Deep Residual Learning For Stress Recognition

Yu Cui[1]

Research School of Computer Science,
Australian NationalUniversity,
Canberra Australia
u7016518@anu.edu.au

Abstract. Stress is a very common problem in modern society. Some stress may not be harmful but other stress may cause serious problems. Thus, developing a reliable stress recognition system is necessary. In this paper, I propose a new contact-free stress recognition neural network architecture - The fused Bi-Cascade Model. This architecture takes RGB and thermal pictures as input. The features are extracted using two pre-trained ResNet [1].Two improved cascade models predict the stress based on the extracted features. The predictions are fused by a multilayer perceptron. Compare to traditional contact-based systems, this architecture helps us get rid of the necessity of touching subjects. The deep learning-based method can provide better high-level features than SVM-based method [9].The improved cascade model (Casada) proposed in this paper overcomes the drawbacks of traditional cascade models [1]. It requires fewer neurons to achieve the better accuracy than traditional models. And it provides a smoother decision edge compare to [1]. It also converges faster than previous works [2] with higher accuracy. Overall the proposed model is a fast and robust stress recognition architecture.

Keywords: stress recognition · residual learning · ResNet · feature extraction · RGB images · thermal images · cascade neural network · Casada · Adam optimizer · Fused Bi-Cascade Model.

1 Introduction

1.1 Stress

Stress is a feeling of emotional or physical tension.[25] It can come from a lot of things or thoughts. They may make us feel angry, nervous. Stress will come after these feelings. Stress is how we react to these things. Some stress goes away quickly. This kind of short-term stress generally won't hurt our health. But some kinds of stress may hide behind other feelings or thoughts and last for a very long time,which are called chronic stress. If we are so used to chronic stress that we don't even consider it as a problem, it can lead to some serious health problems. It's necessary to find a way to help people realize that they are suffering from chronic stress. Some previous works have made some progress in this field. Traditional stress recognition systems use invasive sensors for physiological signal gathering. These systems have some obvious drawbacks. When gathering data using invasive sensors, subjects may have different physiological signals just because of the invasive sensors attached to them. Furthermore, invasive sensors can't monitor the subjects instantaneously and continuously [9], which makes it hard to get a large dataset with accurate records. So, in this assignment, RGB and thermal cameras are used to gather the dataset. RGB and thermal cameras are contactless sensors, which can minimize the negative impacts caused by the data gathering process.

1.2 Feature Extraction and Representation Learning

Generally speaking, images are very hard to understand by the typical neural networks. It's basically because the images are very high-dimension data. It's hard for the neural network to get useful information from millions of dimensions. So, we have to extract features from the images. This is also related to the dimensionality reduction. Many previous works have been done on feature extraction. For example, Principal component analysis (PCA) [11] and Fisher's Linear discriminant analysis (LDA) [12]. These works provide us some efficient methods to reduce the dimensionality while preserving the useful information. However, they can only learn linear mapping. Later works, like Kernel PCA [13] and Isomap [14], can also map the non-linear relations. [9] introduced a stress recognition method based on super-pixel which is also a well-known dimensionality reduction method. In recent years, with the rapid growth of neural network, convolutional neural networks (CNN) are developed for image feature extraction. The aim no longer restricted to dimensionality reduction. The networks are expected to learn a useful representation which can work well with the classifier. CNN goes deeper and deeper to learn some very high-level features. However, a degradation problem has been exposed. With the depth increasing, accuracy stays the same. Surprisingly, overfitting is not the one to blame. [15,16] reported a higher training error after adding more layers to the network. The degradation is detailly analyzed by K. He et al. [17]. The real reason for the degradation is the different optimization difficulties on different models. They proposed a deep residual learning framework (ResNet). ResNet can theoretically go infinitely deep while providing a high-level representation that can be easily used by the classifier. In this paper, a pre-trained ResNet is hired to the learn representation of RGB and thermal Images.

1.3 Cascade Neural Network

To extract the relation between the features and modality, I decide to employ a neural network for classification. In some recent works, neural networks are used in classification which had some very successful applications. Compared to some nonparametric systems [9] used to solve this problem, neural network has some advantages. For example, [9] used an extra equation to fuse two results from different sensors. Neural network can naturally fuse the results from different sensors using the late layers, which is also a more reasonable way to fuse the results rather than simply linear combine two results [9].

Typically, the architecture of the neural network, the number of neurons, and the learning rate are fixed for a specific neural network. However, only after the training is finished can we know the performance of the neural network. Finding the best architecture and hyperparameters takes a considerable amount of time, considering some neural networks may be very complex and have a large number of neurons. So, can we learn and update the hyperparameters during the training so that we don't need to try and test every single possible combination of hyperparameters? The Cascade-Correlation Learning Architecture (Cascor) [1] is one of the solutions. Instead of using a fixed neural network architecture. Cascor starts training with a minimal network that only contains some inputs and outputs, which means Cascor has no hidden neurons at the beginning. When the error no longer decreases, a new neuron will be added to Cascor and the training will continue to maximize the correlation between this neuron's output and the residual error. During the training, the weights of all pre-existing neurons are frozen. This algorithm is very fast considering that there is no backpropagation of error on these frozen weights. Cascor's idea is similar to some greedy algorithms.

Cascor has some potential drawbacks. First, the weights of early neurons are used for feature extraction. However, these weights are only trained for a very short time before most of the neurons added to the neural network. Such an early weight freezing can result in poor feature extraction. Therefore, Cascor needs more neurons to reduce the error caused by the poor feature extraction. This drawback is pointed out by Kwok and Yeung's work [10]. They find that Cascor tends to be an excessively large network than other neural network architectures due to the mentioned drawback. Second, normal neural network architecture's neurons are relevantly sparse which makes the edge between the data points relevantly smooth. Cascor adds a new neuron after the original neurons can't reduce the error anymore, which forces every neuron to fit the data points as much as possible. The edge varies widely between data points. Therefore, Cascor may not generalize well on regression tasks.

Later work [4] proposed the Casper architecture which is an improved Cascade neural network. In order to avoid the drawbacks of the traditional Cascade neural networks. No weights are frozen during the train. After a new neuron is added to the network, all early neurons are continued to be trained in a relevantly lower learning rate. Thus, the early neurons can be refined to get a better feature extraction. In this paper, I propose the Casada architecture which is an improved Casper model. Casada inherits the advantages of Casper. The optimization method is improved. Casada provides us a faster convergence. It also needs fewer neurons for the better accuracy.

2 Method

2.1 Fused Bi-Cascade Model

The ANUStressDB provides both RGB and thermal images for stress recognition. As we know, different sensors capture different kinds of features. Thermal sensors can provide the thermal features while the RGB sensors provide the color features. So, it's reasonable to use two different CNN to extract features and learn the representation. Similarly, using different classifiers for different sensors is also necessary. Because of the adaptive architecture and relatively low computational cost, cascade neural network is chosen for the task. The details of the architecture will be discussed in 2.3. Two models are trained separately on different sensors. Then the predictions of two Casada models are concatenated together and pass through an MLP to fuse the predictions. The output of MLP will indicate the possibility of stress. Figure 1 shows the Fused Bi-Cascade Model.

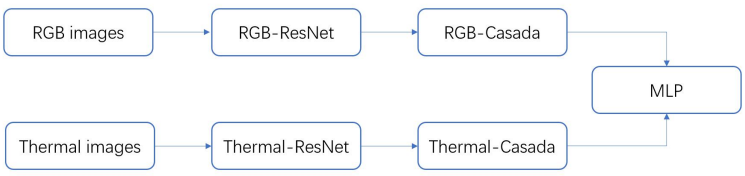

Fig. 1. The flow chart of Fused Bi-cascade architecture

2.2 Residual Learning and ResNet

As we have mentioned in the introduction. To avoid the degradation, K. He et al. introduced a deep residual learning framework. Consider a few stacked layers, we expect them to learn an underlying mapping $H(x)$. Instead of letting the network fit the desired mapping $H(x)$, we optimize them using the residual function $F(x) := H(x) - x$. Then the original mapping can be expressed as $H(x) = F(x) + x$. Both forms are used to approximate the desired function. However, the difficulty of learning may be different. Adding too many layers to the network will result in the degradation, which suggest that the optimizer may have trouble in approximating the identity mapping. The reformulation of $H(x)$ can help the network avoid the approximation of identity mapping. When x is already an optimal representation, $H(x) - x$ simply becomes 0. The optimizer only needs to push it to zero, which is a much simpler problem.

When training a complex neural network, identity mapping may not be optimal even for few stacked layers. But the reformulation can still help to precondition the problem. At least when the optimal function is close to the identity mapping, it should be easier for the optimizer to find the offset with reference to an identity mapping, than to learn it as a new one. In real cases, residual learning is implemented by shortcuts. Residual learning is applied to every few stacked layers. Figure 2 shows the building block. It can be formally defined as:

$$y = F(x, \{W_i\}) + x \tag{1}$$

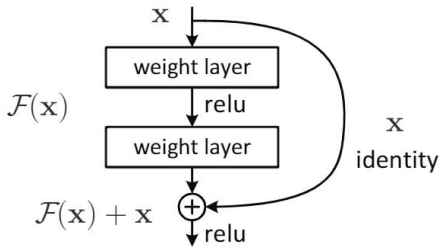

Fig. 2. The residual connection building block [21]

Take fig2 as example, the function can be expressed as:

$$F = W_2\sigma(W_1 x) \tag{2}$$

Where W_1 and W_2 are the weights and σ is the activation function. biases are omitted in this representation. $F + x$ simply performed an element-wise addition. This operation requires F and x have the same dimensionality. When the shortcut connection is applied to the stacked layers that have no channel change, it won't introduce extra computation complexity and memory consumption. When the input/output channels change, we can perform a linear projection W_s so that the shortcut connection can match the dimensions. The extra costs are still acceptable low.

$$y = F(x, \{W_i\}) + W_s x \tag{3}$$

ResNet is a fully convolutional network (FCN) based on residual learning. It can learn the high-level features using a much deeper network than traditional FCNs like VGG nets [23] and Faster R-CNN [22]. ResNet has been widely approved thanks to a large number of successful applications based on it. Consequently, in this paper, I employ a ResNet for feature extraction. Compare to the algorithm-based feature extraction methods used in [9]. ResNet generally promise a better feature quality, which can help the classifier get a higher classification accuracy.

2.3 Casada

Casada algorithm is a cascade neural network with an adaptive learning rate for each weight. Casada is an improved version of Casper [4]. Casada algorithm employed an Adam optimizer for network training. Casada uses the same neural network architecture as Casper. After a new neuron is added, the learning rates of different weights are set to different values based on the positions of the weights. Basically, the weights that are close to the new neuron have a larger step size than other weights. The idea is similar to the Cascor algorithm. The old neurons remain basically the same and the new neuron reduces the network error, but no weight is frozen. The early neurons can still adjust themselves to get a better representation of features. Therefore, we don't need to use a lot of neurons to fix the error caused by poor feature extraction from early neurons. Another advantage of no frozen weight is that the neural network won't be forced to saturate, the edge between data points will be smoother than the Cascor algorithm.

2.4 Optimizer

RPROP Backpropagation (BP) is the most commonly used algorithm for neural network training. It is based on chain rule and uses the partial derivative of the weights to optimize the neural network. Generally speaking, the learning rate of BP, which scales the derivative, is one of the most important hyperparameters in the training process. If it is too large the neural network will suffer from oscillation. On the contrary, if too small, it will take too long to reach the minimum. In the original BP algorithm the learning rates are fixed before the training. Some previous works tried to introduce a momentum term to the traditional BP algorithm so that the learning rate can be adaptive. However, the Momentum-term itself become another very sensitive hyperparameter, which is a problem just like the learning rate. Later works tried to use dynamic learning rate based on the observation of loss function. But these works ignored a very important fact that the step size used for weight update is not only related to the learning rate but also the partial derivatives. Partial derivatives don't always reflect the true need for weight update. Hence, Riedmiller and Braun proposed resilient propagation (RPROP) algorithm [2] for weight update in BP. RPROP manages the step size of each weight separately based on the sign of the partial derivations. If the sign of a partial derivation of weight changes after a weight update, then the step size used in that update is too large. Hence, we need to reduce the step size. On the contrary, if the sign doesn't change then we can use a larger step size on this weight in the next update so that it can reach the optimization target faster. RPROP's step size update can be expressed by the following learning rule:

$$\triangle_{ij}^{(t)} = \begin{cases} \eta^{+} * \triangle_{ij}^{(t-1)}, & \frac{\partial E}{\partial \omega_{ij}}^{(t)} * \frac{\partial E}{\partial \omega_{ij}}^{(t-1)} > 0 \\ \eta^{-} * \triangle_{ij}^{(t-1)}, & \frac{\partial E}{\partial \omega_{ij}}^{(t)} * \frac{\partial E}{\partial \omega_{ij}}^{(t-1)} < 0 \\ \triangle^{(t-1)}, & else \end{cases} \tag{4}$$

After we get a proper step for weight update. We no longer need the value of the partial derivations. RPROP only takes the sign of partial derivations and use the equation: to update the weights where is:

$$\triangle\omega_{ij}^{(t)} = \begin{cases} -\triangle_{ij}^{(t)}, & \frac{\partial E}{\partial \omega_{ij}}^{(t)} > 0 \\ +\triangle_{ij}^{(t)}, & \frac{\partial E}{\partial \omega_{ij}}^{(t)} < 0 \\ 0, & else \end{cases} \tag{5}$$

$$\omega_{ij}^{(t+1)} = \omega_{ij}^{(t)} + \triangle\omega_{ij}^{(t)} \tag{6}$$

Due to the multiplication during the step size update, the step size may explode or disappear. So RPROP set an upper and lower limit for the weights to make sure the system won't break down due to this issue. The experiments have proved that RPROP converges faster and has higher accuracy than traditional methods. The original Casper algorithm used a modified RRPOP algorithm for weight update.

Adam The original Casper algorithm is proposed in 1997. Better optimizers are introduced in more recent years. Adam [6] is a widely used algorithm proposed in 2015 which only requires first-order gradients. It combined the advantages of AdaGrad [7] and RMSProp[8]. Adam computes individual learning rates for different weights, which works well in many previous works. The step size of Adam is basically invariant to the scale of gradient, which is bounded by the hyperparameters. Many successful applications have proved that Adam is a simple and computationally efficient algorithm. The implement of Adam is very straightforward and not so memory-hungry. Overall, Adam is a robust and high efficient optimizer for non-convex optimization problems in machine learning. So, in this assignment, A modified Adam optimizer is used in Casada. When a new neuron is added to the neural network, the momentum will be reset to zero, and continue to train the neural network. The modified Adam algorithm can be expressed as:

Require: $\alpha_{l_1}, \alpha_{l_2}, \alpha_{l_3}$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
Require: N: Maximum neuron number
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
$n \leftarrow 1$ (Add the neuron number)

$\theta_0 \leftarrow \theta_{1,0}$ (update the parameter vector)
while $n \leq N$ **do**:
  $t \leftarrow t + 1$
  $g_t \leftarrow \bigtriangledown_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestept)
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
  if θ_t converged **do**:
    $n \leftarrow n + 1$ (Add a new neuron)
    $f(\theta) \leftarrow f'(\theta)$ (Update the Stochastic objective function)
    $\theta_t \leftarrow \theta_{n,t}$ (update the parameter vector)
    $m_0 \leftarrow 0$ (Set the 1st moment vector to 0)
    $v_0 \leftarrow 0$ (Set the 2nd moment vector to 0)
  end if
end while
return θ_t(Resulting parameters)

3 Experiment and Discussion

3.1 Dataset

The proposed model is trained and tested on a stress recognition dataset, ANUStressDB. This dataset involves 31 subjects each subject has one RGB video and one thermal video each containing 20 clips. Subjects are stressed in half of the clips and calm in the other half of the clips. Videos are shot at 640 * 480 @ 30 FPS. In this paper, I use the data from 4 subjects. Due to the lack of RGB videos, I have to use only the thermal videos for the experiment. The most valuable part of the thermal videos is the face region. In order to avoid the distraction of other parts, I decided to only use the face region for this task. However, current face recognition algorithms are basically designed and trained on RGB images, which can't provide a satisfying result on thermal images. So, I have to manually label the faces in the thermal images. That's why we choose four of the subjects. Fortunately, in some videos, subject's head roughly stay in the same position. The faces are resized to 128*128 for training and test. The dataset is randomly divided into 11 parts for training, testing, and validation. Cross-validation is applied on 10 parts of data for training and testing to make the most of the available data for training and evaluation. In order to compare with the previous work [9] on the same dataset. In each round, I take 6 parts for training and 4 parts for validation. The data set is basically balanced as figure 3 shows. So, no data balancing method is needed. 'Stressful' and 'Calm' are coded using binary numbers, 0 for 'Calm' and 1 for Stressful. The dataset is batch-normalized during the training. For evaluation, the training and validation sets are used to train and select the model that generalizes best to unseen data. Then I run the best model on the test set to evaluate the real performance on unseen data.

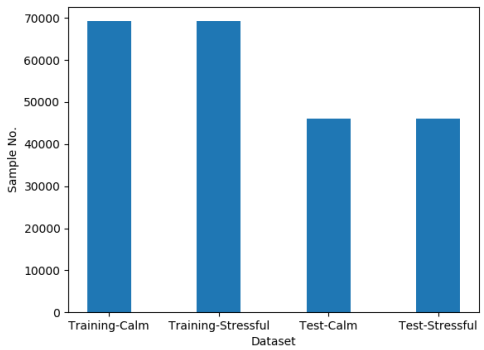

Fig. 3. The distribution of the samples

3.2 Implement details

The proposed architecture is implemented using Python with NumPy, pandas (Python Data Analysis Library), and PyTorch. I hire a ResNet 101 for feature extraction and representation learning. The images are per-pixel mean subtracted [18]. And I also used the standard color augmentation in [18]. Each convolution is followed by a batch normalization operation before the activation function. The model is pre-trained on ImageNet [19]. Casada is used for classification. In order to compare with the Casper algorithm [4], the hyperparameters of the Casada model is set to the same value used in Casper, $L_1 = 0.2$, $L_2 = 0.005$ and $L_3 = 0.001$. The hyperparameters used for RPROP are set to: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\delta_{max} = 50$, $\delta_{min} = 1 \times 10^{-6}$. The hyperparameters used in Adam are $\alpha = 0.0002$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$. I first implemented the Casper algorithm using python. However, the RPROP doesn't work so well no. I tried mini-batch, batch, and SGD. None of the methods make the RPROP algorithm works. Considering the workload of implementing the entire network architecture using python, I gave up and implemented the model using PyTorch. The model details are showed in Figure 4

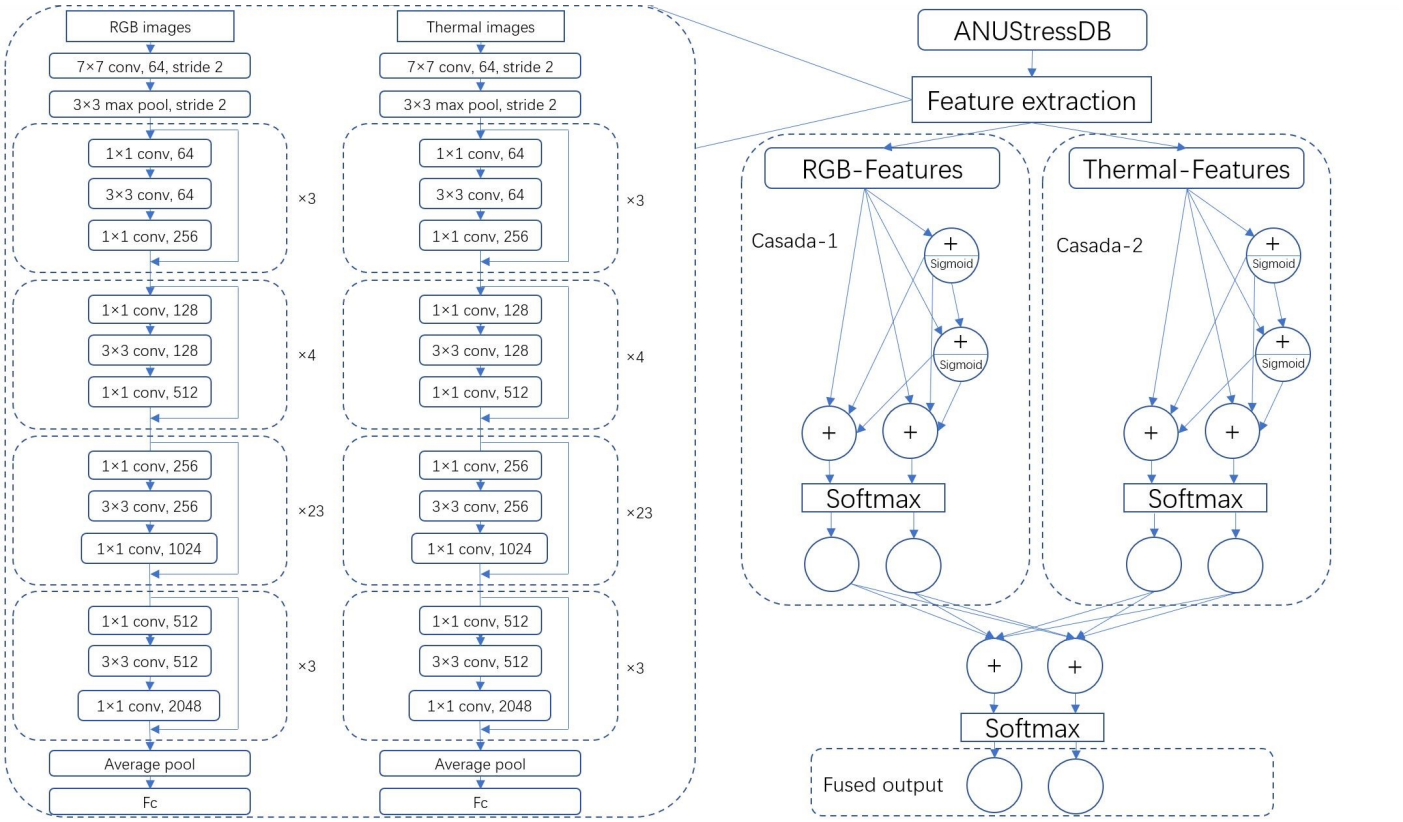

Fig. 4. The implement detail of Fused Bi-Cascade architecture

Loss function I used the cross-entropy loss to measure the prediction of RGB-Casada, thermal-Casada, and the multilayers perceptron. The scaled losses are added together. The scalers are set to 0.5, 1, 0.5 respectively. The loss function can be expressed as follow:

$$L_{total} = \lambda_{RGB} * L_{RGB} + \lambda_{thermal} * L_{thermal} + \lambda_{MLP} * L_{MLP}$$

$$L_{RGB} = \sum CrossEntropyLoss\left(\widehat{p}_{RGB}, target\right)$$

$$L_{thermal} = \sum CrossEntropyLoss\left(\widehat{p}_{thermal}, target\right)$$

$$L_{MLP} = \sum CrossEntropyLoss\left(\widehat{p}_{MLP}, target\right)$$

Early stopping When training a neural network, there may be a point during the training when the model will stop generalizing and begin to overfitting. The training loss may continue to decrease but the validation loss begins to increase which indicates that the model becomes less useful for predicting new data. Thus, we need to avoid

overfitting. Early stopping is a common method that can effectively prevent overfitting. During the training, I use the validation set to validate the model every epoch. When the best validation accuracy is achieved, the accuracy and the model will be saved. In a 10-fold cross-validation. Every round the code run, the best model will be stored. After all 10 rounds, The best model will make a prediction on the test set to check the generality of the model.

Transfer learning In order to train the model faster and get better accuracy. I used a ResNet pre-trained on ImageNet for transfer learning. Transfer learning generally has a positive effect on relevant tasks. Even for very different tasks. The low-level features extraction layer can still help accelerate the training. ImageNet is a concept recognition database that aimed to classify different items. However, this work focuses on stress recognition. Consequently, the high layers in ResNet need to be fine-tuned for stress recognition. The pre-trained model wasn't frozen during the training for fine-tuning. During the experiment, this transfer learning between ImageNet and ANUStressDB is proofed to be successful.

3.3 Results

Transfer learning and fine-tuning I first evaluated ResNet with and without pre-trained weights. During the training, the ResNet without pre-trained weights has higher training loss throughout the entire training procedure (figure 5) even though two ResNets use the same network structure. However, if the pre-trained weight is directly used for classification without fine-tuning, the model had a even higher training loss.

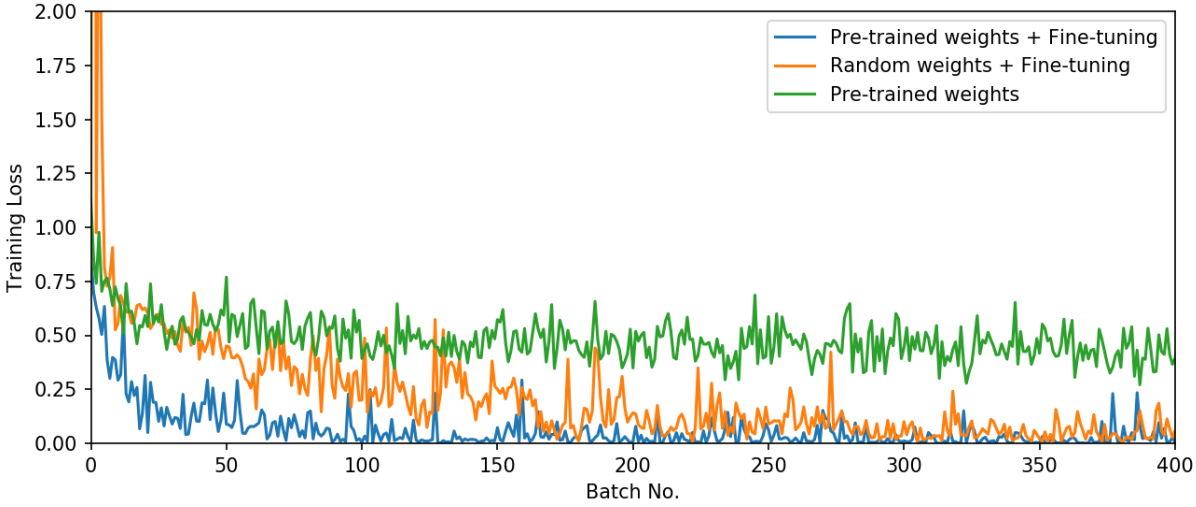

Fig. 5. The training loss of different training methods

The training accuracy tells the same story (figure 6(a)). The model with pre-trained weights and fine-tuning performed best. The model initialed with random weights end up with a pretty good training accuracy but still lower than the previous one. The model with frozen ResNet weights achieved significate lower training accuracy than other models, which indicates that the ImageNet and the ANUStressDB are pretty different. I argue that transfer learning requires fine-tuning to get good accuracy. When we look at the test accuracy (figure 6(b)), we notice that the test accuracy of the pre-trained model without fine-tuning is unstable (figure 6(b)). It takes a much longer time to achieve an optimal result. During the training, the test accuracy dropped significantly after some iterations while the training accuracy didn't change much. This may indicate that the pre-trained weights can help the model generalize better to unseen data.

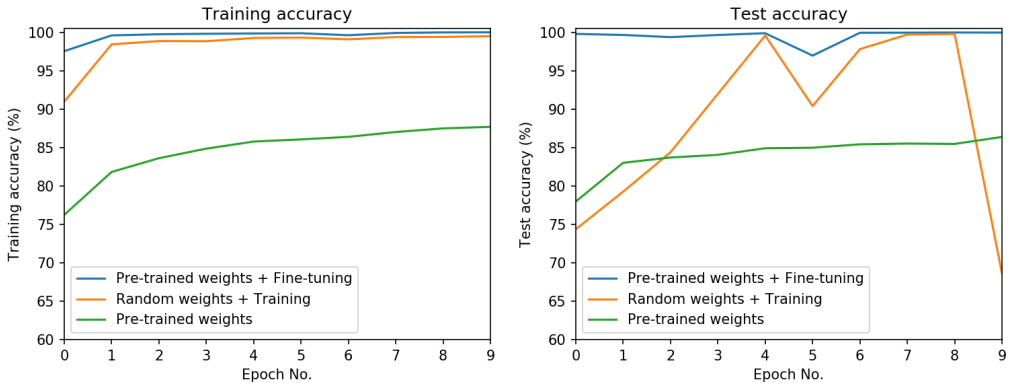

Fig. 6. (b) Training accuracy, **(b)** Test accuracy

Table 1. Best accuracy achieved by different training methods

Method	Training accuracy (%)	Test accuracy (%)
Pre-trained weights + Fine-tuning	99.99	99.98
Random weights + Training	99.50	99.78
Pre-trained weights (Frozen)	87.69	86.38

Casada vs. Casper A previous work [20] by Yu C. has analyzed the Casada architecture's performance. He claimed that Casada converges faster than Casper and needs fewer neurons to get better accuracy than Casper. But the dataset used in that work is small. The result isn't convincing. In this paper, I will further evaluate the performance of Casper and Casada architecture. Figure 7 and table 2 shows the training and test accuracy of Casper and Casada. Casada exhibits considerably higher accuracy, and it also converges faster.

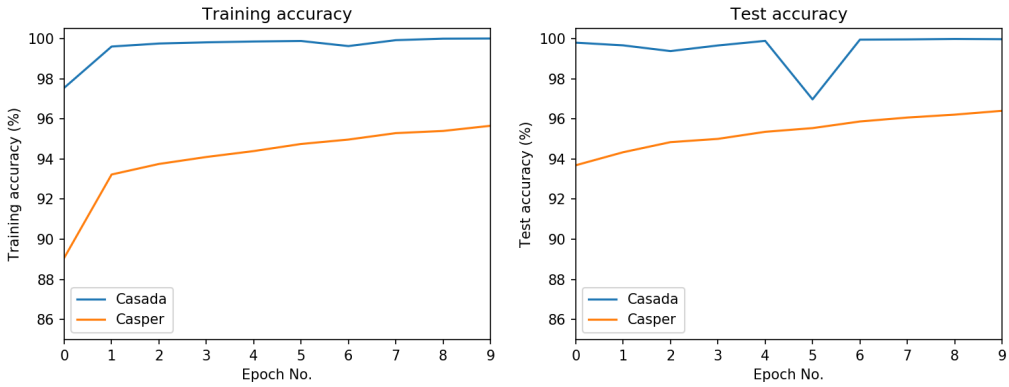

Fig. 7. (a):Training accuracy, **(b)** Test accuracy

The proposed method vs. SVM-based method Previous work used the SVM-based method on ANUStressDB. In this section, I will compare the proposed method with the SVM-based method. During the evaluation, I used the same evaluation method as [9]. As I have mentioned in the dataset section. In order to make the results more reliable, I also used 10-fold cross-validation. Unfortunately, I only get the thermal videos for evaluation. So, only the thermal part of the Bi-cascade model was used. The proposed method is better than the SVM-based method (by 11%) with only the thermal sensor. Table 3 shows the accuracy comparison.

4 Conclusion and Future Work

In this paper, I proposed the Fused Bi-Cascade architecture for stress recognition. Compare to stress recognition methods with superpixel-based feature extraction, Deep learning based Fused Bi-Cascade architecture achieved higher accuracy and faster speed. And I also improved the traditional cascade neural networks. Casada is more

Table 2. The accuracies of Casada and Casper[4]

Method	Training accuracy (%)	Test accuracy (%)
Casada	99.99	99.98
Casper	95.65	96.40

Table 3. The accuracies of different methods

Method	Accuracy (%)
SVM-based (RGB) [9]	62
SVM-based (Thermal) [9]	86
SVM-based (RGB + Thermal) [9]	89
Proposed Method (Thermal)	99.98

accurate than traditional cascade models. Furthermore, it converges faster and needs fewer neurons to get the best accuracy. The proposed system provides a contactless way to recognize the stress, which may promote public health. During this work, only the thermal data is available. So, only part of the model can be tested. The result is pretty good. With only thermal data, the proposed method outperformed the previous work with RGB and thermal data. During the experiment, I noticed a rapid error increase after adding a new neuron to the cascade network. Based on K. He et al.'s theory, it is basically due to the poor ability of the network to fit an identity mapping [21]. Although, residual learning can solve this problem. I can't use it on cascade networks for the following reasons. First, cascade networks add one layer (neuron) at a time. However, adding the residual connection to one single layer can't help the learning [21]. Second, even if we add multiple neurons at a time, the outputs of each layer have different sizes. We have to add a linear projection before adding the residual connection. Unfortunately, when we add this linear projection, the model still needs to learn the projection weights as an identity mapping, which is just what we want to avoid. I designed a cascade model with residual learning (8). However, it can't help us to avoid the loss increases after adding a new neuron. In the future, I will seek a better way to deal with the impact.

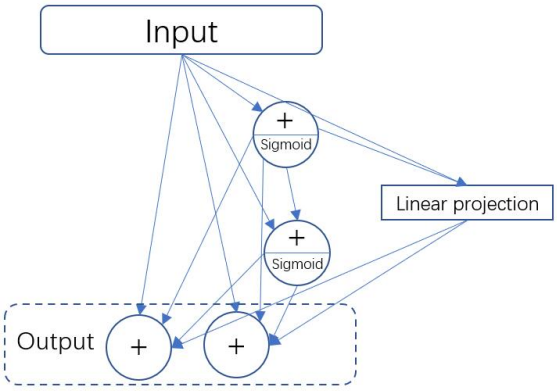

Fig. 8. Residual Cascade Network

References

1. Fahlman,S.E., Lebiere, C.: The Cascade-Correlation Learning Architecture. Advances in Neural Information Processing II, 524–532 (1990)
2. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. Ruspini, H., (Ed.) Proc. of the ICNN 93, San Francisco , 586–591 (1993)
3. Riedmiller, M.: Rprop - Description and Implementation Details. Technical Report, University of Karlsruhe (1994)
4. Treadgold N.K., Gedeon.T.D. : A Cascade Network Algorithm Employing Progressive RPROP.
5. Xu, Q. NAKAYAMA, K: Avoiding Weight-illgroth: Cascade Correlation Algorithm with Local Regularization. IEEE International Conference on Neural Networks , 1954–1959 (1997)
6. Kingma P.D..Ba,J.L: Adam: A Method for Stochastic Optimization. International Conference for Learning Representations, San Diego (ICLR) (2015)
7. Duchi, J., Hazan, E., and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization , 2121–2159 (2011)
8. Tieleman, T. and Hinton, G.: Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report, (2012).

9. Irani,R., Nasrollahi, K.„ Dhall, A„ Moeslund, T.B., Gedeon.T.D.: Thermal Super-Pixels for Bimodal Stress Recognition,
10. Kwok, T., and Yeung, D.: Experimental Analysis of Input Weight Freezing in Constructive Neural Networks. IEEE Int. Conf. NeuralNetworks. 511–516 (1993)
11. Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". Philosophical Magazine. 2 (11): 559–572.
12. Cohen et al. Applied Multiple Regression/Correlation Analysis for the Behavioural Sciences 3rd ed. (2003). Taylor & Francis Group.
13. B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation 10(5):1299-1319.
14. J. B. Tenenbaum, V. de Silva, J. C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science 290, (2000), 2319–2323.
15. K. He and J. Sun. Convolutional neural networks at constrained time cost. In CVPR, 2015.
16. R. K. Srivastava, K. Greff, and J. Schmid Huber. Highway networks arXiv:1505.00387, 2015.
17. K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In ICCV, 2015.
18. A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. InNIPS, 2012.
19. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. arXiv:1409.0575, 2014.
20. Y. Cui. Cascade Neuron Network Based Bimodal Stress Recognition. 4th ANU Bioinspired Conference
21. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. Paper presented at the 770-778.
22. S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS,2015.
23. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
24. A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks, `http://https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/`. Last accessed 7 December 2018
25. MedlinePlus Stress and your health, `https://medlineplus.gov/ency/article/003211.htm`. Last accessed 2 April 2021