Facial Expression Recognition in SFEW Database using Convolutional Neural Network with Casper

Jinhong Ni
Research School of Computer Science, Australian National University
u6742353@anu.edu.au

Abstract. Facial expression recognition (FER) problems have been addressed and studied in many recent researches. The SFEW database [1] contains face images that are close to real-world conditions, which adds additional complexity to the problem. Convolutional neural network (CNN) on the other hands, has been proven to be effective for features extraction in images classification problems. In this paper, we conduct an experiment with Casper algorithm, which has been introduced as an algorithm for training neural networks without explicit network architecture design [2], to learn the features extracted from convolutional layers. We also experiment to transfer some pre-trained convolutional layers to the Casper model, which is inspired by transfer learning. As the database is small, we also perform cross-validation to compute the average accuracy and extract the best trained model for all model architectures. The trained model achieved 43.18% Top-1 validation accuracy and 37.25% testing accuracy with relatively small network. It surpasses the SPI baseline [1]. The model is also compared with traditional CNN with fully connected layers, convolutional Casper model with frozen transferred weights, and CNN model trained from scratch. We find that CNN with Casper and pre-trained convolutional layers transferred seems to be effective for both learning features from a pre-trained CNN and driving learning of CNN.

Keywords: Deep learning, Casper, SFEW, Convolutional neural network, Transfer learning.

1 Introduction

Many recent studies have shown the effectiveness of machine learning and deep learning techniques for classification problems. However, most machine learning and deep learning methods often require humans to manually design the network architecture, as there are only a few effective ways to obtain the optimal network architecture without testing. Casper is introduced as a self-structuring neural network, which is improved based on Cascade Correlation (Cascor) with better generalization [2]. Casper algorithm could be a potential solution to the issue as the network will structure itself as training progresses.

The SFEW database contains facial images in unconstrained conditions and introduces additional complexity to the problem [1]. Most of previous methods have shown high performance on facial expression databases especially under lab-controlled environment, but are not suitable for facial expression recognition (FER) in uncontrolled environment [1]. Convolutional neural network (CNN) has shown better performance comparing to other methods in images classification problems like facial expression recognition (FER) [4]. In this paper, in order to test the effectiveness of Casper in learning features extracted from CNN, we will train a CNN model connected with Casper network to encounter the FER problem in the SFEW database.

2 Method

2.1 Face Detection and Data Preprocessing

The SFEW database is extracted from movie, which covers unconstrained facial expressions, varied head poses, large age range, different face resolutions, occlusions, varied focus and close to real world illumination [1]. The database contains 675 images with resolutions of 720x576, in 7 classes of expressions, namely angry, disgust, fear, happy, neutral, sad, and surprise. Fig.1 shows some sample images from the database.

In addition to the faces, these images also contain complicated background, which adds unnecessary complexity to the problem. Generally, the only relevant information needed for FER is the faces. Also, the rotations of faces might be different in the unconstrained environment. To extract the properly aligned faces from the original images, we need to apply some face detection and alignment methods.

Multitask Cascaded Convolutional Network (MTCNN) is a deep learning approach to the face detection and alignment, and it is the state-of-the-art technique with high precision while keeping nearly real-time performance [3]. As shown in Table.1, the MTCNN is fairly robust since it detects most faces from the database with high precision (No false faces).

Fig. 1. Sample images from the SFEW database. [1]

Table 1. Number of true faces detected in SFEW database by MTCNN.

Expression Classes	Number of true detected faces by MTCNN	Total number of facial images in the SFEW database	Percentage of the detected face
Angry	74	100	74.0%
Disgust	62	75	82.7%
Fear	74	100	74.0%
Happy	87	100	87.0%
Neutral	79	100	79.0%
Sad	74	100	74.0%
Surprise	82	100	82.0%

In our methods, we first perform MTCNN face detection and alignment to get the cropped and aligned faces. As the color information and high resolution are irrelevant for FER, we convert the detected faces to gray scale and resize them to 48x48. Then, all faces in each expression class are randomly shuffled and divided into training and testing sets with 8:2 ratio respectively. The process could be also visualized as Fig.2.

Fig. 2. Database Preprocessing.

As the database is small, the results are likely to be unstable. We perform k-fold Cross Validation to find the suboptimal model and for more robust testing result, with k set as 10. This means the training set will be further separately into training and validation sets with 9:1 ratio respectively. The model with the highest validation accuracy is used for testing.

2.2 Casper

Casper is an algorithm which determine the network architecture itself, while it also generalizes better especially in classification problems than Cascor [2]. Casper starts with one hidden neuron, which is fully connected to the input

neurons and output neurons, as shown in Fig.3. Each time a neuron is added, it is connected to all the previous hidden neurons, input neurons, and output neurons. Fig.4 shows the network topology after one neuron is added to the initial network. The network is trained with a modified version of RPROP algorithm each time an additional neuron is added. RPROP is modified in the way such that different initial learning rates will be set based on the regions that weights belong to.[2] The weights are divided into three regions: weights connected between the newest neuron and previous and input neurons, weights connected between the newest neuron and output neurons, and all other weights [2], each region has its own learning rate L1, L2, and L3 respectively. As L1 corresponds to weights of the newest added neuron which are untrained, it is set significantly higher than L2 and L3 so that the new neuron could learn and optimize according to error quickly. For similar reason, we set L2 larger than L3 but without too much interference from other weights [2]. L3 is set to a small value but not zero to avoid poor generalization issue like Cascor.

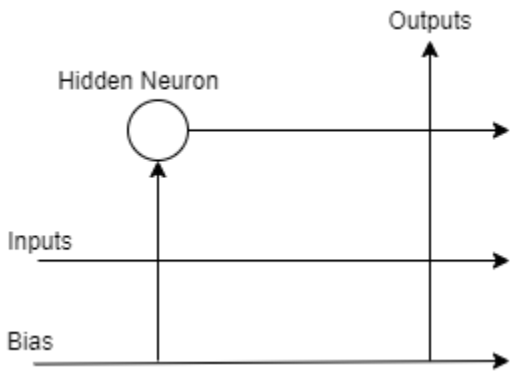

Fig. 3. Initial Network Topology.

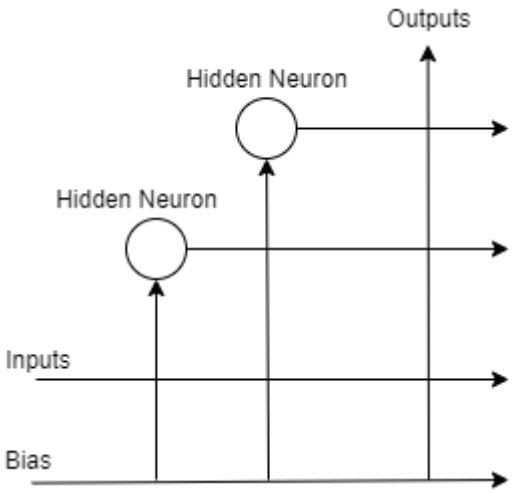

Fig. 4. Network Topology after one neuron is added.

The measurement of the error is taken for each time period computed as $15 + P * N$, where P is a hyperparameter and N is the number of currently installed neurons [2]. For each time period, the error needs to decrease by at least 1% comparing the previous time period, otherwise the network would stop training.

2.3 Convolutional Neural Network

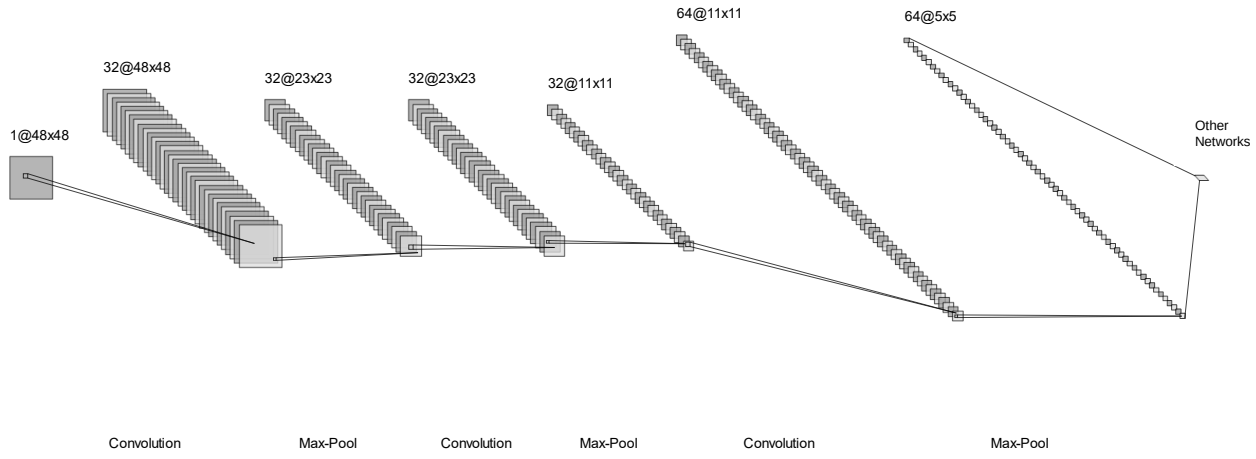

Fig. 5. CNN Architecture.

The architecture of the CNN for feature extraction is shown as Fig.5. It contains three convolutional layers, each followed by a max-pooling layer. Batch normalization is performed at each convolutional layer as well. Convolutional layers extract features from the previous layer by applying filters. Pooling layers are used to reduce the dimensions of the features extracted from convolutional layers. The output of the last layer is then fed into other networks such as Casper or fully connected neural network to learn the features. More details about the parameters set for the CNN are specified in Table.2.

The CNN is pre-trained with the last layer only connected to a linear layer, which directly maps the features to output. More of this will be discuss in 2.4 Transfer learning section below. These convolutional and pooling layers are also connected with Casper neural network and fully connected network for experiment.

Table 2. Parameters set for CNN.

Layer	Filter size	Stride	Padding	Output	Batchnorm
Input				1@48x48	
Convolution	5x5	1	2	32@48x48	0.5
Max-Pool	3x3	2	2	32@23x23	
Convolution	5x5	1	2	32@23x23	0.5
Max-Pool	3x3	2	2	32@11x11	
Convolution	5x5	1	2	64@11x11	0.5
Max-Pool	3x3	2	2	64@5x5	

2.4 Transfer learning

As Casper algorithm has not been proven to be effective in driving the learning of CNN, the performance of CNN connected with Casper that is trained from scratch might not be guaranteed. Inspired by transfer learning, what we could do is pre-training a CNN model, then transfer the trained convolutional weights to the new network instead of training the entire network from scratch. Transferred weights are usually fine-tuned with a small learning rate, we set the learning rate as 0.01 for transferred weights. To experiment, we are also going to freeze the transferred weights to see how the results differ. These models with transferred layers will also be compared with CNN connected with Casper that is built from scratch.

To pre-train the CNN, we follow the CNN architecture demonstrated in Section 2.3 above, then map the output of the convolutional layers directly to outputs. As there is no hidden layer involved, we could treat this linear CNN model as a baseline for comparison with other models with different network architecture connected to the pre-trained CNN.

3 Results and Discussion

In this section, we train the models individually, analyze and compare them by measuring average validation accuracy, top-1 validation accuracy, and testing accuracy. The model is also compared for its precision, recall and specificity results with the baseline based on the SPI protocol [1].

3.1 CNN with linear

We first train a CNN connected with a linear layer as our pre-trained CNN for transfer learning. Learning rate is set as 0.01 and ReLU is used as the activation function for convolutional layers. Cross Entropy Loss and Adam Optimizer are used for training. The model achieved 33.03% average validation accuracy, 38.64% top-1 validation accuracy, and 28.43% testing accuracy.

3.2 Transferred CNN with Casper

We transfer the pre-trained weights in the CNN trained in Section 3.1 to the CNN with Casper model with fine-tuning. This model is then trained with P of 5. Learning rates L1, L2, and L3 are set as 0.2, 0,005, and 0,001 respectively. ReLU activation functions are used for the Casper network. The learning rates for transferred weights are set to 0.001. Cross Entropy Loss and RPROP optimizer are used for training. The model achieved 35.33% average validation accuracy, 43.18% top-1 validation accuracy, and 37.25% testing accuracy; all of these surpass the pre-trained CNN performance. Testing confusion matrix of the model is shown in Fig.6.

3.3 Transferred Frozen CNN with Casper

We transfer the pre-trained weights in the CNN trained in Section 3.1 to the CNN with Casper model without fine-tuning, meaning transferred weights will be frozen. The same parameters as the previous transferred CNN with Casper are set for the model. The model achieved 38.31 % average validation accuracy, 48.84% top-1 validation accuracy, and 35.29% testing accuracy. Testing confusion matrix of the model is shown in Fig.6.

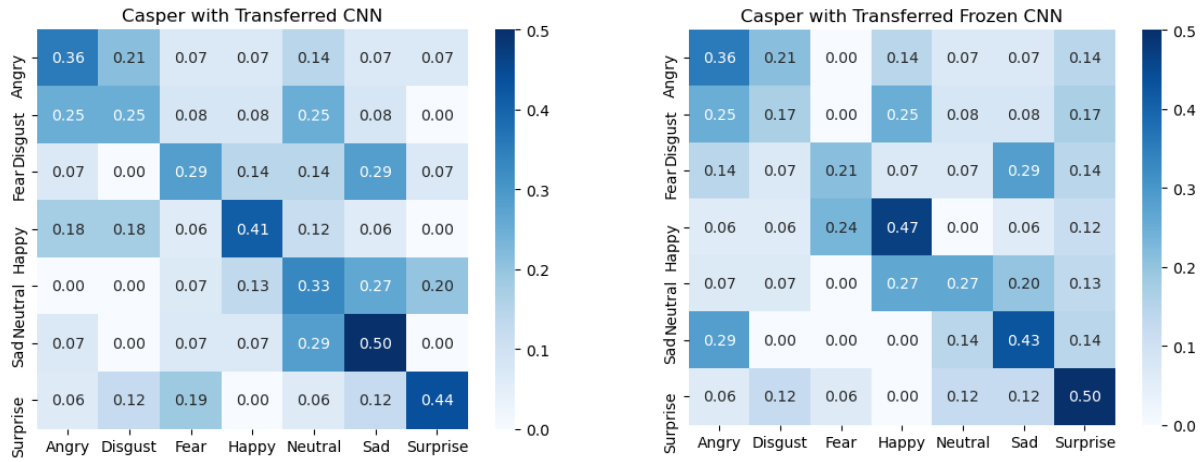

Fig. 6. Testing Confusion Matrices for Transferred CNN with Casper and Transferred Frozen CNN with Casper.

3.4 CNN with Casper

To experiment the effectiveness of Casper in driving learning of CNN further, we train a CNN with Casper from scratch. The same parameters as the previous transferred CNN with Casper and transferred frozen CNN with Casper are set for the model. The model achieved 36.87% average validation accuracy, 57.14% top-1 validation accuracy, and 38.24% testing accuracy. Testing confusion matrix of the model is shown in Fig.7.

3.5 CNN with Fully Connected

Finally, we train a CNN with fully connected layers from scratch for comparison with the Casper CNN models. The fully connected layers contain two hidden layers, with 2048 and 1024 hidden neurons respectively. Dropouts and batch normalization are applied for each hidden layer. ReLU is used as the activation function. Cross Entropy Loss and Adam Optimizer are used for training. The model achieved 34.81% average validation accuracy, 47.62% top-1 validation accuracy, and 36.27% testing accuracy. Testing confusion matrix of the model is shown in Fig.7.

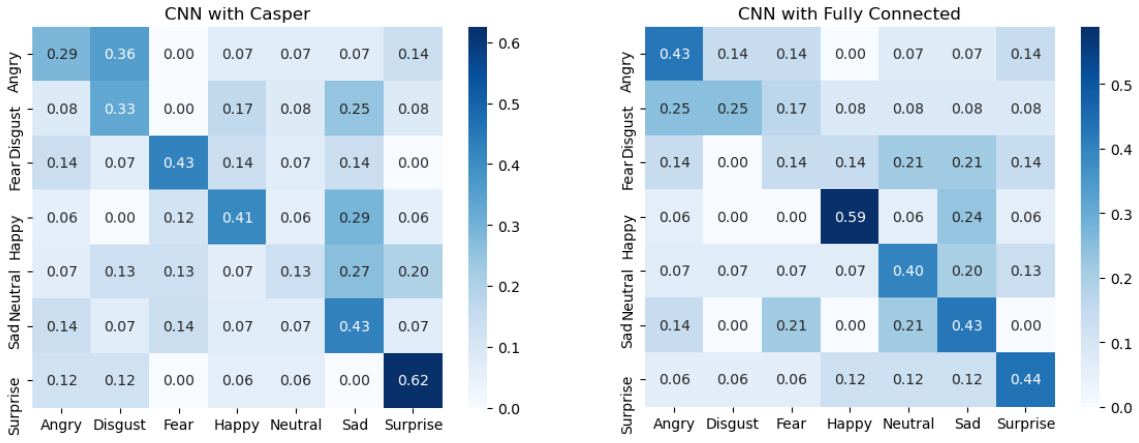

Fig. 6. Testing Confusion Matrices for CNN with Casper and CNN with Fully Connected.

3.6 Comparison between models

Comparisons are made across different models for the average validation accuracy, top-1 validation accuracy and testing accuracy. The accuracy measurements for all models are shown in Table.3.

Table 3. Accuracy measurements for all models

Models	Average validation accuracy	Top-1 validation accuracy	Testing accuracy
CNN + Linear	33.03%	38.64%	28.43%
Transferred CNN + Casper	35.33%	43.18%	37.25%
Transferred Frozen CNN + Casper	38.31%	48.84%	35.29%
CNN + Casper	36.87%	57.14%	38.24%
CNN + Fully Connected	34.81%	47.62%	36.27%

All models have surpassed the CNN with linear baseline. We could see that average validation accuracy of transferred frozen CNN and CNN with Casper are higher compared to the others. However, as the top-1 validation accuracy is relatively high for transferred frozen CNN with Casper and CNN with Casper, which means the results vary a lot. Also, the validation set is very limited in terms of the size, meaning any minor change would change the validation accuracy drastically. Therefore, validation accuracy is not a robust measurement to determine the performance. For testing accuracy, we see the transferred CNN with Casper has an accuracy of 37.25%, which is about 2% higher than the transferred frozen CNN with Casper. This is expected as fine-tuning generally provides better results. Also, we could see CNN with Casper that is trained from scratch has the highest testing accuracy out of five models. Therefore, we could conclude that Casper is effective in driving learning of CNN, but the performance is a bit unstable as the top-1 validation accuracy is much higher than the average validation accuracy. Also, by comparing the CNN with fully connected with the three CNN with Casper models, we could not identify any significant difference between the two. However, the network architecture for Casper is much simpler than the fully connected network. This could be also proved by the size of the model files. The size of the CNN with Casper models is around 400 KB, while the size of the CNN with fully connected is about 20 MB. In general, we have shown that Casper is effective for both learning features extracted and driving learning of CNN.

3.7 SPI Protocol Score for Benchmarking

We compute the scores for both the transferred CNN with Casper model and the CNN with Casper model based on the SPI protocol. The classwise scores are shown in Table.4 and Table.5 for the two models respectively. These results show a significant improvement from the SPI baseline in the SFEW paper [1]. The precision, recall and specificity are computed as follows:

$$\text{Classwise Precision} = \frac{tp}{tp+fp} \tag{1}$$

$$\text{Classwise Recall} = \frac{tp}{tp+fn} \tag{2}$$

$$\text{Classwise Specificity} = \frac{tn}{tn+fp} \tag{3}$$

Table 4. Classwise Precision, Recall and Specificity results for Transferred CNN with Casper model.

Emotion	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.36	0.27	0.33	0.50	0.26	0.35	0.58
Recall	0.36	0.25	0.29	0.41	0.33	0.50	0.44
Specificity	0.90	0.91	0.91	0.92	0.84	0.85	0.94

Table 5. Classwise Precision, Recall and Specificity results for CNN with Casper model.

Emotion	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.31	0.27	0.50	0.47	0.25	0.29	0.56
Recall	0.29	0.33	0.43	0.41	0.13	0.43	0.63
Specificity	0.90	0.88	0.93	0.91	0.93	0.83	0.91

4 Conclusion and Future Work

The paper proposed a CNN connected with Casper network architecture for seven expression classification problem in SFEW dataset. We also experiment the model with transferred pre-trained weights. We have shown the effectiveness of

Casper for both learning from extracted features and driving learning of CNN, while it also keeps the model relatively small and simple.

However, the distinction between transferred CNN and transferred frozen CNN is not clear. As discussed by Y. Clune, et al, the performance of fine-tuning is usually significant for more transferred layers network [5]. This explains why there is no clear improvement with fine-tuning, as our transferred model contains only three layers. The effectiveness of transfer learning is also not significant, again mainly due to the simplicity of the transferred layers. Future works could be done by transferring weights from more complex networks such as VGGNet and ResNet, to explore the effectiveness of transfer learning. As Casper is a network self-structuring algorithm, the similar idea could be extended to convolutional layers to train a fully self-structuring CNN in the future.

References

1. Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011, November). Static facial expressions in tough conditions: Data, evaluation protocol and benchmark. In 1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011.
2. Treadgold, N. K., & Gedeon, T. D. (1997, June). A cascade network algorithm employing progressive RPROP. In *International Work-Conference on Artificial Neural Networks* (pp. 733-742). Springer, Berlin, Heidelberg.
3. Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499-1503.
4. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6). Ieee.
5. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. *arXiv preprint arXiv:1411.1792*.