainment
  - Security and Authentication
- **Invasive vs. Non-Invasive:**
  - Intracranial ElectroEncephaloGraphy (iEEG)
  - ElectroEncephaloGraphy (EEG)

iEEG Brain Map



Img Ref: www.ieeg.org

# Research Objective and Setup

- **Objective:** Develop a neural network model to improve the test accuracy of the system based on extracted information from EEG signals
- **Setup:**
  - Emotiv EPOC for recording EEG signals
  - MATLAB/Simulink for data collection
  - MATLAB/Python for data analysis

Emotiv Headset

Img Ref: https://www.emotiv.com/product/emotiv-epoc-14-channel-mobile-eeg/

# Data Collection

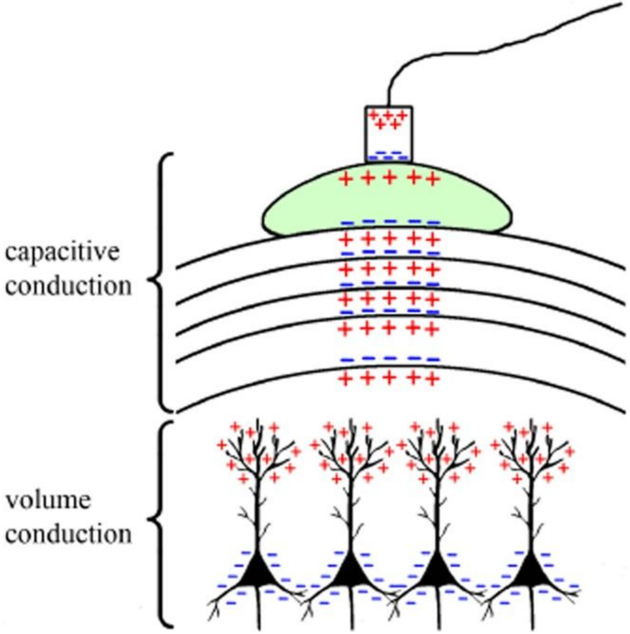
- 2 Subjects
- 2 Phases
  - Phase 1: Distinction between images
  - Phase 2: Distinction between superimposed images
- 8 Blocks each Phase (50 image trials / block)
- 14 Channels
- Time-Series Data Table


| Block Number | Task-Relevant Image | Task-Irrelevant Image |
|---|---|---|
| 1 | Indoor | Outdoor |
| 2 | Male | Female |
| 3 | Indoor | Outdoor |
| 4 | Female | Male |
| 5 | Outdoor | Indoor |
| 6 | Male | Female |
| 7 | Outdoor | Indoor |
| 8 | Female | Male |

# EEG Waves

- Signals are produced by synchronized synaptic activity in the cortical neurons
- Measurable charge is created by the summation of multiple neuron dipoles
- Volume conduction allows for the propagation of EEG signals within the brain
- A capacitor is created to allow for the propagation between volumes
- Electrodes measure voltage fluctuations over time

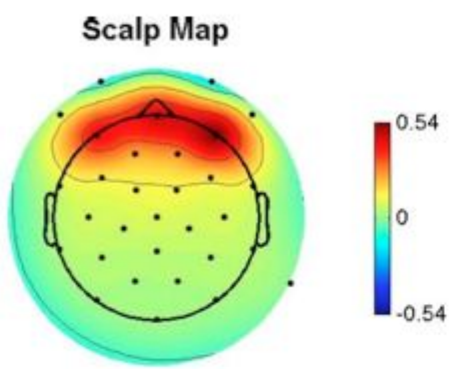
Signal Transmission from Neuron to Electrode

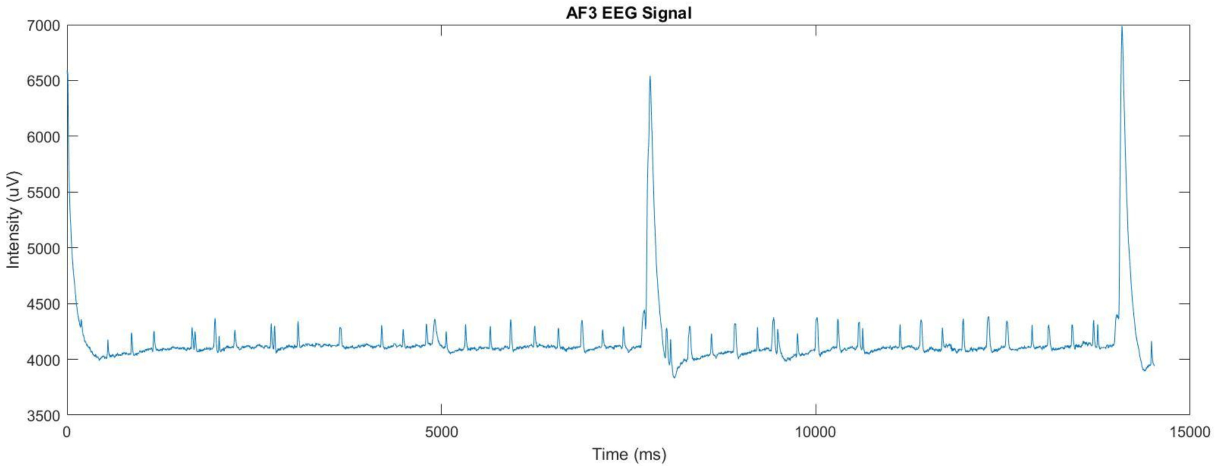



Img Ref: https://onlinelibrary.wiley.com/doi/full/10.1111/psyp.12283

# Noise

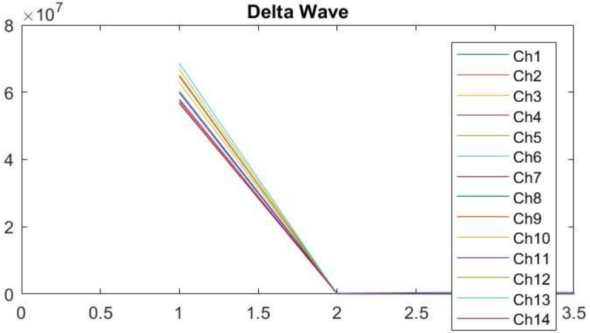
- Inherent electrical properties and physical arrangement of different tissues
- Dipole Size Variance
- Muscle Twitches
- Eye Blinks



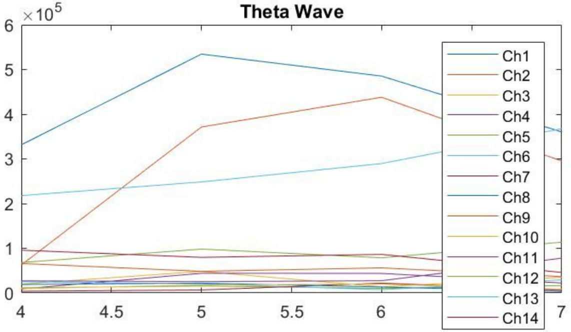

Img Ref: https://github.com/mne-tools/mne-python/issues/3757

# Frequencies within the Brain



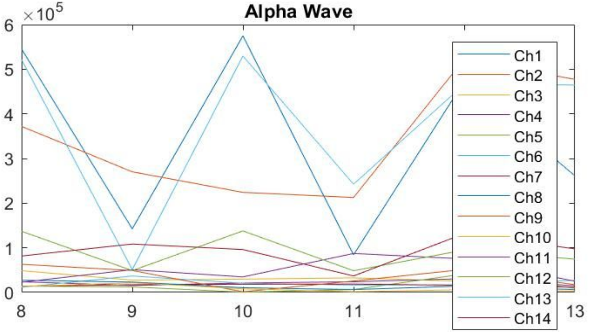

**Delta Waves**

- Subconscious
- Relaxation
- Sleep



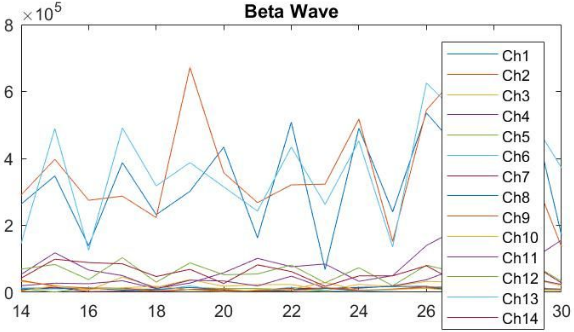

**Theta Waves**

- Subconscious
- Emotion




**Alpha Waves**

Bridge between theta and beta waves




**Beta Wave**

- Conscious States
- Reasoning
- Calculations
- Problem Solving

# Pre-Processing Techniques

- **Butterworth Filter:** Signal processing filter that produces a frequency response that is maximally flat in the passband
  - Good all-around performance
  - High rate of attenuation

General Equation for Frequency Response

Transfer Function

$$\left|\frac{V_{out}}{V_{in}}\right|^2 = \frac{1}{1 + (f / f_c)^{2n}}$$

$f$ = frequency at calculation
$f_c$ = cut off frequency
$n$ = number of elements
$V_{in}$ = input voltage
$V_{out}$ = output voltage

Filter Ref:
http://www.etc.tuiasi.ro/cin/Downloads/Filters/Filters.htm

Img Ref:
http://www.radio-electronics.com/info/rf-technology-design/rf-filters/butterworth-rf-filter-calculations-formulae-equations.php

# Pre-Processing Techniques

- **Zero Phase Filter:** Performs forward-backward filtering on the signal
  - Zero Phase Distortion

Inputs: $x[n]$= input sequence; $h[n]$= impulse response

First Pass: $X(e^{jw})H(e^{jw})$ using FT of $x[n]$ and h[n[

Time Reversal: $X(e^{-jw})H(e^{-jw})$

Second Pass: $X(e^{-jw})H(e^{jw}) H(e^{-jw})$

Output Signal: $Y(e^{jw}) = X(e^{jw})|H(e^{jw})|^2$

Filter Ref:
https://dsp.stackexchange.com/questions/9467/what-is-the-advantage-of-matlabs-filtfilt/9468

# Low-Pass Filter

- Passes Signals with a frequency lower than a certain cutoff frequency and attenuates signals with a frequency higher than the cutoff frequency

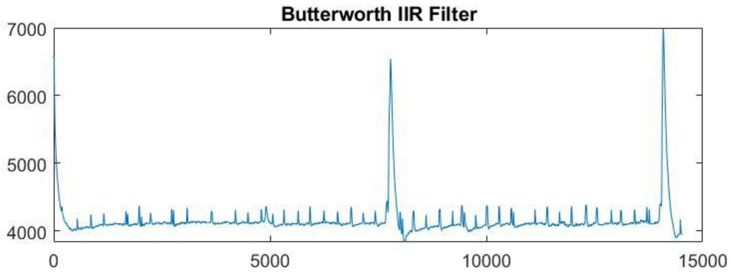
<u>Inputs:</u>

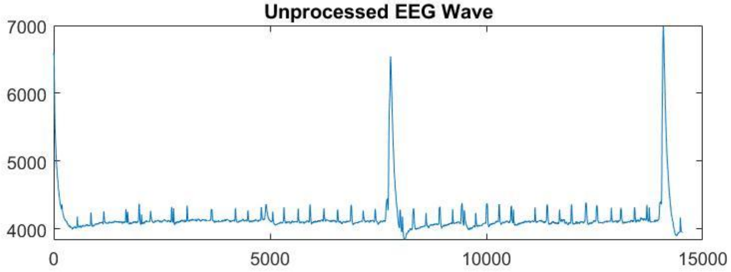
Cut Off Frequency: 40

<u>Outputs:</u>

14x14524 data table for each block

- Not enough distinction

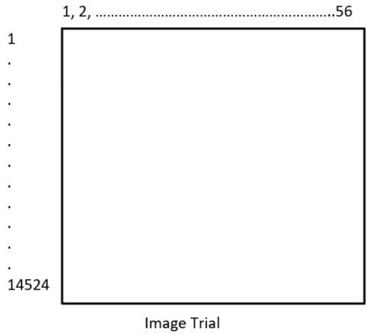


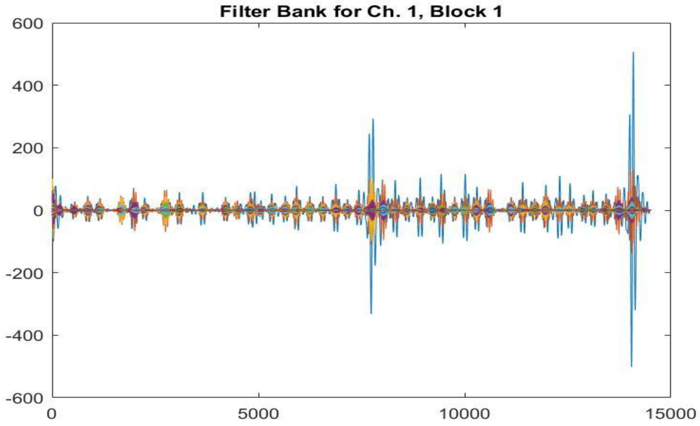



# Band-Pass Filter

- Passes signals within a specific range of frequencies through the filter and attenuates the remaining frequencies on either side of the cutoff range
- Broken into 56 frequency ranges between 1 and 57 Hz for each Image Trial
- 400 Total Image Trials (i.e. 50 Image Trials  for each block)

Channel 1 Filter Bank



# Filter Bank Sample Code

```
% Channel 14
    x14=rawEEG(:,14);
    FilterOrder=4;
    SampleFreq=128;
    i=1;
while i<=56
        Lowcut14(i)=i;
        Highcut14(i)=i+1;
        Wl_14(i)=2*Lowcut14(i)/SampleFreq;
        Wh_14(i)=2*Highcut14(i)/SampleFreq;
        Wn_14=[Wl_14(i) Wh_14(i)];
        [num14, denum14] = butter(FilterOrder, Wn_14, 'bandpass');
        filtered_data14(:,i)=filtfilt(num14, denum14, x14);
```

14524x56 double

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | -0.3998 | -0.1014 | 0.8208 | -1.2712 | -0.3285 | 1.4084 |
| 2 | -0.5810 | -0.6677 | 0.3266 | -2.1491 | -1.3828 | 1.3101 |
| 3 | -0.7646 | -1.2280 | -0.1633 | -2.9161 | -2.3443 | 1.0709 |
| 4 | -0.9495 | -1.7724 | -0.6336 | -3.5342 | -3.1434 | 0.7103 |
| 5 | -1.1345 | -2.2913 | -1.0701 | -3.9731 | -3.7218 | 0.2615 |
| 6 | -1.3184 | -2.7752 | -1.4599 | -4.2118 | -4.0369 | -0.2317 |
| 7 | -1.5001 | -3.2152 | -1.7920 | -4.2396 | -4.0648 | -0.7192 |
| 8 | -1.6785 | -3.6034 | -2.0577 | -4.0563 | -3.8023 | -1.1495 |
| 9 | -1.8522 | -3.9324 | -2.2508 | -3.6729 | -3.2672 | -1.4756 |
| 10 | -2.0202 | -4.1961 | -2.3674 | -3.1100 | -2.4971 | -1.6597 |
| 11 | -2.1813 | -4.3893 | -2.4065 | -2.3976 | -1.5465 | -1.6778 |
| 12 | -2.3343 | -4.5081 | -2.3696 | -1.5727 | -0.4834 | -1.5227 |
| 13 | -2.4781 | -4.5500 | -2.2609 | -0.6778 | 0.6160 | -1.2050 |
| 14 | -2.6118 | -4.5137 | -2.0866 | 0.2413 | 1.6729 | -0.7530 |
| 15 | -2.7342 | -4.3994 | -1.8551 | 1.1380 | 2.6112 | -0.2096 |
| 16 | -2.8443 | -4.2088 | -1.5765 | 1.9677 | 3.3632 | 0.3710 |
| 17 | -2.9414 | -3.9448 | -1.2621 | 2.6891 | 3.8747 | 0.9295 |

# Feature Extraction

- Average Amplitude
- Max Amplitude
- Range of Amplitude

Input:

Individual Ch. Data

400x[14x(128x56)]

Output:

Collective Ch. Data

400x[56x14]

Average Potential for Block 1 Channels

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0.1258 | 0.0415 | 0.0407 | 0.0207 | 0.0060 | 0.0012 | 0.0030 |
| 2 | 0.1948 | 0.0272 | 0.0330 | 0.0131 | 6.2470e-05 | 0.0028 | 0.0023 |
| 3 | 0.0118 | 0.0109 | 0.0055 | 0.0016 | 1.6091e-04 | 3.8594e-04 | 4.5190e-04 |
| 4 | 0.0171 | 0.0101 | 0.0097 | 0.0019 | 4.4266e-04 | 0.0017 | 8.1484e-04 |
| 5 | 0.0030 | 0.0012 | 0.0014 | 4.2826e-04 | 1.1024e-04 | 4.4086e-04 | 2.3261e-05 |
| 6 | 2.0382e-05 | 0.0020 | 0.0016 | 7.1221e-05 | 1.1376e-04 | 2.6157e-04 | 1.5509e-04 |
| 7 | 0.0017 | 0.0014 | 9.4906e-04 | 0.0013 | 6.1541e-04 | 8.8464e-05 | 7.9967e-04 |
| 8 | 0.0048 | 0.0010 | 6.8309e-04 | 0.0010 | 0.0014 | 4.7255e-05 | 7.7436e-04 |
| 9 | 0.0021 | 0.0023 | 0.0011 | 0.0013 | 0.0010 | 7.4907e-06 | 6.2452e-04 |
| 10 | 3.7168e-04 | 8.3394e-04 | 7.7319e-04 | 0.0015 | 7.5122e-04 | 4.1795e-05 | 4.6069e-04 |
| 11 | 0.0028 | 0.0096 | 0.0066 | 0.0017 | 2.8933e-04 | 3.5309e-04 | 1.5966e-05 |
| 12 | 0.0066 | 0.0148 | 0.0086 | 0.0026 | 2.3404e-04 | 8.8439e-04 | 7.7230e-04 |
| 13 | 0.1805 | 0.0234 | 0.0364 | 0.0182 | 0.0064 | 0.0020 | 0.0024 |
| 14 | 0.1163 | 0.0593 | 0.0238 | 0.0086 | 0.0015 | 0.0037 | 0.0049 |

# Neural Network Model

**CNN - TensorFLow | CNN - Keras | RNN - LSTM**

#Training examples = 360

#Testing examples = 40

# CNN - TensorFlow

Epochs = 100

Training Size = 14 (Channels) * 128 (Time Points)

Output y = 0 or 1 (Binary Classification)

Cross entropy = reduce_mean

**Why TensorFlow:**

**Because of the rectangular input size, it is necessary to look into details of CNN to adjust appropriate parameters.**

| Layer | 1 CONV | 2 CONV | 3 CONV | 4 FC | 5 FC |
|---|---|---|---|---|---|
| Filter | 5 * 14 * 1 #1 = 6 | 5 * 14 * 6 #2 = 12 | 5 * 14 * 12 #3 = 24 | 24 * 7 * 32 | 2 (output) |
| Activation | relu | relu | relu | | softmax |
| Pooling | 1 * 2 | 2 * 2 | | | |

# CNN - Keras

Original input x.shape = 128 * 14 (rectangle)

→ Remove the first 2 data in each channel

→ x.shape = 126 * 14 = 42 * 42 (square)

→ Implement by Keras

Why:

Since removing 2 data in each channel gives nearly no difference, input data can be adjusted in to square size and hence Keras Framework is applicable in this case.

Pros:

It becomes more efficient to iterate parameters and change the structure of the neural network to build the best model.

# RNN - LSTM

- ❖ A simple many-to-one model with LSTM Layer.
- ❖ Treat EEG Signals as time series input.
- ❖ Why LSTM: it is more suitable for longer-term dependency and longer sequent input data.

```
model.add(LSTM(128, input_shape=(timesteps, 14)))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])
```

# Model Results - Filtered data

CNN - TensorFlow (45%)

```
reuub17user0@REUUB-17:~/attention/Attention/Original$ python oringinal_modelcnn.
py
/opt/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: C
onversion of the second argument of issubdtype from `float` to `np.floating` is
deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type
`.
  from ._conv import register_converters as _register_converters
/opt/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: Depre
cationWarning: This module was deprecated in version 0.18 in favor of the model_
selection module into which all the refactored classes and functions are moved.
Also note that the interface of the new CV iterators are different from that of
this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
EEGimg_subj.csv
2018-06-26 11:51:24.336570: I tensorflow/core/platform/cpu_feature_guard.cc:137]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: SSE4.1 SSE4.2
step 0, training accuracy 0
step 10, training accuracy 0.5
step 20, training accuracy 0.5
step 30, training accuracy 0.5
step 40, training accuracy 0
step 50, training accuracy 0.5
step 60, training accuracy 0.5
step 70, training accuracy 0.5
step 80, training accuracy 0.5
step 90, training accuracy 0.5
step 100, training accuracy 0.5
step 110, training accuracy 0
step 120, training accuracy 0
step 130, training accuracy 0.5
step 140, training accuracy 0.5
step 150, training accuracy 1
step 160, training accuracy 0
step 170, training accuracy 0.5
step 180, training accuracy 0.5
step 190, training accuracy 1
test accuracy on Phase1 0.45
```

CNN - Keras (42.5%)

```
360/360 [==============================] - 2s 5ms/step - loss: 0.6937 - acc: 0.4861 - val_loss: 0.696
4 - val_acc: 0.4250
Epoch 5/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
7 - val_acc: 0.4250
Epoch 6/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.4972 - val_loss: 0.695
9 - val_acc: 0.4250
Epoch 7/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6937 - acc: 0.4750 - val_loss: 0.696
7 - val_acc: 0.4250
Epoch 8/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.696
4 - val_acc: 0.4250
Epoch 9/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
9 - val_acc: 0.4250
Epoch 10/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
4 - val_acc: 0.4250
Epoch 11/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6937 - acc: 0.4639 - val_loss: 0.695
6 - val_acc: 0.4250
Epoch 12/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
5 - val_acc: 0.4250
Epoch 13/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
5 - val_acc: 0.4250
Epoch 14/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6937 - acc: 0.5083 - val_loss: 0.695
0 - val_acc: 0.4250
Epoch 15/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
2 - val_acc: 0.4250
Epoch 16/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6937 - acc: 0.5000 - val_loss: 0.695
8 - val_acc: 0.4250
Epoch 17/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6937 - acc: 0.4944 - val_loss: 0.696
0 - val_acc: 0.4250
Epoch 18/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.696
1 - val_acc: 0.4250
Epoch 19/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.696
0 - val_acc: 0.4250
Epoch 20/20
360/360 [==============================] - 2s 5ms/step - loss: 0.6938 - acc: 0.5083 - val_loss: 0.695
9 - val_acc: 0.4250
40/40 [==============================] - 0s 365us/step
Test loss: 0.6959069728851318
Test accuracy: 0.425
```

RNN - LSTM (57.5%)

```
None
Epoch 1/20
2018-06-26 11:37:42.791765: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports in
structions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2
 - 34s - loss: 0.6983 - acc: 0.5417
Epoch 2/20
 - 34s - loss: 0.6997 - acc: 0.4417
Epoch 3/20
 - 34s - loss: 0.6960 - acc: 0.4806
Epoch 4/20
 - 34s - loss: 0.6959 - acc: 0.4944
Epoch 5/20
 - 34s - loss: 0.6956 - acc: 0.5056
Epoch 6/20
 - 34s - loss: 0.6966 - acc: 0.4389
Epoch 7/20
 - 34s - loss: 0.6977 - acc: 0.4806
Epoch 8/20
 - 35s - loss: 0.6966 - acc: 0.4528
Epoch 9/20
 - 35s - loss: 0.6979 - acc: 0.4778
Epoch 10/20
 - 35s - loss: 0.6986 - acc: 0.4583
Epoch 11/20
 - 35s - loss: 0.6980 - acc: 0.4694
Epoch 12/20
 - 34s - loss: 0.6985 - acc: 0.4528
Epoch 13/20
 - 34s - loss: 0.6958 - acc: 0.5083
Epoch 14/20
 - 34s - loss: 0.6969 - acc: 0.4778
Epoch 15/20
 - 34s - loss: 0.6943 - acc: 0.5139
Epoch 16/20
 - 34s - loss: 0.6960 - acc: 0.4417
Epoch 17/20
 - 34s - loss: 0.6944 - acc: 0.5028
Epoch 18/20
 - 34s - loss: 0.6973 - acc: 0.4917
Epoch 19/20
 - 34s - loss: 0.6954 - acc: 0.5028
Epoch 20/20
 - 34s - loss: 0.6970 - acc: 0.4833
Accuracy: 57.50%
```

# Model Results - Filtered data

| CNN - TensorFlow | CNN - Keras | RNN - LSTM |
| --- | --- | --- |
| (45%) | (42.5%) | (57.5%) |

Remark:

1. Basically the accuracy is still at a chance level.
2. RNN model runs for a longer time.
3. Change of parameters does not improve the results.
4. Random split of training and testing sets gives random evaluation results. (Cross - Validation)

# Prediction Output

★ With Cross - Validation (10-fold), the prediction of 1st split is output.

★ Prediction of 40 testing examples are almost the same.

★ Trained model cannot differentiate input signals.

Reason:

1. Covered by much noise, EEG signals nearly have no difference → **Better Preprocessing**
2. EEG intensity signals analysis and Neural Network maybe do not match up → **Turn to Frequency Analysis**.
3. Limited by small number of trials. → **Data Augmentation**

True labels:

[1 0]

[1 0]

[1 0]

[1 0]

[1 0]

…

[1 0]

```
[[0.573349   0.42665097]
 [0.57279444 0.42720553]
 [0.5726019  0.42739812]
 [0.57256806 0.42743188]
 [0.57255125 0.42744875]
 [0.57255936 0.4274406 ]
 [0.5725237  0.42747626]
 [0.5725888  0.42741117]
 [0.5725373  0.42746276]
 [0.5726254  0.42737457]
 [0.57305837 0.4269417 ]
 [0.57249385 0.42750618]
 [0.57252014 0.42747986]
 [0.5725681  0.42743188]
 [0.5725568  0.4274432 ]
 [0.57255286 0.42744717]
 [0.57255006 0.4274499 ]
 [0.5722837  0.4277163 ]
 [0.5726323  0.42736766]
 [0.57258546 0.4274145 ]
 [0.57261544 0.42738461]
 [0.5725482  0.42745182]
 [0.5725212  0.42747885]
 [0.5724686  0.42753142]
 [0.5726335  0.42736647]
 [0.5725341  0.42746595]
 [0.57248706 0.42751294]
 [0.57304364 0.42695633]
 [0.57323384 0.4267661 ]
 [0.5726043  0.42739567]
 [0.57250917 0.4274909 ]
 [0.5726551  0.42734498]
 [0.57245874 0.42754126]
 [0.5734186  0.42658138]
 [0.57226557 0.42773443]
 [0.5724375  0.42756245]
 [0.5724966  0.42750338]
 [0.5729259  0.4270741 ]
 [0.572586   0.42741403]
 [0.5726358  0.42736417]]
Train on 360 samples, validate on 40 samples
```

# Future Steps

1. Try to remove some “bad” channels signals before inputting into models. (Preprocessing)

2. Deep Belief Neural network with Stacked Denoising Autoencoder. (Popular model for analyzing EEG)

3. Extract more features from each example. (Enlarge size of each input)

4. Cut the whole time domain (1 second) by a window with fixed size for FFT analysis. (Increase #examples)

# Thank you!

# Q & A