Emotion recognition on the SEED-IV dataset using Convolutional Neural Networks and LSTMs

Akum Kang
Georgia Institute of Technology
kangakum@gatech.edu

Atneya Nair
Georgia Institute of Technology
atneya@gatech.edu

Abstract

Mulitmodal emotion recognition is becoming an increasingly important area of deep learning. Over the last few years, various groups have published extensive datasets which contain experimental information on the physiological signals (EEGs, Eye movements) of subjects while experiencing specific emotions. The goal of such datasets is to encourage development of robust deep learning algorithms for emotion classification. In this paper, we present an analysis of various deep learning methods on the SEED-IV dataset. We achieve an accuracy of 65% using a Deep CNN with 3D convolutional layers. Additionally, we test CNNs with other layer dimensionalities and an LSTM.

1. Introduction

Emotion is a fundamental part of the human experience and greatly influences the way humans think about and interact with each other. As computers become integrated with the daily lives of humans, emotion recognition is becoming more and more important for machines. Emotion recognition can be done through pictures or text, as shown by Bravo-Marquez *et. al*, who were able to classify tweets based on emotion [1].

In recent years, researchers have focused on emotion recognition using EEG signals. The electroencephalogram (EEG) is a test that evaluates the electrical activity of the brain using electrodes placed on the skull. EEG signals are represented as a time series of impulses which indicate the strength of activity in different areas of the brain. Physiological data signal, such as eye movement or EEG signals, are typically not used by humans (at least overtly) in emotion recognition tasks. EEG signals are typically used in a medical context for diagnosing various epileptic disorders, but are also used as an additional prognostic tool for a myriad of brain disorders due to relatively low cost. Despite this, brain-computing interface researchers have identified EEG signals as a potential area of interest for the emotion recognition problem. Emotion classification using EEG signals began with Liu *et. al*, who in 2010 developed an algorithm for real-time emotion classification [2]. In 2015, Zheng *et. al* began the investigation into EEG-based emotion recognition using deep learning algorithms [3]. More recently, there have been countless experiments done on various datasets, all attempting to classify emotion based on EEG signals using deep learning. In 2016, Zheng *et. al* created the SEED-IV dataset and various multimodal classification algorithms for emotion recognition [4]. There have also been attempts at more complex models using graph-based approaches, autoencoders, or spatial-temporal LSTMs. In 2018, Song and Zheng *et. al* developed a method that uses Dynamical Graph CNNs for emotion recognition [5]. These experiments all have similar approaches for classifying emotions, with variations in network structure. Typically, the experiment consists of discriminative EEG feature extraction and emotion classification. Typical feature extraction methods include differential entropy (DE) and power spectral density (PSD) combined with either linear dynamical systems (LDS) or PCA [4][6].

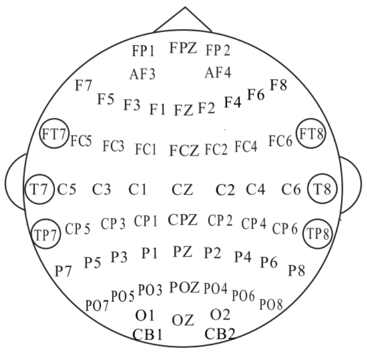
As previously stated, today's researchers are finding success classifying emotions using Deep Neural Networks. A popular type of Deep Neural Network, the Convolutional Neural Network, has been extremely successful in image classification and object detection, among other tasks involving spatial recognition[7].

Another popular type of Deep Neural Network, the LSTM, is a version of the Recurrent Neural Network. LSTMs have proven to be useful in tasks requiring the analysis and classification of time-series data, including EEG-based emotion classification [8]. The success of CNN's in tasks involving spatial detection and classification and LSTMs in tasks involving time-series data leads us to explore the merits of both CNNs and LSTMs in EEG-based emotion classification. Since EEG signals are taken by placing electrodes on the head, it is worth attempting classification based on the spatial placement of electrodes and their corresponding signals using a CNN. Additionally, EEG brain signals are taken over a period of minutes and sampled at a rate of 1000 Hz, so there is certainly enough time series data to warrant attmpting classification using an LSTM.

2. Dataset Specifics

2.1. SEED-IV background

A public dataset, SEED-IV, was used in this paper [9]. The researchers used film clips to elicit specific emotions in subjects. They then asked subjects to categorize their emotion while watching the clips into one of four different categories (happy, sad, neutral, or fear). Each subject participated in 3 sessions of emotion categorization, and each session had 24 trails (6 per emotion). There were a total of 15 participants. In addition, the researchers used 62 EEG channels distributed across the brain as shown below.


FP1 FPZ FP2
AF3 AF4
F7 F5 F3 F1 FZ F2 F4 F6 F8
FT7 FC5 FC3 FC1 FCZ FC2 FC4 FC6 FT8
T7 C5 C3 C1 CZ C2 C4 C6 T8
TP7 CP5 CP3 CP1 CPZ CP2 CP4 CP6 TP8
P7 P5 P3 P1 PZ P2 P4 P6 P8
PO7 PO5 PO3 POZ PO4 PO6 PO8
O1 OZ O2
CB1 CB2

2.2. Preprocessing and Feature-Extraction

The dataset was preprocessed by the same researchers who created it [9]. They applied a band-pass filter between 1 and 75 Hz to filter unrelated artifacts. They then extracted two types of features, power spectral density (PSD) and differential entropy (DE) using fourier transforms with a 4s time window without overlapping. They define in their paper the differential entorpy feature as follows:

$$h(\mathbf{X}) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(x-\mu)^2}{2\sigma^2} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(x-\mu)^2}{2\sigma^2} dx = \frac{1}{2} \log 2\pi e \sigma^2$$

They then computed PSD and DE features in five frequency bands per channel: delta (1-4 Hz), theta (4-8 Hz), alpha (8-14 Hz), beta (14-31 Hz), and gamme (31-50 Hz). They then applied the linear dynamical system (LDS) dimensionality reduction to filter out noise unrelated to EEG features [9][10]. The researchers also filtered out noise by computing the moving average of each trial at various points during data collection.

To organize the data into a format suitable for training deep neural networks, we used a combination of the feature-extracted and dimensionality-reduced data. The original researchers applied 2 feature extraction methods and 2 dimensionality reductions per trial, giving 4 features per trial, namely `de-MovingAve`, `de-LDS`, `psd-MovingAve`, and `psd-LDS`. Each combination gave an array with dimensions $(8, 9, 5, 64)$, representing the 8×9 channel locations on the skull, the 5 frequency bands per channel, and the 64 timestamps per frequency band. Concatenating the 4 feature sets gave an array of size $(4, 8, 9, 5, 64)$ per trial. Since there were 1080 trials (15 participants, 72 trials per participant), our final dataset size was $(1080, 4, 8, 9, 5, 64)$

Additionally, each trial was given a label based on the emotion elicited, 0 for neutral, 1 for sad, 2 for fear, and 3 for happy, giving a corresponding array of labels with size 1080.

3. Model Description

Here we describe technical details of the 4 models used in the experiment. The first two use separable 1D-convolutonal layers, the third uses 3D-convolutonal layers, and the fourth uses an LSTM.

3.1. Convolutional Neural Networks

We first describe the models which use 1D-convolutional layers. The input to these models is an array of size $(1080, 64, 1440)$, where 1080 is the number of trials, 64 is the number of channels per trial, and 1440 is a combination of the other dimensions in the dataset. For each trial, we have a 64×1440 input matrix:

$$x = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,1440} \\ x_{2,1} & x_{2,2} & \dots & x_{2,1440} \\ \dots & \dots & \dots & \dots \\ x_{64,1} & x_{64,2} & \dots & x_{64,1440} \end{pmatrix}$$

We also have a convolutional filters W given by:

$$W_k = (W_1, W_2, W_3)$$

Where $k = 128$ (kernel size of 3 with 128 filters). After applying the filters to the input matrix, we obtain an output:

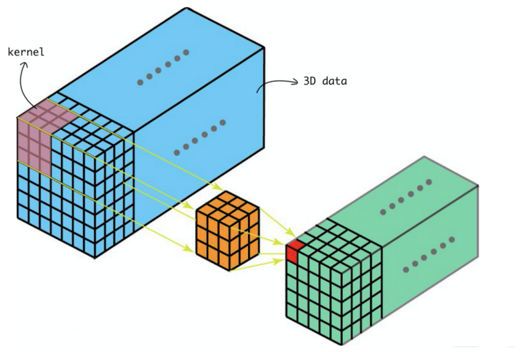
$$\alpha = W_k \times x + b_k$$

Where b_k are the bias values. The convolution is 1-dimensional because each filter acts on each **channel**, moving to the right as it convolves over the channel, and not on the entire input matrix. α is then fed into a Max Pooling layer, which takes the maximum value over each window (we select a window size of 2). The activation function used was $ReLU = max(0, x)$. $Adam$ was the optimizer used. Since we are doing multi-class classification, the loss function used was sparse categorical crossenttropy, given by:

$$\text{loss} = -\sum_{n=1}^{N} \hat{y}_{i1}\log(y_{i1}) + \hat{y}_{i2}\log(y_{i2}) + \hat{y}_{i3} + \log(y_{i3}) + \hat{y}_{i1}\log(y_{i4})$$

The first model, which we call `sc` for single-conv, contains only one convolutional and max-pooling layer. The second model, which we call `fc` for fully-conv, contains 3 convolutional and pooling layer pairs.

The third model, called `3d-conv` uses a 3D-convolution, which instead acts on an input matrix of size $(1080, 64, 8, 9, 20)$. We treat the final dimension of the matrix as the channel count of the convolution, noting that the channels span across both the dimensionality reductions and the frequency bands. Thus we convolve on three dimensions, namely, the two spatial dimensions of electrode placement, as well as the time dimension. The kernel is now three-dimensional instead of one dimensional, but still slides and convolves over the input matrix as in the previous two models. We have included a visual representation below:



This model uses 3-dimensional Max Pooling layers, ReLu activation, an Adam optimizer, and sparse categorical crossentropy loss. Naive implementations of higher order convolutional networks lead to substantial overfitting. As such, in order to combat overfitting, dropout and batch normalization were utilized.

3.2. LSTMs

The final model we experimented with was an LSTM. This model also has an input shape of $(1080, 64, 1440)$.

An LSTM is a type of recurrent neural network (RNN). RNN layers contain large numbers of processing units called cells. Each cell takes the activation from the previous cell a^{t-1}, as well as current input from the dataset x^t. The a^t and the output are calculated as follows:

$$a^t = \sigma(w_{aa}a^{t-1} + w_{ax}x^t + b_a)$$
$$y^t = \tanh(w_{ya}a^t + b_y)$$

Where each cell has weights w_{aa}, w_{ax}, w_{ya} and biases b_a, b_y.

LSTMs are a variation on RNNs that provide a solution for the vanishing gradient problem [11]. Each cell in an LSTM has a state c^t. A candiate for the cell state at timestamp t is represented by:

$$c'^t = \sigma(w_c[a^{t-1}, x^t] + b_c)$$

LSTMs have three core gates which allow them to forget, update, and output information as necessary, thereby combatting the vanishing gradient problem of RNNs. These are the forget, update, and output gate, which are computed as follows:

$$\Gamma_f = \sigma(w_f[a^{t-1}, x^t] + b_f)$$
$$\Gamma_u = \sigma(w_u[a^{t-1}, x^t] + b_u)$$
$$\Gamma_o = \sigma(w_o[a^{t-1}, x^t] + b_o)$$

The cell state and activation are then computed by:

$$c^t = \Gamma_u c'^t + \Gamma_f c^{t-1}$$
$$a^t = \Gamma_o \tanh(c^t)$$

The LSTM we used has 100 cells, has a 128-neuron dense layer after the LSTM layer with a ReLU activation, and uses Adam and Sparse Categorical Crossentropy.

4. Experimental Results

We partioned the processed SEED-IV dataset into a training and validation set. In addition, models were trained utilizing cross-validation within the training set. For all of the models except $3D$ convolution, the spatial dimensions were flattened into the channel axis. We report the final validation accuracy of the models, outside of the cross-validation dataset. As this is a classification problem, we utilize the simple metric of classification accuracy, noting that with four categories, a 25% accuracy would be acheived by a trivial model. Our models acheived the following validation set accuracy $(n = 216)$ on the dataset.

Model	Accuracy (%)
`sc`	57.4
`fc`	54.1
`lstm`	46.7
`3d-conv`	65.3

We note that a recent paper noted a best performing (subject-independent) accuracy of 69.03% on this dataset, with the second best model surveyed performing at 65.59% [12]. These models use alternative, generative architectures, all of which do not utilize spatial convolution across electrode placement. As such, we can see that a $3D$ convolutional approach acheives satisfactory performance on the dataset. We note that the training accuracy of our best performing model was only approximately 10% higher, however the model utilizes aggresive techniques to combat overfitting as previously discussed.

We note that the dataset considers the feature labels (neutral, happy, sad, fear). It is conceivable that additional context will be available in possible brain-computing interface application domains, and as such, accuracy of this level for a relatively simple model indicates promise.

5. Conclusions

In this paper we assess the performance of various deep-learning models for the problem of EEG emotion recognition. The best performing network is a $3D$ convolutional neural network, which performs at parity with other state of the art methods. Thus, we show the effectiveness of considering the relationship of spatially proximal EEG signals when training models in the context of emotion recognition. In continuing work, we seek to implement an additional dimension of convolution, across the frequency domain, in addition to performing additional tuning on the $3D$ convolutional model.

6. References

[1] [Bravo-Marquez et al., 2015] Felipe Bravo-Marquez, Eibe Frank, and Bernhard Pfahringer. Positive, negative, or neutral: Learning an expanded opinion lexicon from emoticon-annotated tweets. In IJCAI'15, pages 1229–1235. AAAI Press, 2015.

[2] [Liu et al., 2010] Yisi Liu, Olga Sourina, and Minh Khoa Nguyen. Real-time EEG-based human emotion recognition and visualization. In 2010 International Conference on Cyberworlds, pages 262–269. IEEE, 2010.

[3] [Zheng and Lu, 2015] Wei-Long Zheng and Bao-Liang Lu. Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks. IEEE Transactions on Autonomous Mental Development, 7(3):162–175, 2015.

[4] [Zheng and Lu, 2016] Wei-Long Zheng and Bao-Liang Lu. Multimodal Emotion Recognition Using Multimodal Deep Learning

[5] [Song and Zheng] EEG Emotion Recognition Using Dynamical Graph Convolutional Neural Networks

[6] [W.-L. Zheng, J.-Y. Zhu, Y. Peng, and B.-L. Lu] EEG-based emotion classification using deep belief networks, in Proc. IEEE Int. Conf. Multimedia Expo, 2014, pp. 1–6

[7] [K. He, X. Zhang, S. Ren, and J. Sun] Deep residual learning for image recognition, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778

[8] [Garg, Bedi, Kapoor, and Sunkaria] Merged LSTM Model for emotion classification using EEG signals

[9] [Wei-Long Zheng, Wei Liu, Yifei Lu, Bao-Liang Lu, and Andrzej Cichocki] EmotionMeter: A Multimodal Framework for Recognizing Human Emotions. IEEE Transactions on Cybernetics, Volume: 49, Issue: 3, March 2019, Pages: 1110-1122, DOI: 10.1109/TCYB.2018.2797176

[10] [L.-C. Shi and B.-L. Lu] Off-line and on-line vigilance estimation based on linear dynamical system and manifold learning, in Proc. Int. Conf. IEEE Eng. Med. Biol. Soc., 2010, pp. 6587–6590.

[11] Hochreiter, Sepp, and Jürgen Schmidhuber. ”Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.

[12] Yang Li, Lei Wang, Wenming Zheng, Yuan Zong, Lei Qi, Zhen Cui, Tong Zhang, and Tengfei Song. ”A Novel Bi-hemispheric Discrepancy Model for EEG Emotion Recognition,” in IEEE Transactions on Cognitive and Developmental Systems.

7. Appendices

7.1. Experimental Details

7.1.1 Algorithms

We use standard CNN and LSTM models, the algorithms for which are described in section 3. Our performance metric is accuracy, which is simply the number of correct labels divided by the total number of trials in the test set.

7.1.2 Source code

Publicly available here

7.1.3 Hardware

The computing infrastructure used for training and running the models in the paper was 1 Tesla V100-PCIE-16GB GPU (through Georgia Tech's PACE computing cluster).

7.1.4 Runtime

The average runtime was about 10 minutes.

7.1.5 Parameters

Details on parameters are included in sections 3 and 4.

7.1.6 Validation Performance

Details on validation performance are included in section 4.

7.2. Hyperparameter search

We tuned hyperparameters (kernel size, filters, LSTM cells) using k-fold cross validation hyperparameter sweeps, and settled on a 128 filters with kernel size of 3 for the 1D CNNs, 32 filters with a kernel size of 10 for the 3D CNN, and 100 LSTM cells.

7.2.1 Datasets

We used the SEED-IV dataset, which is available here after requesting permission from the creators.