One-shot Architecture Search with Cascade Network for Facial Emotion Recognition

Chunyi Sun
Research School of Computer Science, Australian National University
u6342253@anu.edu.au

Abstract

The facial expression recognition task has been investigated for many years. However, classify the facial expression 'in the wild' still a tough task among image classification tasks. Most previous research on it tends to design a better network architecture that more suitable for this specific task. This paper will introduce a high efficiencies algorithm that could automatically decide the network architecture during the training stage. It is based on the cascade network and using the gradient-based architecture search method to design the network architecture. The automatically designed network improves the average test accuracy on SFEW around 20% compared to standard CNN with random search. Also, the searched network could achieve a comparable result with previous researches on the SFEW dataset.

Keywords

Cascade Network; Network Architecture Search; multi-label learning; Facial expression recognition.

1 Introduction

SFEW [1] dataset is an 'in the wild' dataset, the classification is affected by many environmental factors such as the different background, light condition, rotation, and scales. This dataset only contains 673 images for training, validation, and also testing, thus human-designed neural networks may cost too much time to tune the hyper-parameters to deal with the variance-bias trade-off problem. In this paper, we will introduce an algorithm that could decide the cascade network architecture in one training pass and introduce a fine-tuned model with multi-label training to further improve the network performance.

Most previous works on network architecture search are using the evolution algorithm [24, 27] or reinforcement learning [25, 26], which are based on vast computational resources and often prohibitively expensive to be widely used. Our method tends to reduce the time of the network architecture search process in three dimensions: Limit the search space, accelerate the search using the gradient-based search strategy and reduce the cost of the performance estimation step. Because introduce too much human knowledge to limit the search space will lead the search to be human-biased [22], in this work, we will limit the search space by using the property of cascade network and break up the optimize process properly. The gradient-based search method [21] tends to make the search space been differentiable, thus it allows the gradient descent been used to optimize the network architecture which could significantly accelerate the search process, our method will get advantage from this search method. Previous methods using evolutional algorithms often measure the fitness of candidate network architecture by evaluating the validation accuracy after it has been fully trained, however, only estimate the final performance of the network could be biased and time-consuming. In our method, we will define a weight for each candidate sub-networks and optimize them use gradient descent, then estimate the sub-net performance by those weights during training. We consider this parameter will be able to learn the long-term performance of each sub-net and able to implicitly learn its learning performance.

Different from standard network architecture, the cascade type neural networks able to build network architecture during the training state. The constructive cascade network has already introduced for many decades, in the original paper [2], it has proof that this network has great generalization performances and network construction properties. Many modern networks modified based on the cascade structure such as Cascade RNN [4] and Cascade CNN for face detection [5] are achieve remarkable results. Also, using prior knowledge is a useful way to reduce overfitting and improve the generalization ability of the networks. The V-A space is widely used for facial emotion recognition tasks which is a two-dimensional space. The x-axis represents the valence of each emotion while the y-axis represents the arousal of each emotion. In this work, we will separate the label space by positive arousal and negative arousal as the second label to guide the facial emotion recognition on searched network architecture.

Based on the discussion above, the purpose of this work is to find an efficient way to optimize the network structure based on the cascade network and achieve the desired result on the facial recognition task. Our main idea is starting our training with a large network that contains all candidate sub-networks, each sub-network is a standard cascade network of the initial state. Then, we will optimize the weights of each sub-network using the validation loss and continuing drop the sub-network with the lowest soft-max score during training. The detailed process will be discussed after introducing some related works such as the original cascade network [2] and Darts [21] in the next section. Section 4 will mainly be used to discuss the underlying properties of this algorithm and show how this network could overcome the problems we discussed before. We will also compare our network with traditional human-designed CNN and other published results on the SFEW dataset in this section.

To summarize, our method significantly reduces the network architecture searching time and improves the network performance, the algorithm is mainly getting benefit from (1). finding a reasonable way to limit the search space; (2). using the gradient-based NAS method and (3). properties of the cascade network.

2 Related Work

2.1 LoCC

LoCC (local feature constructive Constructive Cascade Neural Network) [2] is a cascade network for image classification. It builds the network architecture during the learning process. Fig 1 shows the architecture of the LoCC, it starts from the initial network that only contains the input layer, hidden layer, and the output layer. The hidden layer and each cascade layer are fully connected to output layer.
Each architecture variant will continue training for 100 epochs. As the middle structure is shown in fig 1, the cascade layer will get input from the input layer and the hidden layer by convolution operation.
When the maximum epoch is reached, a new cascade layer will be added, the new cascade layer will get input from the input layer, hidden layer, and all previous cascade layers. The training is using RPROP, which is an adaptive learning algorithm that adapts the weight only use the sign of the gradient.

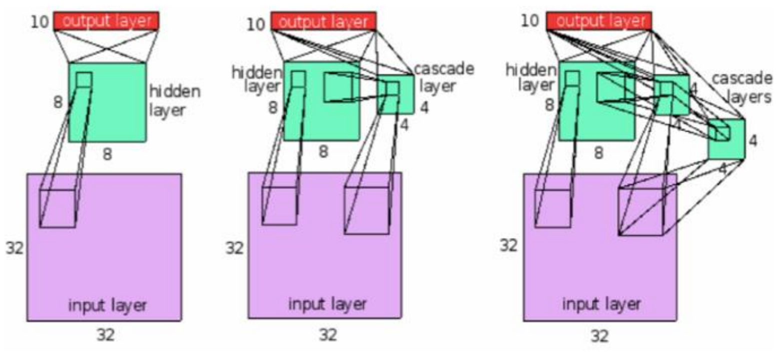


Fig 1 Architecture of LoCC

2.2 Darts (Differentiable Architecture Search)

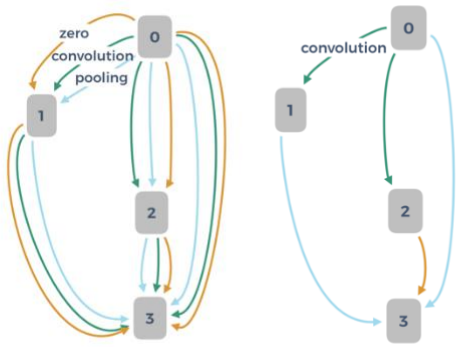


Fig2 Overview of Darts

The core idea of Darts is to search the network architecture in a differentiable way. Similar to most one-shot models, Darts is based on the relaxation of the representation of structures, but it is allowing the use of gradient descent to search architecture. In fig 2, each gray block in (a) is one cell which is a directed acyclic graph with N nodes, we need to connect those celss together by some operations such as pooling or convolution. To make the search space continues, we will give a weight α for each operation between two cells, each color represents one operation. The algorithm use validation loss with gradient descent to optimize weights for each operation while the weights within the operations will be optimize in normal way that using the training loss to optimise. For each operation between two nodes, calculate the SoftMax value for its corresponding weights α_{ij} over all weights on i_{th} edge. Then, choose operations for each edge based on the SoftMax value of α_{ij}.

2.3 Multi-label learning

Multi-label learning has been widely used in image classification and text categorization tasks. Multi-label learning can somewhat help the network to understand why an object belongs to a certain class than just make the prediction. In ‘binary relevance multi-label learning’ papers [6,7,8], they all point that a well-exploited correlation between labels could help achieve an excellent result. In most previous multi-label works [9,10,11], they all use another model to predict the groups of labels, which also require a large amount of data to make accurate prediction. In this paper, we will directly use the prior knowledge of the label space to build a group of labels.

3 Method

Some previous works on the one-shot architecture search start on a macro network graph and desired to find the best sub-graph. In our work, each sub-graph is an individual cascade network, before feeding into the macro-net, we do soft-max over it and initialize the sum of weights for sub-nets architectures to 1. Our training process could split into two sub-process: (1). construct a big network that contains all candidate sub-networks and deciding the network architecture during training. (2). fine-tune the decided architecture by adding a valence output and re-training. Thus, in the following sections, we will demonstrate our process with three parts: data pre-processing, network architecture search, and network fine-tuning.

3.1Data pre-processing

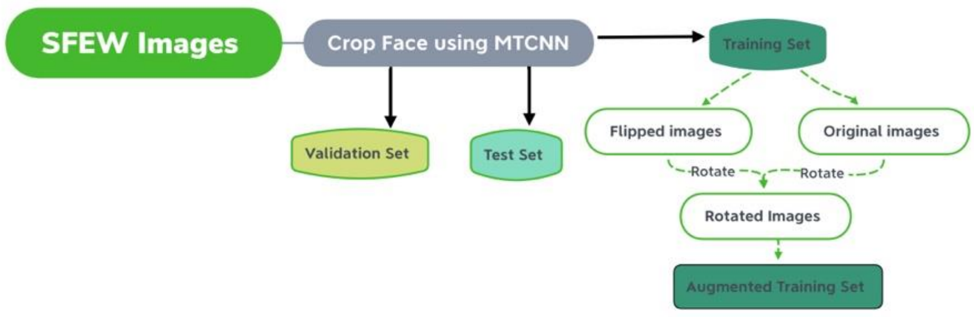


Fig 3 Data pre-processing flow-chart

Fig 3 shows the basic steps of the data pre-processing process. We first use MTCNN [21] to crop the face and resize the image to 256*256. Then, we split the dataset into the training set, validation set, and test set. There are 10% data for the validation set and 2 images for each class for the test set. Because the training data is very limited, we first flip all images in the training set, then rotate all original images and flipped images from -45 degree to 45, increase 15 degrees for each iteration. Then, collect all processed images to form the augmented training set.

3.2 Architecture overview

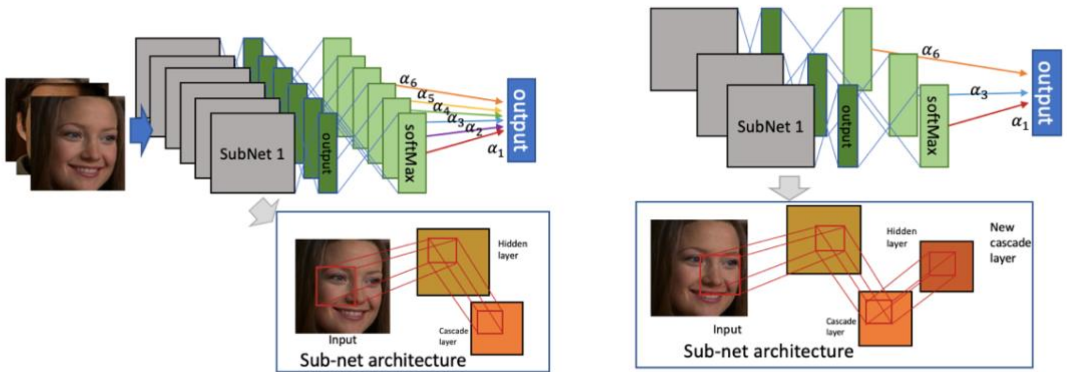


Fig 4 Architecture overview

The overview of the one-shot network architecture is shown in Fig4. The left-side is the initial state of the network, the network contains serval small sub-nets, all of them are the standard cascade network that we have introduced in section 2. The input will be feed into each subnetwork and we will perform SoftMax on their output. Finally, the SoftMax result of each sub-network will be parameterized by α before connecting them into the final output layer. Here, we have two sets of weights that need to be optimized, they are weights w for each sub-network and weights α which represents the contribution of each sub-network. During training, we will use the training loss to optimize the weights w as same as normal network training, then use the validation loss to optimize the weights α using the validation loss.
We will define a hyper-parameter call D, which controls when to drop the worst sub-network in the current revolution. As each sub-net are a cascade network, they are still allowed to add a new cascade layer during training. The right side of Fig 4 shows one possible state of the network during training, that some sub-networks have been dropped while kept sub-networks continue adding cascade layer.

3.3 Search the architecture

3.3.1 Build search space

This section will be used to describe how the search space been designed. The macro-architecture is combined with several cascade networks. Thus, to define the search space, the sub-network architectures need to be decided first. The search space grows exponentially as the optimized hyperparameter increase. Large search space will increase the searching time to be much larger than we excepted. However, if the search space is too small, then the algorithm is not able to do enough exploration to find good architecture. Our purpose finds a reasonable way to limit the search space which could balance the time consumption and the performance.

Cascade network is a suitable choice to be the sub-net because it is able to limit the search space in a reasonable way. For 3 layers CNN, it will have n^3 possible combinations for its architecture when we only consider optimizing the kernel size. For CNN, the order of the kernel size is important. For example, the algorithm chooses the first kernel size that too large for the current layer, then the layer after it will have no chance to recover the information loss. However, in the cascade network, due to the connection between the input layer and each cascade layer, the importance of order for the kernel size will be decreased. Also, the cascade network has constrained on the last stride size, as the previous section shows, the output from input to cascade layer and the output from the hidden to cascade layer should have the same size.

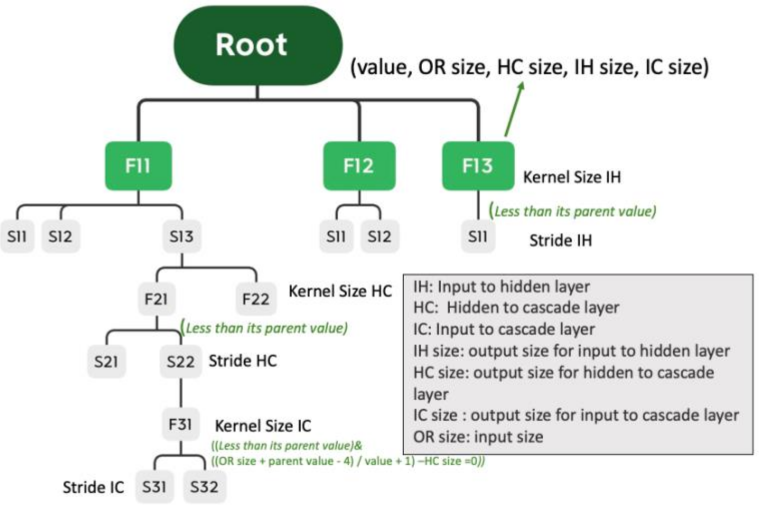

Root
(value, OR size, HC size, IH size, IC size)
F11
F12
F13
Kernel Size IH
S11
S12
S13
S11
S12
(Less than its parent value)
S11
Stride IH
F21
F22
Kernel Size HC
(Less than its parent value)
S21
S22
Stride HC
F31
Kernel Size IC
((Less than its parent value)&
((OR size + parent value - 4) / value + 1) –HC size =0))
Stride IC
S31
S32
IH: Input to hidden layer
HC: Hidden to cascade layer
IC: Input to cascade layer
IH size: output size for input to hidden layer
HC size: output size for hidden to cascade layer
IC size : output size for input to cascade layer
OR size: input size

Fig 5 Architecture search tree

We will build a search tree to search for all possible sub-net architectures. Fig 5 shows the architecture of the search tree. Each node in the search tree will contain the value for the current node, the original input size, the output size for input to the hidden layer, the output size for hidden to cascade layer, and the output size for input to the cascade layer. The output size of each layer in each node will be calculated and updated when we insert a new node. We will limit the kernel size only be 3,5,7, and 9, which are reasonable choices for the kernel size. The padding size is fixed to be kernel size minus 2. Also, the stride size should be limited to less than the kernel size for the current layer; and the stride size for 'input to cascade' layer should make the output size of the input to cascade layer and output size of hidden to cascade layer to be the same.

3.3.2 Training detail

For this task, the input size is 256, there are 15 possible candidate architectures for sub-net. Because in the fine-tuned process, there will be a second label represent the valence of the emotion. One fully connected layer between the valence output and the hidden layer will not enough to learn from a large output. Thus, we filter the architectures by limit the hidden layer output size less than 70. After filtering, the macro-net will contain 9 sub-nets.

The hyper-parameter D is used to decide when to dropping sub-network, when training iteration of the current architecture is larger than this number, we will drop the worst sub-net in this variant. The maximum iteration in the algorithm is the same as the one in LoCC [2], which controls when to add a new cascade layer. We would like to make the D less than the maximum iteration because we want to drop more sub-networks before the sub-networks become more complex. In this model, we choose the learning rate for the inner network optimizer to be 0.007 and the learning rate for the parameter optimizer to be 0.001; the maximum iteration is set to be 200 and the hyper-parameter D is set to be quarter of the maximum iteration.

Algorithm1: Training process for one-shot model

```
Start from the initial one-shot network with sub-nets $\mathcal{N} = \{N_1, N_2, \ldots N_9\}$.
Initialise the weights $\alpha$ for each sub-net to (1/len ($\mathcal{N}$))
**For** iteration in range(epoch):
    1. update w by descending $\nabla\mathcal{L}_{train}(w, \alpha)$
    2. update $\alpha$ by descending $\nabla\mathcal{L}_{validation}(w - \gamma\nabla\mathcal{L}_{train}(w, \alpha), \alpha)$
    3. if (iteration % (maximum iteration) == 0):
           add new cascade layer for each sub-net
    4. if ((iteration% D == 0) & (len ($\mathcal{N}$)>0))):
           remove the sub-net $N_i$ with lowest $\alpha$ from $\mathcal{N}$
           re-initialise the weights $\alpha$ for remaining sub-net to (1/ len ($\mathcal{N}$))
**end for**
Return the final architecture.
```

3.4 Fine-tune process

After decided the network architecture, fine-tune and re-training could help the decided network to further improve the performance. In the architecture searching process, it is using bi-level optimization [28], the training process is unstable. Re-training will only use the training loss that could make the training more stable. Instead of directly use the architecture got form the macro-net. We will add a valance output layer connect to the hidden layer, it used to classify the output class is on high valance or low valance. In V-A space, ‘Angry’, ‘Disgust’, ‘Fear’, ‘Sad’ are assigned as high valence emotion while ‘Happy’, ‘Neutral’, ‘Surprise’ are assigned as low valence emotions. To get the valence label, we encode high valance emotion to 0 and low valance emotion to 1. This valance output layer will connect with the hidden layer.

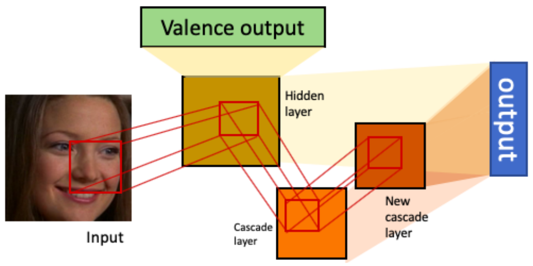


Fig 6 Fine-tuned Network Architecture

Algorithm2: Training process for fine-tuned network

```
Start from the initial network
**For** i in range (maximum epoch):
    1. calculate the final emotion prediction loss
    2. update weights by applying SGD with final emotion prediction loss
    3. if (i mod 5 ==0), do:
           calculate the valance prediction loss
    4. update weight by applying SGD with valance prediction loss.
**End for**
Save the current model and add new cascade layer, then continue the above training process.
Return the model with the best result.
```

We chose SGD for gradient descent and chose cross-entropy as loss function for valance prediction and final emotion prediction. The learning rate for the fine-tuned network is 0.007 and the maximum iteration for each variant is 500. In the real training process, the process could be terminated when the acceptable result is reached, or the testing accuracy has continued decreasing for a while.

4 Results and Discussion

4.1 Architecture evaluation

As discussed in the previous section, we made the hypothesis that our algorithm could reduce the network architecture searching time, and the performance evaluation method is able to determine the long-term performance of the sub-nets. In this section, we will construct some experiments based on our algorithm self.

4.1.1 Evaluate the performance of network architecture predictor

Compare to most works on the network architecture search, our algorithm does not select the architecture by directly evaluate the model with its validation accuracy. One reason is this process is time-consuming and another one is this estimation could be biased. Inspired by how humans search the network, it is not necessary to make decision after the network been fully trained. In practice, humans will analyse the training loss curve or the validation performance on the early stage to make decisions that continue training current architecture or tune the hyper-parameter.

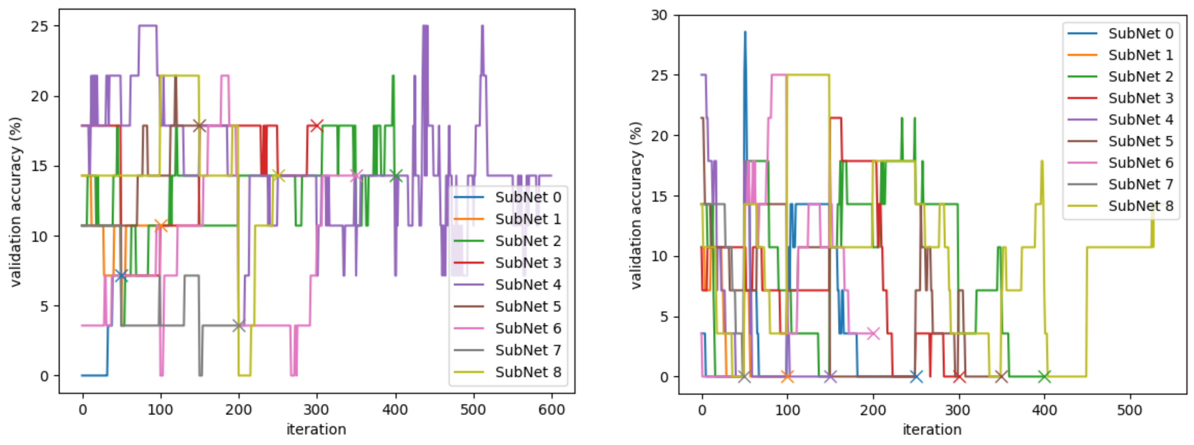


Fig 7 Plot of validation accuracy for each sub-net

The left side of Fig 7 plots the validation accuracy for each sub-set during the training process. While the cross dot means this sub-set has been drop at that correspond iteration. The validation accuracy for each sub-net is unstable and changing rapidly during training. This supports our previous hypothesis that the decision only based on the validation accuracy at a single iteration could be biased. From the overall trends of the prediction, the predictor evaluates the overall performance of each subnet during previous training iteration and the evaluation depends on current accuracy more than the previous accuracy.

Before the algorithm starts a new variant, the weights of each network will be re-scale to 1 divide the number of the sub-nets. This is because the weights of each subnet are very sensitive to the initial value, while the training iteration for each variant is not large enough to remove this effect. The right side of Fig 7 is the plot for the validation accuracy for each sub-net without re-scale the weights in training. The average performance of the sub-net is lower than the algorithm with re-scale the weights. Many sub-net validation accuracies go to zero before it been dropped. In the one-shot network architecture, before feeding into the macro-net, we perform SoftMax on the output of each sub-nets. This will push each sub-net output the prediction for emotion and make the sub-net validation accuracy be a meaningful criterion. Re-scale the weights on each architecture will make the weights on architecture works like voting, which makes the macro net prediction accuracy higher correlate to the sub-nets' prediction accuracy.

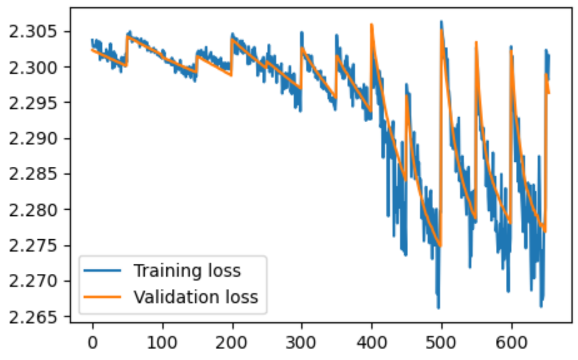
To evaluate the performance of the selection, we will measure the validation accuracy of all fine-tuned candidate sub-nets. In this experiment, we will run the one-shot model with rescaling and without rescaling, run each of them 5 times. Table 1 records the result of the validation accuracy for each net and counts the times that each sub-net been selected by the algorithm with re-scale the weights of architecture and without re-scale the weights of architecture.

Table 1 shows that the rescaled model makes more accurate prediction than non-rescaled model and the non-rescaled model always make different predictions for each experiment, it tends to drop the sub-net that been initialized with lower weights and reduce the effect of the validation loss. From table 1, the model with re-scaling selects the sub-net with the highest validation accuracy 4 times and select Net 8 once, while net 8 also achieves the second-highest validation accuracy among all candidate sub-nets. The model without the re-scale only selects Net 4 once and other choices achieve the validation accuracy only higher than the average validation accuracy.

Table 1. Fine-tuned accuracy for each candidate sub-net and the times been selected by the algorithms.

Nets	Net 0	Net 1	Net2	Net 3	Net4	Net5	Net 6	Net 7	Net 8
Times been selected (without re-scale)	0	1	1	0	1	0	2	0	0
Times been selected (with re-scale)	0	0	0	0	4	0	0	0	1
Fine-tuned validation accuracy (%)	35.71	42.86	42.86	39.28	50.00	39.28	42.96	39.28	46.42

4.1.2 Analyse the fine-tune process.



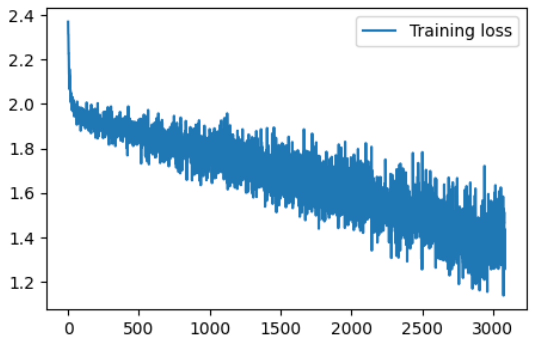


Fig 8 Training loss plot for one-shot model and fine-tuned model

The upper-level optimizer usually has complete knowledge of the lower-level problem, while the lower level predictor only could observe the decisions of the upper-level optimizer and then optimizes by its own strategies. This may lead to bi-level optimization problems involving uncertainties [28]. In our bi-level optimization, the number of weights for outer optimizer decreased as more sub-nets been dropped while the number of weights within the sub-net will continually increase as the cascade network grows during training. In our training, both sets of the parameters will be updated in each iteration and the learning rate for both optimizers do not been dynamically adjust during training. At the end of the training, the inner optimizer may have thousands of parameters that need to be optimized and the outer optimizer only have one parameter that needs to be optimized. Also, the weight is for the kept single sub-net, thus it is meaningless. As the left plot in figure 8, this is plotting the training loss and the validation loss during one-shot model training. The training loss is vibrating with the validation loss. Because After the best architecture has been selected, this architecture still needs to be optimized for more training iteration and the bi-level optimization becomes useless. This is why re-training the selected model is important. As the plot on the right side of figure 8, it is the training loss plot for the fine-tuned process, the training loss is well converged.

The next section will compare our fine-tuned model with standard CNN and other published results on the SFEW dataset to make a more comprehensive evaluation.

4.2 Comparison with other network architectures

In this section, we will first compare the results of our network with standard CNN on the SFEW dataset. In order to know how many improvements is caused by the cascade architecture and how many improvements is caused by our network architecture optimize strategy, we will directly train the cascade architecture with multilabel, that is, we did not do one-shot network search in this comparison. Both for our network and standard CNN, we will decide the network architecture for 5 times using random search and run each network architecture for 5 times. We will use the best result for each network architecture to calculate the average accuracy and does not counts the architecture with the worst result. We will compare the best and average performance on the validation set and report the test accuracy on the one that achieves the best performance on the validation set.

Then, we use our one-shot model to automatically decide the best network architecture and fine-tune the selected network architecture as we discussed in the 'Method' section. We will compare the fine-tuned network results with previous methods on the SFEW dataset.

Table 2. Comparisons with standard CNN.

Architecture	Validation accuracy %(average/best)	Test accuracy %
Cascade network with multi-label	32.57/42.85	35.71
CNN	28.57/35.71	28.57

In this experiment, we will use a random search to search the hyper-parameters for our cascade network and the hyper-parameter decided to be searched is the same as how we build the search space in the 'Method' section. The CNN used in this experiment has fixed pooling size 3, the activation function for the hidden layer is Relu and the activation function for the last layer is Tanh. The depth of the network is limited in range from 3 to 5. Other non-defined hyper-parameters will be decided using the random search.

Table 3. Comparisons with previous methods on SFEW dataset.

Architecture	Validation accuracy %(average/best)	Test accuracy %
Cascade network with multi-label(fine-tuned)	42.86/50.00	50.00
MvDA	22.70/-	-
E2E-FC	22.33/-	-
AUDN	26.14/-	-
STM-ExpLet [14]	31.73/-	-
Multiple Deep Net [12]	-/55.96	61.29
IPA2LT (LTNet) [23]	-/55.11	58.29

As Table 2 shows, all measured accuracy of cascade network is better than standard CNN. Compared to the standard CNN, there will be two reasons that be considered why cascade structure could improve the network performance.

1. Early stop and implicit pre-training

Early stop and pre-training are commonly used strategies to avoid overfitting in deep learning. In the cascade network, we define the maximum iteration for each architecture variant, when the maximum iteration is reached, the training for the current architecture variant will be stopped. After the entire training, the network variant with the maximum validation accuracy will be selected, while in standard CNN, the validation accuracy during the training process cannot be interpreted and the stopping epoch may over-trained. Before adding a new cascade layer, the previous architecture variant has already trained, thus the new cascade layer is learning from a pre-trained model.

2. Avoid gradient vanishing problem

Training deep network could extract more 'level' of features to improve the classification accuracy; however, gradient vanishing problem will make the training failed. The cascade neural network just like other networks that contain connections straight to earlier layers, such as [15,16,17], that connection can rescale the gradient to prevent the gradient vanishing problem.

While the above analysis shows the manually designed cascade network could achieve better results than the standard CNN approach. Table 3 is shown the result of the fine-tuned network with one-shot search and other methods on the same dataset. Our method improves accuracy around 20% compared to most previous works on the SFEW dataset. The 'IPA2LT' [23] is reported as state-of-art methods in facial emotion recognition; and 'Multiple Deep Net' [12] is reported as state-of-art methods on SFEW dataset. The 'IPA2LT' is using the SFEW dataset that contains 879 training data and 406 extra validation data and the 'Multiple Deep Net' is using the FER dataset to augment the training set, where FER is another facial emotion image dataset that contains more than 3000 images. That we can conclude our method achieve comparable results with them. Also, the 'Multiple Deep Net' improves their final performance via voting 6 single models by assign different weights to them. Without the voting step, their validation accuracy was 52.29% and test accuracy was 58.06%. The voting step improves the performance of the network around 3%. To adapt our network to be a voting network is extremely easy, the only modification is stopping the searching process when there are k sub-nets remained. Then, our one-shot model will output a voting network with top-k sub-nets.
The facial emotion recognition task is unlike other knowledge-based tasks such as medical image analysis, language translation that humans have strong and correct domain knowledge on it. This allows the automatically searched network could achieve better performance than most human-designed network architecture.

5 Conclusion and Future works

This paper introduces a network architecture search method for cascade neural networks and introduces a fine-tuned cascade based neural network with multi-label learning for emotion classification. The search method together with the fine-tuned network achieves comparable results with the state-of-art method for the facial emotion recognition task. This paper also provides a way of thinking about how to combine cascade type neural networks and gradient-based search methods with other existing modern methods by discovering the basic property of them.

In this work, we only search for the best cascade network architecture and limit the search space to 9. In future development, it is possible to enlarge the search space without increase the search time significantly. For example, one possible approach is getting the advantage of weights sharing. We have built our search space using the tree search, if the searching constraints been relaxed, there will have more children share the common node. This hierarchical representation allows us to develop children-networks for each sub-net and also optimize the structure of them. This approach will only grow the time consumption linearly and still keep the optimization process within one training pass. Also, for the decided architecture, it could be further optimized by giving weights for each edge and optimize those weights, then drop the edge with low SoftMax value as we did before. Because the cascade network contains many inner connections, further optimization is able to generate diversity networks that could either be traditional feed-forward networks or be the networks with residual connections.

6 Reference

[1] A. Dhall, R.Goecke, S. Lucey, T. Gedeon. Static Facial expression analysis in tough conditions: Data, Evaluation Protocol and Benchmark.
[2] S. Khoo, T. Gedeon. Generalisation Performance vs. Architecture Variations in Constructive Cascade Networks
[3] Z. Yu, C. Zhang. Image based static facial expression recognition multiple deep network learning
[4] G.D.Praveenkumar, M.Dharmalingam. Recurrent Cascade Neural Network Classification.
[5] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua. A Convolutional Neural Network Cascade for Face Detection
[6] M.-L. Zhang, Z.-H. Zhou. A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering, 26(8):1819–1837, 2014.
[7] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data.
[8] E. Gibaja and S. Ventura. A tutorial on multilabel learning. ACM Computing Surveys, 47(3):Article 52, 2015.
[9] L. Jacob, J.-P. Vert, and F. R. Bach. Clustered multi-task learning: A convex formulation. In NIPS, 2009.
[10] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. In ICML, 2011.
[11] A. Kumar and H. Daume III. Learning task grouping and overlap in multi-task learning. ICML, 2012.
[12] Z. Yu, C Zhang. Image based Static Facial Expression Recognition with Multiple Deep Network Learning
[13] D, Hui, S. Zhou, R. Cellappa FaceNet2ExpNet: Regularizing a Deep Face Recognition Net for Expression Recognition.
[14] S. Eleftheriadis, Discriminative shared Gaussian processes for Multiview and view-invariant facial expression recognition.
[15] K.He, X.Zhang, S.Ren, J.S, Deep Residual Learning for Image Recognition.
[16] C.Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply- supervised nets. *arXiv:1409.5185*, 2014.
[17] C.Szegedy,W.Liu,Y.Jia,P.Sermanet,S.Reed,D.Anguelov,D.Er- han, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
[18] P.Zhong, M.Fukushima. A Regularized Nonsmooth Newton Method for Multi-class Support Vector Machines. 2005.
[19] I.Fischer, J.Poland. Amplifying the Block Matrix Structure for Spectral Clustering. Telecommunications Lab. 2005.
[20] M.Bilenko, S.Basu, J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. ICML. 2004.
[21] H.Liu, K.Simonyan, Y.Yang Darts: Differentiable Architecture Search
[22] T.Elsken, J.H.Metzen, F.Hutter Neural Architecture Search: A Survey
[23] J.Zeng, S.Shan, X.Chen Facial Expression Recognition with Inconsistently Annotated Datasets
[24] E.Real, A.Aggarwal, Y.Huang, Q.V.Le, Regularized Evolution for Image Classifier Architecture Search
[25] H.Cai, T.Chen, W.Zhang, Y.Yu, J.Wang, Efficient Architecture Search by Network Transformation
[26] B.Zoph, Q.V.Le, Neural Architecture Search With Reinforcement Learning
[27] E.Real, S.Moore, A.Selle, S.Saxena, Large-Scale Evolution of Image Classifiers
[28] A.Sinha, P.Malo, K.Deb A Review on Bilevel Optimization: From Classical to Evolutionary Approach