Evaluation and Pruning on Neural Network Models for Stress Recognition

Yuanchen Hua

Research School of Computer Science, Australian National University,
Canberra ACT 2600, Australia
u6642108@aun.edu.au

Abstract. The stress problem of people is getting more and more serious, and there have been many studies on stress recognition through RGB images or thermal images. This paper attempts to combine the top 5 components of RGB and the top 5 components of thermal through full connect neural networks and LSTM neural networks to recognize whether people are under stress. Also, this paper will prune the hidden neurons of the full connect network according to the "distinctiveness of hidden units". the accuracy of full connect neural networks can reach about 57%-62%, the accuracy of LSTM neural networks can reach 98% which is higher than [1]. Pruning the trained network by "distinctiveness of hidden units" and merge or offsetting the weights can keep the original accuracy and reduce retraining time.

Keywords: Stress Recognition, Pruning, LSTM, Full Connect.

1 Introduction

The problem of stress is becoming more and more obvious in modern society. People who are under excessive stress for a long time may suffer health damage. In some special occasions such as vehicle driving or aircraft driving, stress may affect people to make inappropriate judgments or operations, resulting in serious consequences. Therefore, we need to recognize whether people are under pressure in a timely manner.

In recent years, there has been more and more research on stress identification. Earlier stress recognition systems include "self-report" and "measure physiological signals using invasive sensors". However, these systems have limitations. For example, "self-report" is not effective, and "invasive sensors" need to be in contact with people, which is inconvenient [2].

Based on the fact that psychological stress will show through some physical appearance changes, some researchers began to recognize the pressure by observing lips, mouth and eyebrows [3][4]. For example, Liao et al [5] establish a stress recognition model based on the blinking frequency, average eyes closure speed and percentage of saccadic eye movement.

Many researchers study how to detect stress by "physiological response", such as "Heartbeat rate"[6], "respiration rate" [7][8], "muscle fatigue"[9], etc. In recent years, we can get these indicators through technologies such as" RGB video recorder" or "thermal imaging". "Physiological response" is getting more and more popular based on the fact that the "physical appearance" is not as reliable as the "physiological response"[1]. We need to pay attention to the "reliability" here. Reliable indicators refer to the phenomena that exist among different individuals. "Physical appearance" is considered to be less reliable, because it contains many individual differences, such as the blinking frequency of some people is relatively high, or the lips of some people are often kept still. The researchers believe that the "physiological response" is more based on the subconscious response. But in fact, the "physiological response" between different individuals also has unreliable factors. For example, under the same pressure, some people have lower facial congestion, while others have higher facial congestion which is more bright in thermal image. In the following sections, we will examine this issue in detail.

Irani et al. use Super-pixels technology to process RGB images and thermal images with SVM classifiers, and then combined their SVMs' result to detect whether a person is under stress [1]. This research is to detect the stress state of people for a continuous period of time. In other words, the input is RGB video and thermal video, and the output is a segmentation of this video, according to the stress state of the person in the video.

Our research data is 10 component of video sampling from RGB and thermal video, instead of RGB or thermal images/video. In other words, our data is the top5 components of the RGB and thermal which are processed by Super-pixels technology. In addition, these data are not continuous, but in some discrete samples, the data size is smaller than [1], and there is no clear time sequence between each piece of data.

Irani et al. combine these SVM results of the two different modalities through a score level fusion [1]. Motivated by the successful application of neural network in different vision algorithms. Some researchers suggest that neural network may be better than hand-crafted feature engineering from experts [10]. For this purpose, we try to build a variety of neural network structures.

In neural network, the number of neurons of input layer and output layer are relatively easy to determine, but the number of neurons in the hidden layer is often difficult to determine. If the number is too small, the network will be difficult to converge. If the number is too large, the model size will increase. At the same time, this will also cause overfitting. Therefore, in this paper, we also prune the neural network following the principle of "distinctiveness of hidden units".

The rest of the paper is organized as follows: Section 2 explains the details of model building, Section 3 discusses the experimental results, and Section 4 concludes the paper.

2 Model Building

This paper process data and construct neural network through PyTorch. PyTorch is an open source machine learning library based on the Torch library. In the code, we set $torch.manual\_seed(20)$ to ensure the initial weights of neural network is repeatable.

2.1 Preprocess Data

First, we preprocess the data. The original data has a total of 620 patterns from 31 people, each with 20 records. We randomly divide the 620 items into 80% and 20%. 80% are used for training and 20% are used for testing. In the code, we set $np.random.seed\ (20)$ to ensure that training set and test set are constant and the 80/20 split is repeated. Each piece of data contains 12 columns of information. One column is ID_Subject, which represents different subjects, that is, 31 people who participated in the test to watch the film which is stress stimulator and were recorded by the video. We move this column to the end to facilitate the writing of the code. There is a column called Labels, which means whether people feel pressure, we use 0 for calm, and 1 for stressful. Columns 3 to 7 are the top 5 RGB components, and columns 8 to 12 are the top 5 thermal components. Due to the large difference in values between different components, we normalize the components of the same type to make them all between -1 and 1. Part of the data after preprocessing is shown in Table 1.

Table 1. Header of preprocessing data.

Labels	RGB_Five_Fist_Top_Components_1	...	Thermal_Five_Fist_Top_Components_5	ID_Subject
1	-0.347366	...	-0.627937	1
1	-0.370564	...	-0.729077	1
1	-0.588750	...	-0.714827	1
0	-1.513246	...	-0.628445	1
0	-1.080018	...	-0.659319	1
...	...	...	...	...

2.2 Loss Function and Model Evaluation

Because [1] takes the accuracy to evaluate the model, we also use the same method in order to compare the results. It is a classification problem, so we use the cross-entropy loss as loss function. PyTorch 's cross-entropy loss function, which contains Softmax activation function [11], can get the

probability distribution that the pattern belongs to and the sum of each category is 1. Pattern is classified into the category with the highest probability, accuracy = number of correct patterns / total number of patterns.

2.3 Neural Network A

Irani et al. classify RGB and thermal separately, and then combine their results [1]. They adopt the formula (**1**) for the fusion of RGB and thermal SVMs:

$$S_{Modal} = \tanh(\gamma.(\omega_1.S_{RGB} + \omega_2.S_T + Threshold)). \qquad (\mathbf{1})$$

where, S_{RGB} is the output of RGB SVM modal, S_T is the output of thermal SVM modal, S_{Modal} is the final output after fusion, $\omega_1$and ω_2 are weight coefficients of RGB and thermal inputs of the fusion, and $Threshold$ is a threshold for making decision if the frame is stressful or not. Frames with corresponding value less than threshold are stressful frame and those larger than threshold are non-stressful [1].

We build a neural network whose structure is similar to the way in [1] handle components. The structure of Network A is shown in Fig. 1.

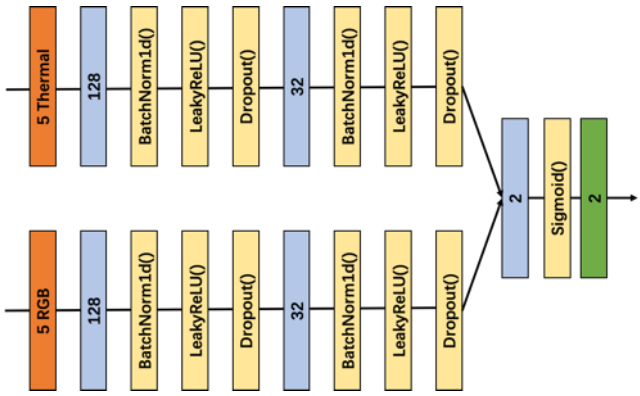

Fig. 1. The Structure of Neural Network A. The orange box is the input layer, the green box is the output layer.

Top five RGB components are calculated through a two-layer network to obtain 32 hidden Neurons. Each layer of the network contains a full connect layer, a LeakyReLU activation function, a BatchNorm layer, a Sigmoid activation function, and a dropout laye. The BatchNorm layer is used to prevent gradient dispersion and overfitting [12], and the dropout layer is used to prevent overfitting [13]. Top five thermal components are calculated through a two-layer network with the same structure as RGB above, and 32 hidden Neurons2 are obtained. We then combine these two types of hidden Neurons as 64 features. We calculate through a layer of full connect network to get the output of feature 2, namely the probability of being recognized as calm, and the probability of being recognized as stressful.

The LeakyReLU layer parameter is 0.1. The dropout layer parameter is 0.2, which means that the probability of an element to be zeroed is 0.2. The optimizer is Adagrad[15], the learning rate is 0.01, the learning rate decay is 0.001, and the training epoch number is 1000.

2.4 Neural Network B

Considering that RGB components and thermal components are calculated separately, the relationship between the two types of components is weakened, so we input 10 components together into a layer of network. As a result, we establish the following network B, as shown in Fig. 2.

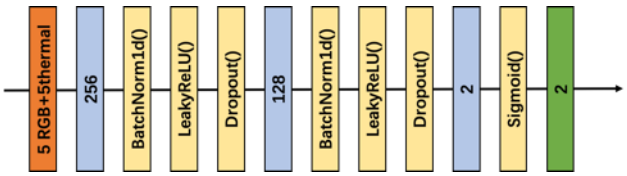

Fig. 2. The Structure of Neural Network B. The orange box is the input layer, the green box is the output layer.

10 components go through a layer of network to get 256 hidden neurons, then go through a LeakyReLU activation function, a BatchNorm layer, and a dropout layer. Then they go through a layer of network, get 64 neurons, and then go through a LeakyReLU activation function, a BatchNorm layer, and a dropout layer. Then the result is connected to a fully connect layer to get the output with feature 2, namely the probability of being recognized as calm and the probability of being recognized as stressful.

The LeakyReLU layer parameter is 0.1. The dropout layer parameter is 0.2, which means that the probability of an element to be zeroed is 0.2. The optimizer is Adagrad[15], the learning rate is 0.01, the learning rate decay is 0.001, and the training epoch number is 1000.

2.5 Neural Network C

Since 620 pieces of data come from the sampling of 31 participants' video, each participant has his/her own unique physiological response baseline under pressure. Therefore, we try to use LSTM as the technology to process the initial data, and then classify through the fully connected layer. So we build neural network C, as shown in Fig. 3.

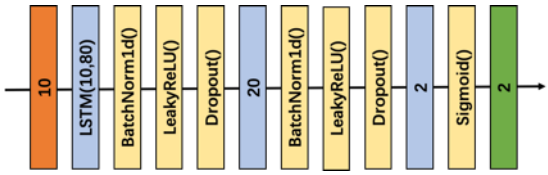

Fig. 3. The Structure of Neural Network C. The orange box is the input layer, the green box is the output layer.

The input size of the LSTM layer is 10, the hidden size is 80, and the number of layers is 1. Similarly, we continue to use the BatchNorm layer to prevent gradient dispersion, and the dropout layer to prevent overfitting. Fully connect layer 1 converts elements from 80 to 20, and fully connect layer 2 converts elements from 20 to 2.

The LeakyReLU layer parameter is 0.1. The dropout layer parameter is 0.2, which means that the probability of an element to be zeroed is 0.2. The optimizer is Adagrad[15], the learning rate is 0.01, the learning rate decay is 0.001, and the training epoch number is 1000.

2.6 Prune Full Connect Layer

Unlike input neurons and output neurons, the number of hidden layer neurons is always difficult to determine, and rules of thumb are often adopted. TD et al. state that the distinctiveness of hidden units is determined from the unit output activation vector over the pattern presentation set and the recognition of similarity of pairs of vectors is done by the calculation of the angle between them in pattern space [14]. We calculate the activation vectors by the following formula (2):

$$A = \mathrm{w} \ @ \ \mathrm{data.T+b} \qquad (2)$$

A represents the matrix composed of activation vectors, *w* represents the weight matrix of the full connect layer, a row in the matrix *w* represents a neuron, *b* represents the bias of the full connect layer, and *data*. *T* represents the transpose matrix of all data which is the input of the full connect layer.

Then, we take each row in the matrix *A* as an activation vector and use each two of these vectors as a pair to calculate the angle between them. Angles less than 15 are considered similar, and angles

greater than 165 are considered opposite [14]. In PyTorch, we use $torch.cosine\_similarity$ () as the calculation method, so a pair of vectors with cosine_similarity greater than 0.9659, that is, their angle is less than 15 degrees. Conversely, if the cosine_similarity of a pair of vectors is less than -0.9659, then their angle is greater than 165 degrees. Matrix A and matrix w have the same number of rows, that is, the number of neurons. If the two horizontal vectors in matrix A are similar, it means that the two neurons in matrix w are similar.

For a similar pair of neurons, we can delete one of them, and merge the weight and bias of the deleted neuron to the non-deleted neuron which is called merge rule. For the opposite pair of neurons, since they always give the opposite effect on the pattern, we will delete this pair of neurons which is called delete rule. According to this theory, we pruned the full connect layer in network C and re-assign the weight matrix to get a new neural network.

3 Results and Discussion

The training result of neural network A and neural network B is shown in Fig. 4. Same as [1], it combines RGB and thermal, and the accuracy of network A in the training set is about 57%, and [1] gets 89%. There is still a big gap between the two. The accuracy of neural network B in the training set is 62%, and there is still a certain gap compared with 89% in [1]. But compared to neural network A, which processes RGB and thermal separately, the accuracy of neural network B is improved by about 5 percent.

In addition, the accuracy of Network A in the test set is about 42%. This shows that even if Network A adopts BatchNorm and dropout technologies, there is still a problem of overfitting. Similarly, the accuracy of neural network B in the test set are about 42%, so neural network B also have the issue of overfitting.

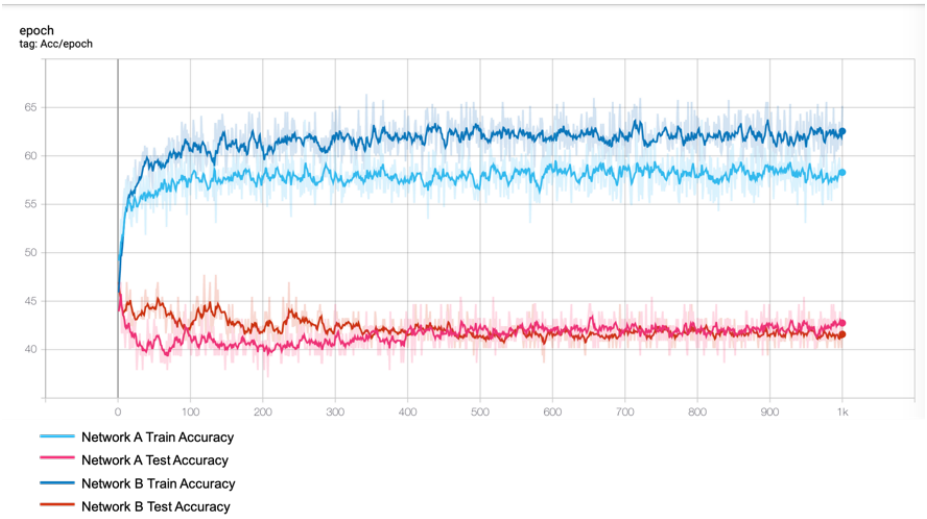

Fig. 4. The Accuracy of Neural Network A and Neural Network B. The horizontal axis is epoch and the vertical axis is accuracy percentage.

The training results of neural network C are shown in Fig. 5.

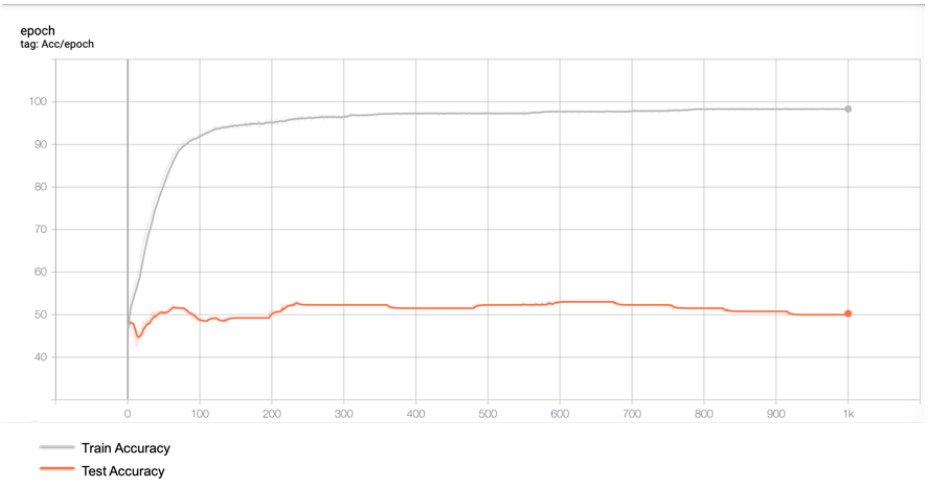

Fig. 5. The Accuracy of Neural Network C. The horizontal axis is epoch and the vertical axis is accuracy percentage.

The accuracy of neural network C in the training set is about 98%. Compared with neural network B, neural network C increase by 36%. The difference between neural network C is that the neural network C uses a LSTM layer to process the input raw data. Network C takes into account individual participant information. This shows that there are also unreliable factors of "physiological response" among the individual participants. In other words, under the same stress state, the RGB information and thermal information expressed by different individuals are also different. Therefore, if our network can learn the RGB and thermal information of the same individual, it can more accurately determine the current stress state of this person. Similarly, the accuracy of neural network C in the test set are about 50%, so neural network C also have the issue of overfitting. Since the accuracy of A, B, C in the test set are all equivalent, we infer that the overfitting issue is caused by the small size of the data set.

The training results of neural network C and pruned neural network C are shown in Fig. 6. Matrix A of first hidden full connect layer of network C is shown in Table 2. Because it is a 20 * 488 matrix, Table 2. omits part of the data display.

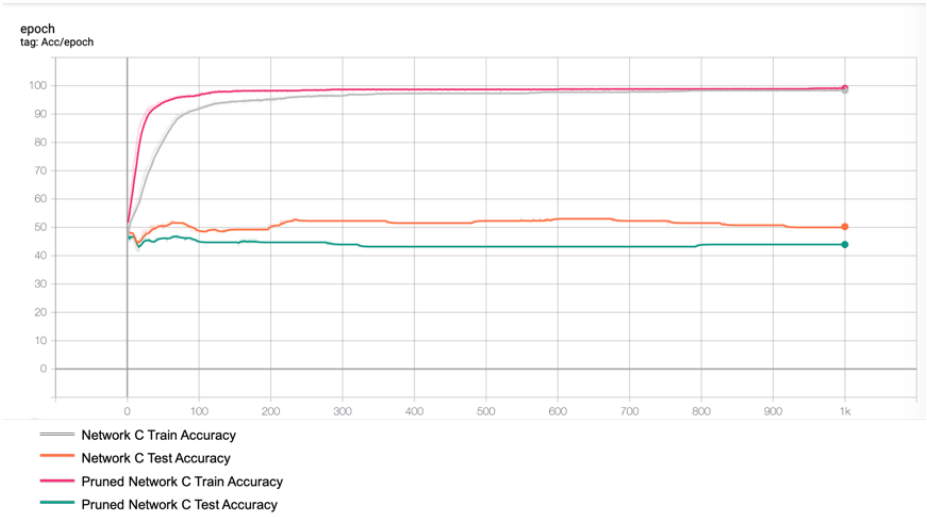

Fig. 6. The Accuracy of Neural Network C and Pruned Neural Network C. The horizontal axis is epoch and the vertical axis is accuracy percentage.

Table 2. Matrix *A*.

			...			
-0.0575	-0.0102	-0.0576	...	-0.0458	-0.0456	-0.0464
0.0797	0.0857	0.0958	...	0.0972	0.0955	0.0915
-0.0530	-0.0621	-0.0556	...	-0.0419	-0.0422	-0.0424
...	...	...	...	...	...	...

0.0829	0.0698	0.0673	...	0.1038	0.1033	0.1024
0.0267	0.0308	0.0310	...	-0.0210	-0.0209	-0.0213
-0.1020	-0.0621	-0.1066		-0.0591	-0.0576	-0.0561

The similarity calculation shows that the angle between the 5th row and the 20th row of matrix *A* is less than 15 degrees, so the 5th neuron in matrix *w* is similar to the 20th neuron, and we merge them. There are no pair vectors with an angle greater than 165 in matrix *A*, so there are no neurons with the opposite effect in matrix *w*.

The vector of the 5th and 20th rows in matrix *A* are as follows in Table 3.:

Table 3. The Vector of The 5th and 20th Rows in Matrix *A*.

[-0.1419, -0.0905, -0.1599, -0.1523, -0.1386, -0.1340, -0.1329, -0.1325, -0.1355, -0.1351, -0.1338, -0.1309, -0.1316, -0.1314, -0.1292, -0.1271, -0.0701, -0.1178, -0.1195, -0.1163, -0.1154, -0.1108, -0.1052, -0.1020, -0.1001, -0.1049, -0.0997, -0.0671, -0.0651, -0.0646, -0.0650, -0.0635, -0.0639, -0.0947, -0.0909, -0.0937, -0.0729, -0.1056, -0.1026, -0.0984, -0.0991, -0.0991, -0.0973, -0.0976, -0.1005, -0.0971, -0.1001, -0.0981, -0.0988, -0.0972, -0.0952, -0.0713, -0.0413, -0.0629, -0.0992, -0.0608, -0.0604, -0.0724, -0.0798, -0.0796, -0.0796, -0.0834, -0.0899, -0.0943, -0.0974, -0.0957, -0.0932, -0.0931, -0.1651, -0.1645, -0.1404, -0.1337, -0.1615, -0.1527, -0.1580, -0.1593, -0.1587, -0.1595, -0.1613, -0.1604, -0.1619, -0.1610, -0.1617, -0.1613, -0.0846, -0.0943, -0.1067, -0.1044, -0.1115, -0.1125, -0.1125, -0.1132, -0.1134, -0.1138, -0.1123, -0.1115, -0.1118, -0.1128, -0.1136, -0.1140, -0.0959, -0.0651, -0.0603, -0.0725, -0.0672, -0.0811, -0.0744, -0.0751, -0.0778, -0.0757, -0.0735, -0.0722, -0.0714, -0.0734, -0.0715, -0.1478, -0.0893, -0.0915, -0.0890, -0.0878, -0.1078, -0.0678, -0.0874, -0.0972, -0.0972, -0.0971, -0.1031, -0.0938, -0.0956, -0.0920, -0.0787, -0.0714, -0.0474, -0.0499, -0.0300, -0.0545, -0.0519, -0.0525, -0.0549, -0.0554, -0.0571, -0.0564, -0.0548, -0.0559, -0.0543, -0.0561, -0.0575, -0.0488, -0.0425, -0.0443, -0.0417, -0.0365, -0.0408, -0.0430, -0.0400, -0.0426, -0.0391, -0.0350, -0.0370, -0.0368, -0.0358, -0.0379, -0.1149, -0.0587, -0.0583, -0.0836, -0.0826, -0.0802, -0.0659, -0.0765, -0.0775, -0.0745, -0.0729, -0.0811, -0.0812, -0.0800, -0.0781, -0.1244, -0.1302, -0.1434, -0.1483, -0.1440, -0.1431, -0.1436, -0.1414, -0.1452, -0.1433, -0.1457, -0.0783, -0.0280, -0.1058, -0.1073, -0.1068, -0.1046, -0.1077, -0.1067, -0.1081, -0.1095, -0.1088, -0.1111, -0.1091, -0.1103, -0.1106, -0.1124, -0.0533, -0.0303, -0.0795, -0.0760, -0.0713, -0.0740, -0.0696, -0.0677, -0.0662, -0.0636, -0.0633, -0.0636, -0.0653, -0.0598, -0.1110, -0.0991, -0.0372, -0.0635, -0.0594, -0.0925, -0.0945, -0.0938, -0.0929, -0.0932, -0.0914, -0.0952, -0.0899, -0.0910, -0.0898, -0.0906, -0.1181, -0.1209, -0.1221, -0.1215, -0.1213, -0.1210, -0.0946, -0.0881, -0.0879, -0.0879, -0.0881, -0.0890, -0.1274, -0.1300, -0.1210, -0.1256, -0.0802, -0.1146, -0.1322, -0.1321, -0.1292, -0.1316, -0.1325, -0.1304, -0.1328, -0.1310, -0.1326, -0.1299, -0.1235, -0.1301, -0.1287, -0.1299, -0.1218, -0.0970, -0.1098, -0.1093, -0.1055, -0.1093, -0.1093, -0.1002, -0.1056, -0.1164, -0.1001, -0.0995, -0.0982, -0.1002, -0.1003, -0.0855, -0.0583, -0.0788, -0.0698, -0.0680, -0.0702, -0.0683, -0.0689, -0.0714, -0.0700, -0.0701, -0.0679, -0.0658, -0.0677, -0.0993, -0.0967, -0.0988, -0.0749, -0.0852, -0.0890, -0.0873, -0.0937, -0.0878, -0.0875, -0.0886, -0.0853, -0.0867, -0.0832, -0.0869, -0.0864, -0.0867, -0.1295, -0.1176, -0.1228, -0.1297, -0.1303, -0.1321, -0.1303, -0.1329, -0.1288, -0.1278, -0.1269, -0.1274, -0.1245, -0.1276, -0.1222, -0.1282, -0.0700, -0.0893, -0.0378, -0.0721, -0.0725, -0.0719, -0.0699, -0.0727, -0.0697, -0.0717, -0.0703, -0.0725, -0.0551, -0.0568, -0.0441, -0.0776, -0.0783, -0.0770, -0.0776, -0.0796, -0.0913, -0.0791, -0.0783, -0.0833, -0.0839, -0.0829, -0.0901, -0.0801, -0.0520, -0.0841, -0.0799, -0.0815, -0.0829, -0.0828, -0.0805, -0.0806, -0.0790, -0.0805, -0.0827, -0.0811, -0.0815, -0.0809, -0.0848, -0.0806, -0.1243, -0.1259, -0.0878, -0.1266, -0.1466, -0.1468, -0.1506, -0.1523, -0.1502, -0.1484, -0.1476, -0.1491, -0.1493, -0.1484, -0.1491, -0.1470, -0.1474, -0.1052, -0.0866, -0.0701, -0.0914, -0.0992, -0.1015, -0.0986, -0.0968, -0.0963, -0.0938, -0.0916, -0.0969, -0.0942, -0.0956, -0.0923, -0.0937, -0.0920, -0.0914, -0.0882, -0.0775, -0.1084, -0.1517, -0.1517, -0.1357, -0.1244, -0.1300, -0.1286, -0.1276, -0.1278, -0.1521, -0.1518, -0.1528, -0.1529, -0.1535, -0.1498, -0.1500, -0.1510, -0.0893, -0.0805, -0.0661, -0.0688, -0.0530, -0.0442, -0.0830, -0.0986, -0.0953, -0.0979, -0.0986, -0.0986, -0.0999, -0.0992, -0.0896, -0.0673, -0.1075, -0.1089, -0.1074, -0.1058, -0.1039, -0.1018, -0.0551, -0.1003, -0.1028, -0.1031, -0.1028, -0.1035, -0.1052, -0.1023, -0.1003, -0.1012, -0.0696, -0.1095, -0.1087, -0.1066, -0.1078, -0.1066, -0.1058, -0.1056, -0.1059, -0.1053, -0.1051, -0.1071, -0.1067, -0.1063, -0.1058, -0.0643, -0.0379, -0.0560, -0.0080, -0.0420, -0.0243, -0.0460, -0.0544, -0.0535, -0.0535, -0.0511, -0.0524, -0.0528, -0.0510, -0.0497]	[-0.1020, -0.0621, -0.1066, -0.1089, -0.0939, -0.0916, -0.0895, -0.0945, -0.0913, -0.0940, -0.0929, -0.0894, -0.0892, -0.0929, -0.0889, -0.0878, -0.0595, -0.0974, -0.0963, -0.0949, -0.0958, -0.0901, -0.0875, -0.0880, -0.0791, -0.0893, -0.0815, -0.0664, -0.0625, -0.0607, -0.0606, -0.0596, -0.0593, -0.1004, -0.0964, -0.0941, -0.0740, -0.1005, -0.0999, -0.0987, -0.1006, -0.1008, -0.1010, -0.1011, -0.1019, -0.1005, -0.1015, -0.1019, -0.1018, -0.0999, -0.0996, -0.0794, -0.0651, -0.0877, -0.0989, -0.0790, -0.0786, -0.0818, -0.0857, -0.0864, -0.0870, -0.0888, -0.0908, -0.0925, -0.0936, -0.0935, -0.0920, -0.0916, -0.1113, -0.1136, -0.0997, -0.1226, -0.1214, -0.1043, -0.1121, -0.1108, -0.1110, -0.1115, -0.1135, -0.1109, -0.1142, -0.1132, -0.1132, -0.1132, -0.0544, -0.0716, -0.0758, -0.0742, -0.0823, -0.0813, -0.0803, -0.0825, -0.0802, -0.0826, -0.0824, -0.0841, -0.0841, -0.0813, -0.0824, -0.0812, -0.1112, -0.0894, -0.0812, -0.0915, -0.0919, -0.0951, -0.0945, -0.0934, -0.0964, -0.0946, -0.0930, -0.0913, -0.0917, -0.0919, -0.0920, -0.0845, -0.0471, -0.0516, -0.0382, -0.0394, -0.0987, -0.0344, -0.0414, -0.0575, -0.0569, -0.0583, -0.0684, -0.0515, -0.0509, -0.0494, -0.0348, -0.0362, -0.0674, -0.0843, -0.0516, -0.0737, -0.0719, -0.0719, -0.0743, -0.0748, -0.0755, -0.0759, -0.0756, -0.0763, -0.0754, -0.0754, -0.0782, -0.0155, -0.0121, -0.0567, -0.0045, -0.0019, -0.0033, -0.0019, -0.0007, -0.0023, -0.0032, 0.0018, -0.0009, -0.0021, 0.0001, -0.0010, -0.0962, -0.0655, -0.0528, -0.0826, -0.0834, -0.0823, -0.0753, -0.0809, -0.0819, -0.0810, -0.0805, -0.0832, -0.0839, -0.0841, -0.0825, -0.0876, -0.0868, -0.1005, -0.0983, -0.1025, -0.1053, -0.1025, -0.1028, -0.1029, -0.1034, -0.1045, -0.0577, -0.0203, -0.0895, -0.0888, -0.0900, -0.0871, -0.0894, -0.0903, -0.0888, -0.0910, -0.0929, -0.0927, -0.0933, -0.0951, -0.0948, -0.0905, -0.0810, -0.0728, -0.0907, -0.0886, -0.0853, -0.0875, -0.0849, -0.0836, -0.0835, -0.0828, -0.0829, -0.0826, -0.0836, -0.0813, -0.1118, -0.1024, -0.0397, -0.0703, -0.0638, -0.0943, -0.0979, -0.0969, -0.0967, -0.0949, -0.0933, -0.0947, -0.0927, -0.0949, -0.0945, -0.0949, -0.0852, -0.0865, -0.0857, -0.0844, -0.0841, -0.0852, -0.1193, -0.1148, -0.1144, -0.1143, -0.1145, -0.1155, -0.0910, -0.1094, -0.0980, -0.0970, -0.0940, -0.0978, -0.1139, -0.1121, -0.1118, -0.1140, -0.1140, -0.1140, -0.1152, -0.1131, -0.1140, -0.1135, -0.1097, -0.1142, -0.1146, -0.1152, -0.1059, -0.0803, -0.1015, -0.0978, -0.0944, -0.0971, -0.0975, -0.0940, -0.0913, -0.0827, -0.0959, -0.0939, -0.0938, -0.0963, -0.0970, -0.0765, -0.0664, -0.0844, -0.0695, -0.0702, -0.0717, -0.0716, -0.0691, -0.0717, -0.0706, -0.0705, -0.0696, -0.0667, -0.0678, -0.1009, -0.1020, -0.1033, -0.0879, -0.0909, -0.0909, -0.0933, -0.0937, -0.0926, -0.0898, -0.0913, -0.0899, -0.0899, -0.0906, -0.0915, -0.0908, -0.0907, -0.1000, -0.0891, -0.1213, -0.0948, -0.0970, -0.0857, -0.0897, -0.0899, -0.0863, -0.0867, -0.0871, -0.0867, -0.0847, -0.0840, -0.0838, -0.0882, -0.0834, -0.0998, -0.0600, -0.0860, -0.0859, -0.0857, -0.0842, -0.0847, -0.0834, -0.0845, -0.0841, -0.0858, -0.0720, -0.0727, -0.0726, -0.0756, -0.0770, -0.0781, -0.0760, -0.0762, -0.0794, -0.0765, -0.0760, -0.0808, -0.0816, -0.0806, -0.0813, -0.0704, -0.0460, -0.0629, -0.0670, -0.0678, -0.0677, -0.0684, -0.0700, -0.0697, -0.0650, -0.0682, -0.0673, -0.0696, -0.0660, -0.0680, -0.0648, -0.0699, -0.0887, -0.0887, -0.0612, -0.0917, -0.1022, -0.1029, -0.1018, -0.1036, -0.1018, -0.1038, -0.1037, -0.1049, -0.1067, -0.1038, -0.1049, -0.1010, -0.1000, -0.1166, -0.1081, -0.0871, -0.0880, -0.0995, -0.1007, -0.0995, -0.0993, -0.0993, -0.0984, -0.0959, -0.1002, -0.0973, -0.0984, -0.0978, -0.0948, -0.0948, -0.0955, -0.0947, -0.0551, -0.1033, -0.1152, -0.1114, -0.0980, -0.0933, -0.0615, -0.0606, -0.0602, -0.0602, -0.1139, -0.1133, -0.1136, -0.1130, -0.1134, -0.1134, -0.1143, -0.1134, -0.0906, -0.0896, -0.0814, -0.0892, -0.0650, -0.0666, -0.0879, -0.0894, -0.0868, -0.0878, -0.0894, -0.0883, -0.0888, -0.0887, -0.0773, -0.0595, -0.0935, -0.0929, -0.0917, -0.0919, -0.0889, -0.0892, -0.0602, -0.0885, -0.0891, -0.0891, -0.0894, -0.0891, -0.0905, -0.0857, -0.0841, -0.1102, -0.0933, -0.1114, -0.1110, -0.1088, -0.1091, -0.1109, -0.1095, -0.1092, -0.1095, -0.1091, -0.1101, -0.1113, -0.1117, -0.1102, -0.1095, -0.0723, -0.0529, -0.0678, -0.0315, -0.0539, -0.0584, -0.0564, -0.0631, -0.0627, -0.0625, -0.0599, -0.0596, -0.0591, -0.0576, -0.0561]

The first hidden full connect layer of neural network C has 20 neurons, while the pruned neural network C only has 19 neurons.

We find that the neural network C train 400 epochs and then converge, and the pruned neural network C with original weight data converge after about 150 epochs, and the final accuracy was about 98%, which is same as the original neural network C. Through "distinctiveness of hidden units", we can indeed prune the neural network and retaining the weights through merge or delete rules can make the network have high accuracy without retraining. The network that retains the original weight data after pruning is retrained, and the network can also converge very quickly.

4 Conclusion and Future Work

Using full connect neural networks is worse than [1]. When we use LSTM technology to process the input data and use the fully connect layer for classification, the accuracy rate has been significantly improved. At present, the data we use is only the top 5 component of the video data after sampling. In the future, we can also use CNN technology to directly process video data.

Through LSTM technology, the characteristics of each individual's response to stress are taken into account, we find that the accuracy of the network's recognition of stress is improved, which shows that there is a certain degree of unreliability in the "physiological response" of different individuals. In the future stress identification model, you can consider using a small amount of time to pre-learn the stress data of the person in order to be more accurate in the subsequent long-term detection. At present, the research on the stress recognition model is a general research on all human beings. In the future, we can try to conduct study about rapid stress recognition modeling for individuals based on a certain general model.

The vector similarity method we currently use for pruning can only prune full connect layer weights, and cannot be used for networks such as LSTM. This method is limited. In the future, we need to study pruning methods for other networks.

The issue of overfitting in the neural network model build in this paper still exists. We analyze from two aspects: on the one hand, the network is too complicated. The RGB and thermal component are PCA eigenvalues, ranging from -10^6 to 10^6, it is weakly-informative so a complex network is required. On the other hand, it may result from the limitation of data. Combining these two points, a more complex network with insufficient data are more likely to produce overfitting.

5 References

1. Irani, R., Nasrollahi, K., Dhall, A., Moeslund, T., Gedeon, T.: Thermal super-pixels for bimodal stress recognition. 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA). (2016).
2. Chen, T., Yuen, P., Richardson, M., Liu, G., She, Z.: Detection of Psychological Stress Using a Hyperspectral Imaging Technique. IEEE Transactions on Affective Computing. 5, 391-405 (2014).
3. Metaxas, D., Venkataraman, S., Vogler, C.: Image-Based Stress Recognition Using a Model-Based Dynamic Face Tracking System. Computational Science - ICCS 2004. 813-821 (2004).
4. Dinges, D., McGlinchey, E., Venkataraman, S., & Metaxas, D.: Optical computer recognition of behavioral stress in space flight. Habitation International Journal for Human Support Research. 10(3/4),233 (2006).
5. Pavlidis, I., Levine, J.: Thermal image analysis for polygraph testing. IEEE Engineering in Medicine and Biology Magazine. 21, 56-64 (2002).
6. Balakrishnan, G., Durand, F., Guttag, J.: Detecting Pulse from Head Motions in Video. 2013 IEEE Conference on Computer Vision and Pattern Recognition. (2013).
7. Fei, J., Pavlidis, I.: Thermistor at a Distance: Unobtrusive Measurement of Breathing. IEEE Transactions on Biomedical Engineering. 57, 988-998 (2010).
8. Fei, J., Zhu, Z., Pavlidis, I.: Imaging Breathing Rate in the CO 2 Absorption Band. 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference. (2005).
9. Irani, R., Nasrollahi, K., Moeslund, T.: Contactless measurement of muscles fatigue by tracking facial feature points in a video. 2014 IEEE International Conference on Image Processing (ICIP). (2014).
10. Kennardi, A., Plested, J.: Evaluation on Neural Network Models for Video-Based Stress Recognition. Communications in Computer and Information Science. 440-447 (2019).
11. PyTorch master documentation, https://pytorch.org/docs/stable/nn.html#crossentropyloss.
12. Ioffe, S., & Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167. (2015).
13. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958. (2014).
14. Gedeon, T. D., & Harris, D.: Network reduction techniques. In Proceedings International Conference on Neural Networks Methodologies and Applications, vol. 1, pp. 119-126. (1991).
15. PyTorch master documentation, https://pytorch.org/docs/stable/optim.html?highlight=adagrad#torch.optim.Adagrad.