Assignment 3
Group 30

John Walker Adam Fiksen Giovanni Charles

November 21, 2012

1 Results

1.1 Confusion Matricies

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Table 1: Confusion Six output Network

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Table 2: Confusion Single Output Networks

1.2 Average Classification Rate, Recall Precision and F1 Measures

Emotion	name	Precision rate	Recall rate	f1 measure
1	anger	0	0	0
2	disgust	0	0	0
3	fear	0	0	0
4	happiness	0	0	0
5	sadness	0	0	0
6	suprise	0	0	0

Table 3: Six Output Evaluation Results

Emotion	name	Precision rate	Recall rate	f1 measure
1	anger	0	0	0
2	disgust	0	0	0
3	fear	0	0	0
4	happiness	0	0	0
5	sadness	0	0	0
6	suprise	0	0	0

Table 4: Single Output Evaluation Results

2 Fold Perfomance

Average performance per fold based on 10 fold cross validation:

Fold	Average f1 measure
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Table 5: Six Output Fold Results

Fold	Average f1 measure
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Table 6: Single Output Fold Results

3 Implementation Details

4 Questions

4.1 Discuss how you obtained the optimal topology and optimal values of network parameters. Describe the performance measure you used (and explain why you preferred it over other measures) and the different topologies / parameters you experimented with

We experimented with the following parameters

1. Number of hidden layers

2. Number of hidden neurons in each layer
3. Transfer function
4. Training Function (and function parameters)

We decided to make the search space for 1. between 1 - 2 we were able to limit the search space like this through research. For instance in the lecture notes for neural networks it says "Networks with many hidden layers are prone to overfitting and harder to train, for most problems one hidden layer should be enough, 2 hidden layers can sometimes lead to improvement". Through some rudeametary testing we established that as we had quite a large problem 2 layers did usually perform better although it took quite a lot longer so we decided that was the maximum we could fesibaly have. Thus we decided the search space was 1 - 2.

We decided to make the search space for 2. between 5 - 30 layer 1, 0 - 17 layer 2. Layer 2 was dictated mainly through time constraints we found that if layer 1 was large enough any size of layer 2 much greater than 17 would take far too long to train. The constraints from layer 1 were chosen through a mixture of expirence and research. We found that any number of neurons in layer 1 below 5 gave a horribly low classificatiobn rate we we set the lower bounday as 5. We then did some research and found a few 'rules of thumb' two given by Jeff Heaton author of Introduction to Neural Networks In Java were "the optimal size of the hidden layer is usually between the size of the input and size of the output layers" "A good number of neurons is likely to be around the average of the inputs and outputs" we tried out some networks and noticed past 30 training time simply far surpassed any improvement and therefore we decided to make this the ceiling.

We limited out speach space to the following transfer functions:

1. trainbfg
2. traincgb
3. traincgf
4. traincgp
5. traingda
6. traingdm
7. traingdx
8. trainlm
9. trainoss
10. trainrp
11. trainscg

These were selected from a list of around 20 gathered from the Matlab NNToolbox documentation. We then removed the training functions which weren't suitable for batch training and we were left with 13. We then read futher into the documentation and decided to not use 'trainb', 'traingd' and 'traigdm' the reason being "The previous section presented two backpropagation training algorithms:gradient descent, and gradient descent with momentum. These two methodsare often too slow for practical problems. In this section we discuss several high performance algorithms that can converge from ten to one hundred times faster than the algorithms discussed previously.".

4.2 Explain what strategy you employed to ensure good generalisation ability of the networks and overcome the problem of overfitting

We optimised our networks based on the f1 measure for predictions of unseen target values, this would mean that overfitted networks should perform worse and be discarded by our genetic algorithm. To make our genetic algorithm faster we discarded networks which were likely to overfit the data by bounding some of the parameters. We made sure the number of neurons per level to above 5, so it could understand the data sufficiently, and below 30 so that it is not overfitted. We also restricted the number of epochs to 100 so that the network does not become too familiar to the data.

4.3 In Part VIII you used the optimal parameters that you found in part VI to train your networks. However, there is a problem with this approach, the data you used for validation at some point will be used for testing in cross-validation. Can you explain why this a problem? Ideally how should you optimise the parameters in cross-validation?

We were given a number of examples and targets we then divided these into 2/3 training and 1/3 validation data. The validation data then influences the neural network which is produced. Essentially you are training the neural network to perform well for the validation data and you hope that the validation data is large and representitive enough so that the neural network you produce generalises well and can classify unseen examples. When you come to perform cross validation you then use a portion of the data as testing data and the rest as training/validation data. This means that eventually you will be testing a section of data which was the network was geared for this will produce a very low error and thus will skew the average error rate affecting your cross validation result. To combat this the parameters (data) given to cross validation should be rearranged into a random order this will reduce the effect of any 'seen' data and any effects will be evenly spread and the average will not be skewed, althought it could be raised very slightly.

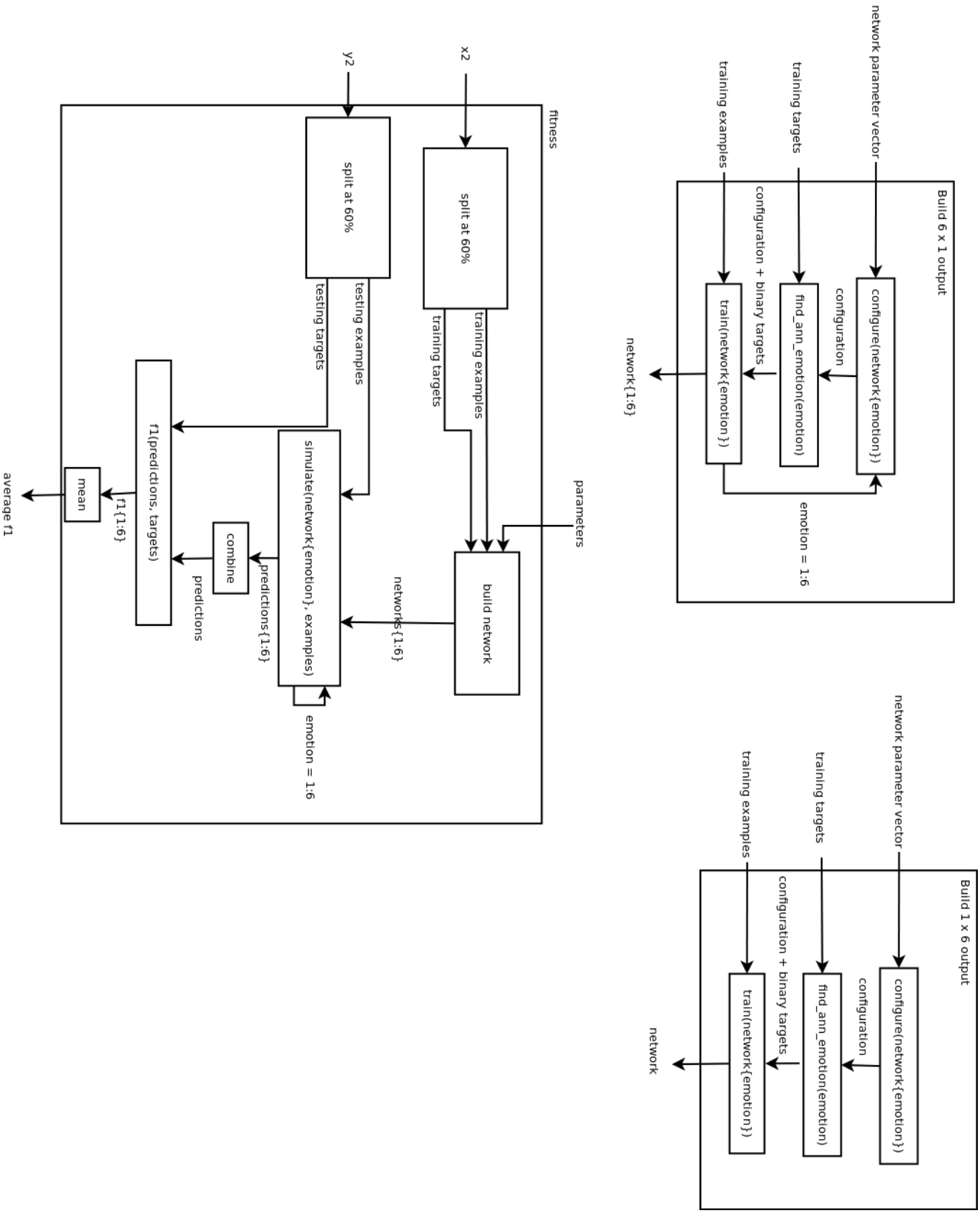
4.4 Is there any difference in the classification performance of the two different classification approaches. Discuss the advantages / disadvantages of using 6 single-outut NNs vs. 1 six-output NNs

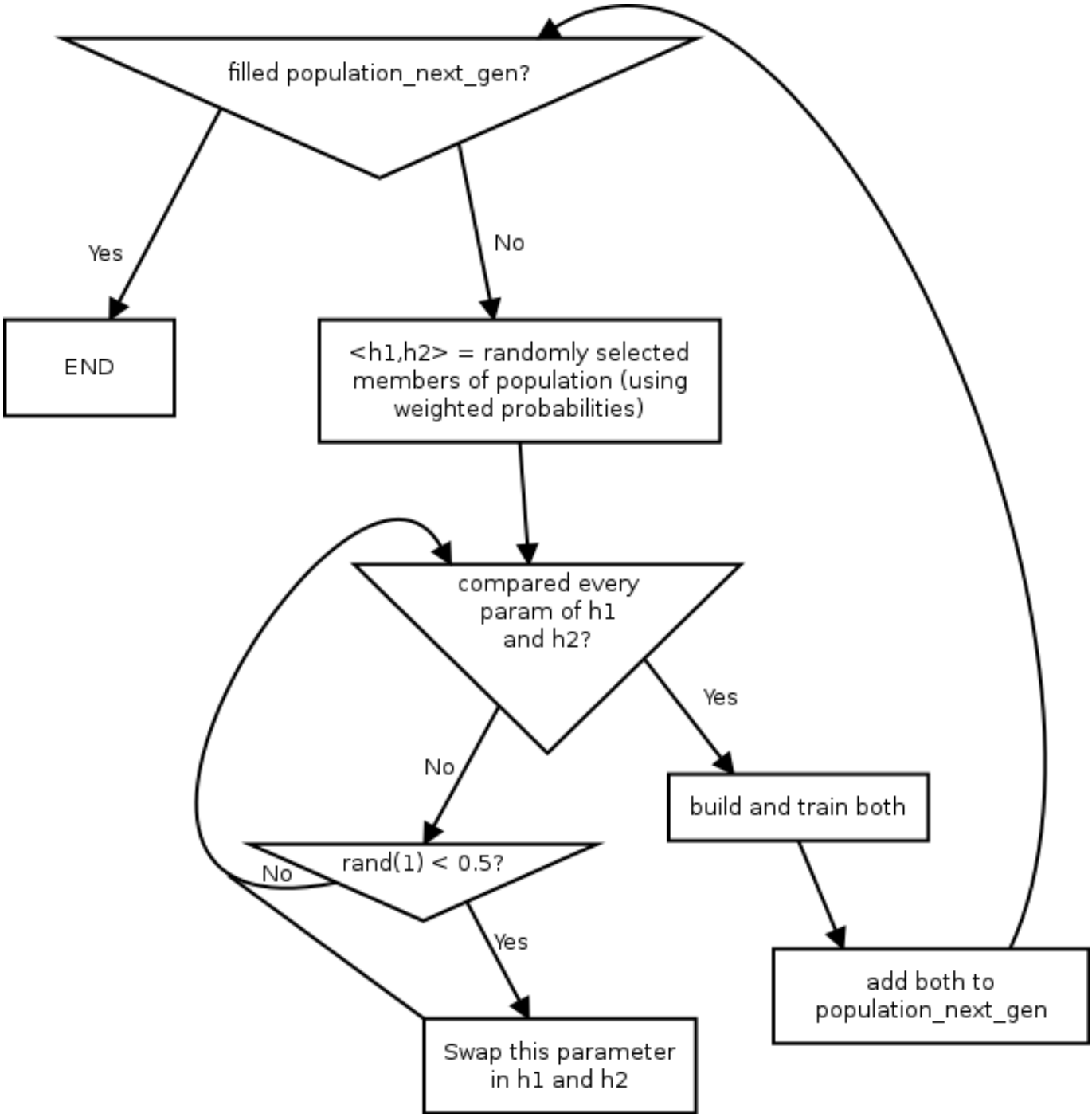
Six single output neural networks perform much better than a single network.

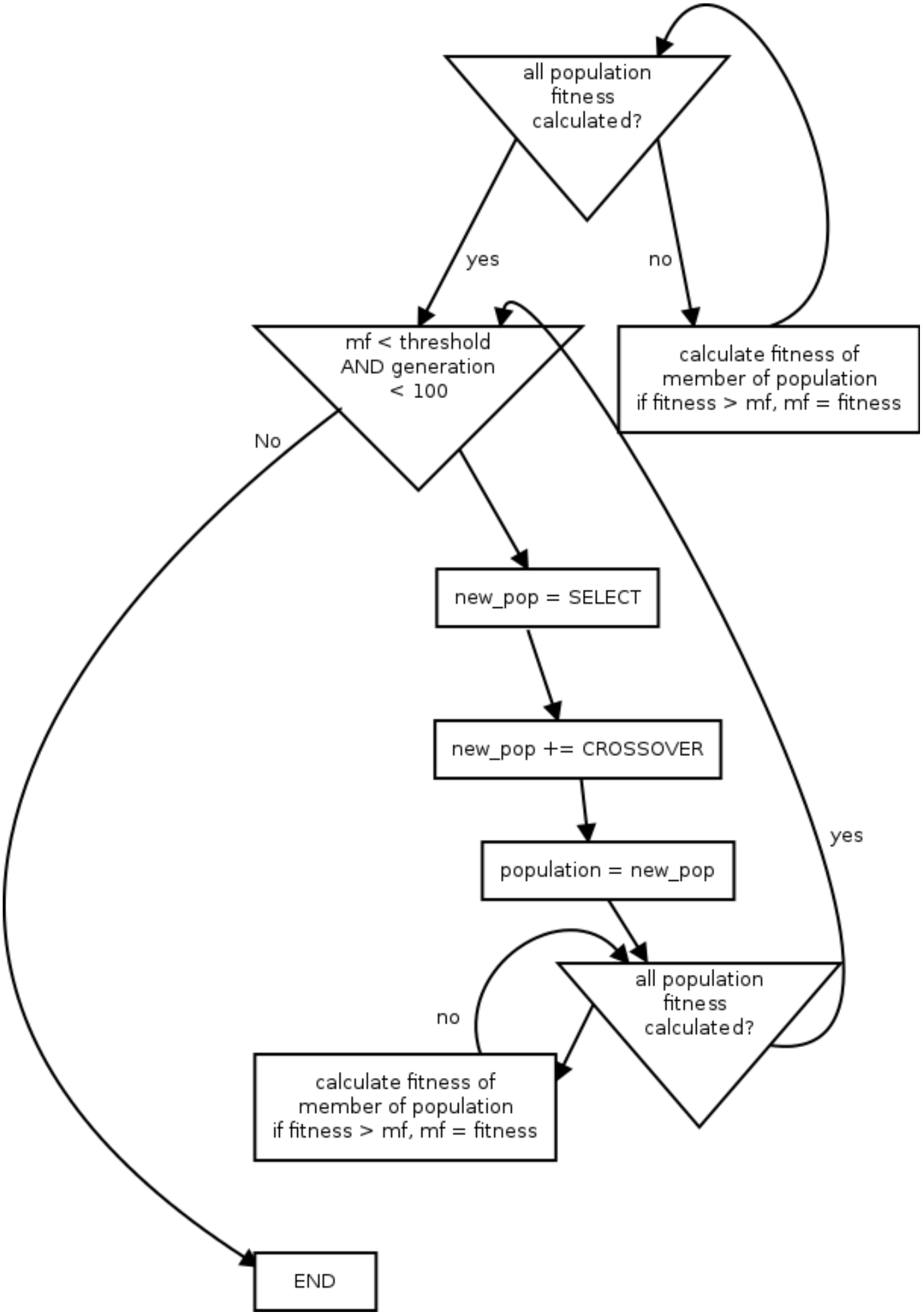
The six networks have the advantage of collectively forming a larger, more complicated and understanding, network without overfitting individually. The isolation between the networks allows for parallelism and for greater parameter specialisation for different emotions. One disadvantage is that having multiple networks may introduce merging conflicts when resolving an answer. This could be solved by limiting outputs to continuous output so that results are very unlikely to match or by ranking networks based on reliability, for example f1 measure, classification rate, most general/simple (i.e. fewer neurons), which also adds to the processing.

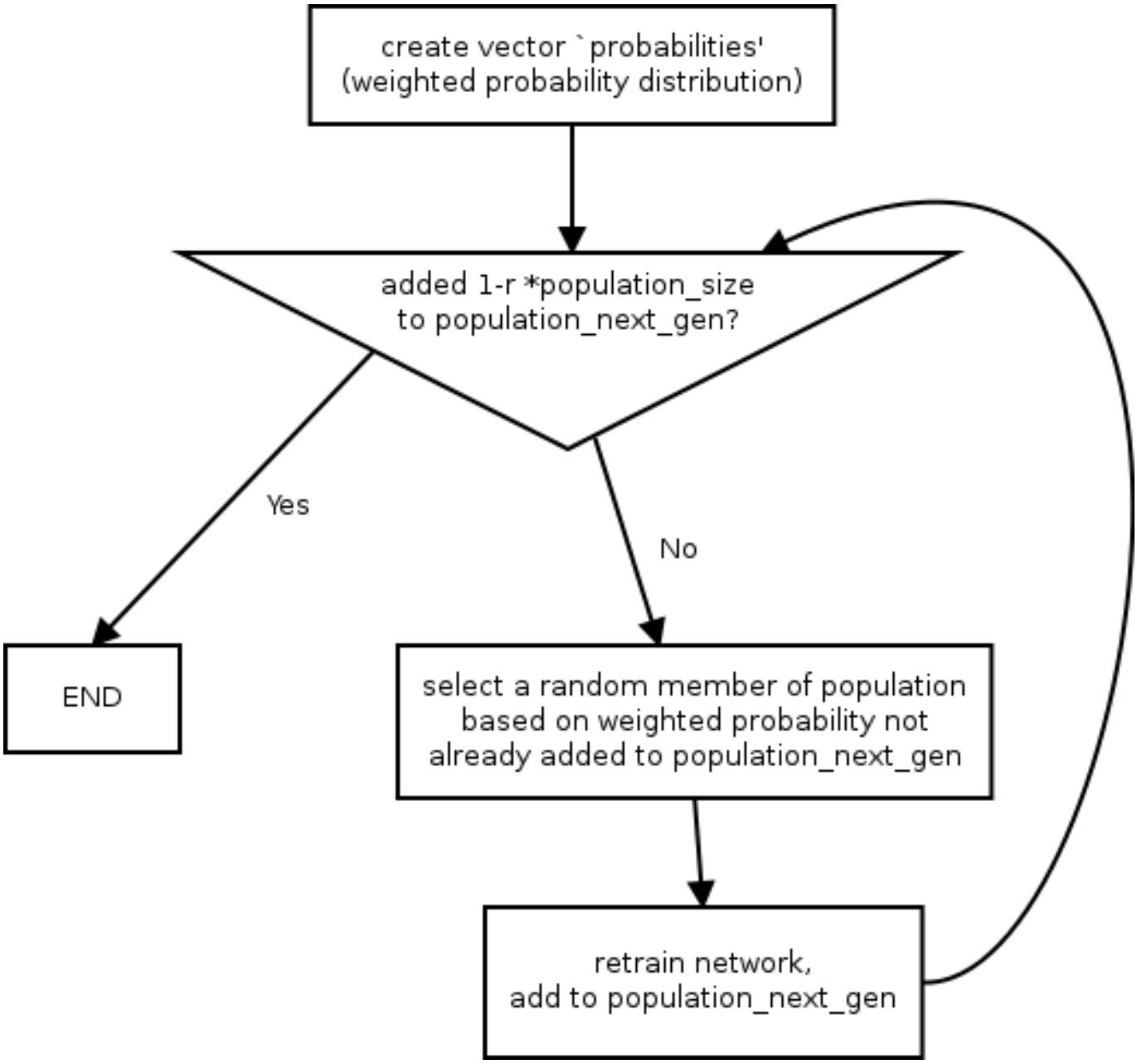
The single network however forms a smaller network which is less memory intensive and quicker to train and run. This means more aggressive optimisations can be performed resulting in parameters which are more finely tuned. This network would be more suited for machines whrecomputing resources are limited.

5 Code Flow Charts




filled population_next_gen?
Yes
No
END
<h1,h2> = randomly selected members of population (using weighted probabilities)
compared every param of h1 and h2?
Yes
No
build and train both
rand(1) < 0.5?
No
Yes
add both to population_next_gen
Swap this parameter in h1 and h2


all population fitness calculated?
yes
no
mf < threshold AND generation < 100
calculate fitness of member of population if fitness > mf, mf = fitness
No
new_pop = SELECT
new_pop += CROSSOVER
population = new_pop
yes
all population fitness calculated?
no
calculate fitness of member of population if fitness > mf, mf = fitness
END


create vector `probabilities'
(weighted probability distribution)
added 1-r *population_size
to population_next_gen?
Yes
No
END
select a random member of population
based on weighted probability not
already added to population_next_gen
retrain network,
add to population_next_gen