# Machine Learning Homework 3

## P1: Build Convolution Neural Network

### Image Preprocess

First center the image pixelwise so each pixel has zero mean. Then divide each pixel with maximum absolute value among all pixels, so all value of all the pixels are within [-1, 1] [1].

### CNN Structure

The final structure of my CNN is

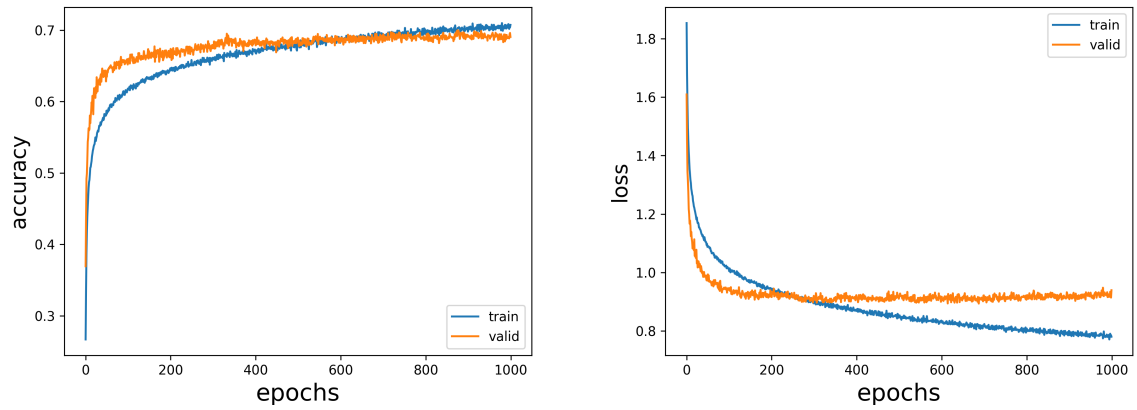
- Input (48, 48, 1)
- Image Augmentation (48, 48, 1): Augmentate images in batch by randomly
  - rotating in (-30, 30) degrees.
  - flipping horizontally.
  - shifting horizontally and vertically in (-4.8, 4.8) pixels.
  - zooming by (0.8, 1.2)
- Conv2d (24, 24, 32): Conv2d with kernel size (3, 3), stride 1, same padding, followed by batch normalization, leaky ReLU, max pooling with kernel size (2, 2), stride 1.
- Conv2d (12, 12, 64): Conv2d with kernel size (3, 3), stride 1, same padding, followed by batch normalization, leaky ReLU, max pooling with kernel size (2, 2), stride 1.
- Dropout with rate 0.1
- Conv2d (6, 6, 128): Conv2d with kernel size (3, 3), stride 1, same padding, followed by batch normalization, leaky ReLU, max pooling with kernel size (2, 2), stride 1.
- Dropout with rate 0.2
- Conv2d (3, 3, 256): Conv2d with kernel size (3, 3), stride 1, same padding, followed by batch normalization, leaky ReLU, max pooling with kernel size (2, 2), stride 1.

- Dropout with rate 0.2
- Dense (512): followed by batch normalization, leaky Relu.
- Dropout with rate 0.5
- Dense (256): followed by batch normalization, leaky Relu.
- Dropout with rate 0.1
- Dense (7): followed by activation function softmax.

### Training Settings

Optimizer Adam is used, with learning rate 0.001, decay 0.00005.

### Training Process

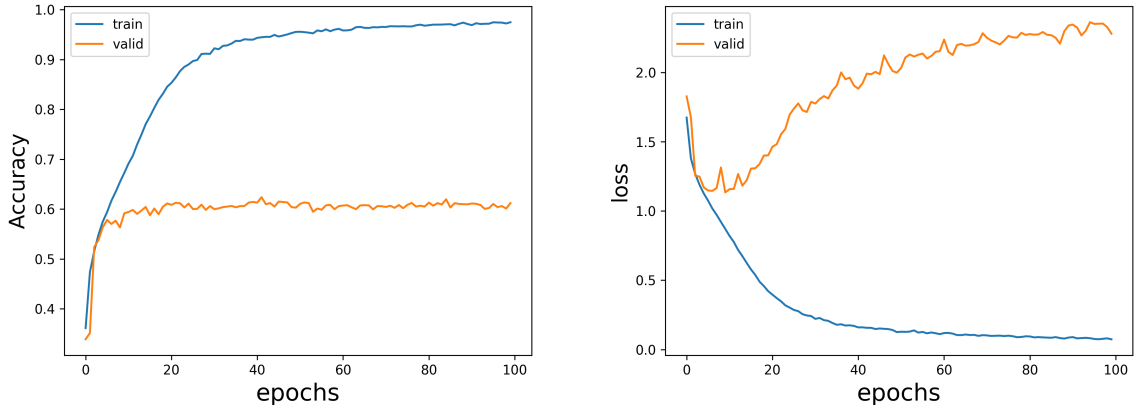



(a) Accuracy (b) Loss

Figure 1: Training Procedure

### Experiments

The same structure without image augmentation is also tried. As shown below, it overfit severely start from about 10 epochs. Therefore, image augmentation is necessary to get higher accuracy.




(a) Accuracy (b) Loss

Figure 2: Training Procedure Without Image Augmentation

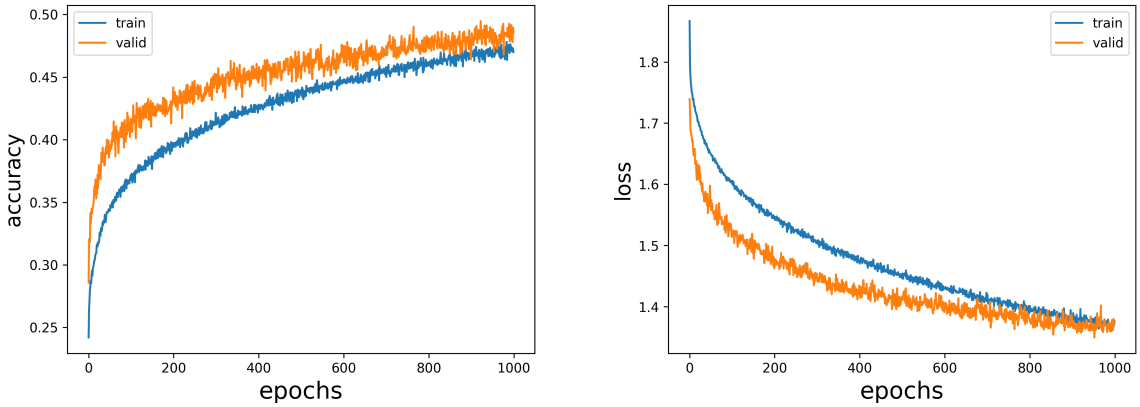
## P2: Build Deep Neural Network

To compare with CNN model, I built a DNN model with same number of layers and approximately same number of parameters. The structure of DNN model is described bellow:

- Input (48, 48, 1)
- Flatten (2304)
- Dense (512), followed by batch normalization and leaky ReLU.
- Dense (512), followed by batch normalization and leaky ReLU.
- Dropout with drop rate 0.1.
- Dense (512), followed by batch normalization and leaky ReLU.
- Dropout with drop rate 0.1.
- Dense (256), followed by batch normalization and leaky ReLU.
- Dropout with drop rate 0.1.
- Dense (256), followed by batch normalization and leaky ReLU.
- Dropout with drop rate 0.1.
- Dense (256), followed by batch normalization and leaky ReLU.
- Dropout with drop rate 0.1.
- Dense (7), followed by activation function softmax.

There are 1974791 trainable parameters, while for CNN model, there are 1703623 parameters.

As shown in figure 3, it is very hard for the DNN model to fit. Even after 1000 epochs, the training loss and accuracy still can hardly catch up with validation. In comparison, for the CNN model, the training loss gets lower than validation loss start from about 210 epochs. If the accuracy is compared, apparently CNN model is much more powerful.




(a) Accuracy (b) Loss

Figure 3: Training Procedure of DNN

## P3: Analyze the Model by Confusion Matrix

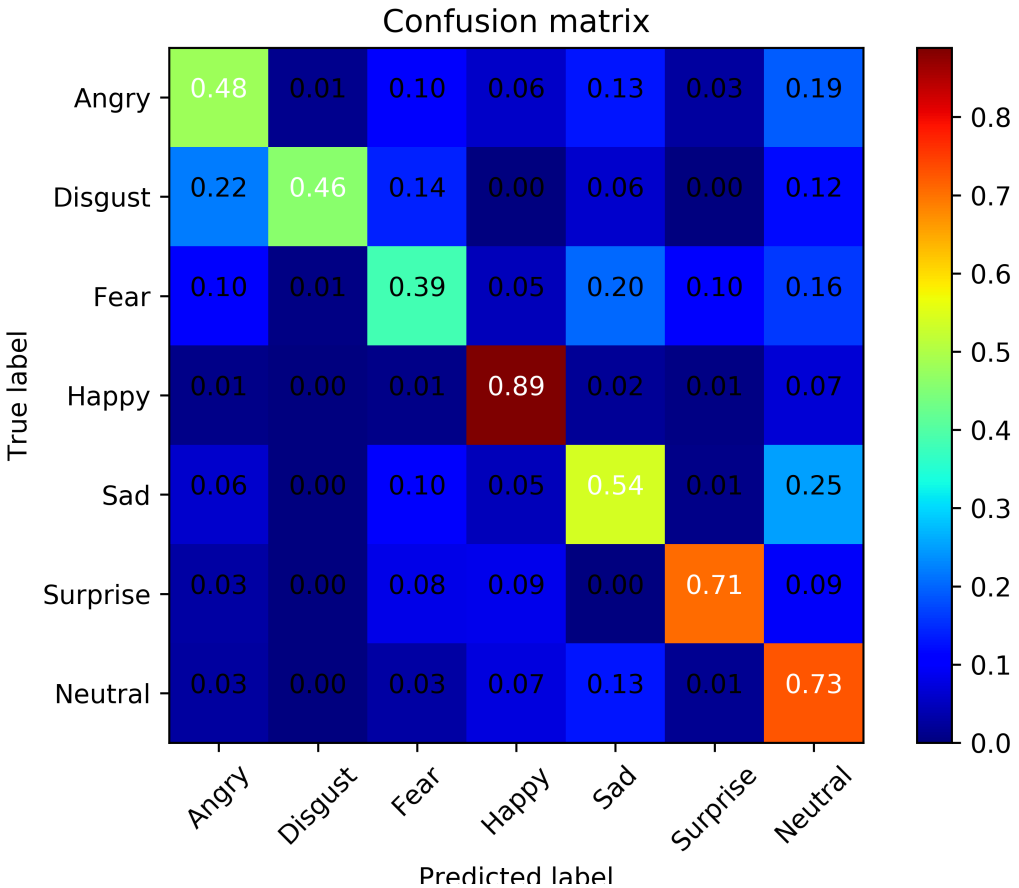
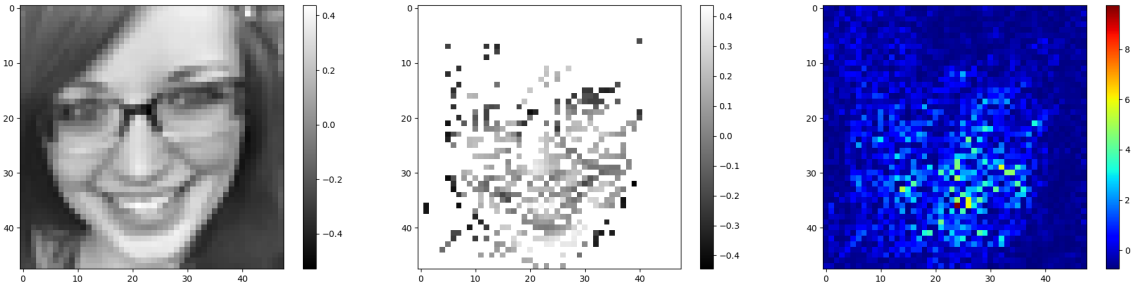



Figure 4: Confusion Matrix

As shown in Figure 4, happy is the most easy facial expression to identify for the model, while fear is the most difficult. Also, 25% of sad expressions is identify as neutral. That may implies that the facial expression of sad is not very apparent.

## P4: Analyze the Model by Plotting the Saliency Map

The figures below are the saliency map of my CNN on figure 4 in the test data set. The heatmap is generate by calculating absolute value of gradient. From the image where low heat pixels are removed, we can identify the facial expression easily. Also, most part of the mouth is preserved. Therefore, the shape of mouth should be a important feature for the model to distinguish facial expression.



(a) Original Image (b) Remove low heat (c) Saliency Map

Figure 5: Training Procedure

## P5: Analyze the Model by Visualizing Filters

Following images are the output of filters in layer 2 and images that activate those filters most. From the output of those filters, we can identify organs such as eye, eyebrow, mouth. Thus those filters seem to try to mark those facial organs in different ways. Also, some filters seem to be complementary, such as filter 25 and 26, while some of them are very similar. The similarity between filters may imply that number of filters is more than necessary. However, those redundant filters may provides better ability to resist noise.

From the images that activate the filters most, we can barely find any information. The reason may be that, those filters are to find more subtle features of part of organs, while they are too obscure to our human beings.

The second layer of another model, referred as dlsvm, proposed by the winner of FER 2013 [2], can be compared here. For this model I can only tune to about 65% accuracy. As shown in figure, some filters, such as filter 26, are also trying to mark those organs. However, apparently its filters preserve less features and less delicate. That may be reasonable since it has less number of layers.

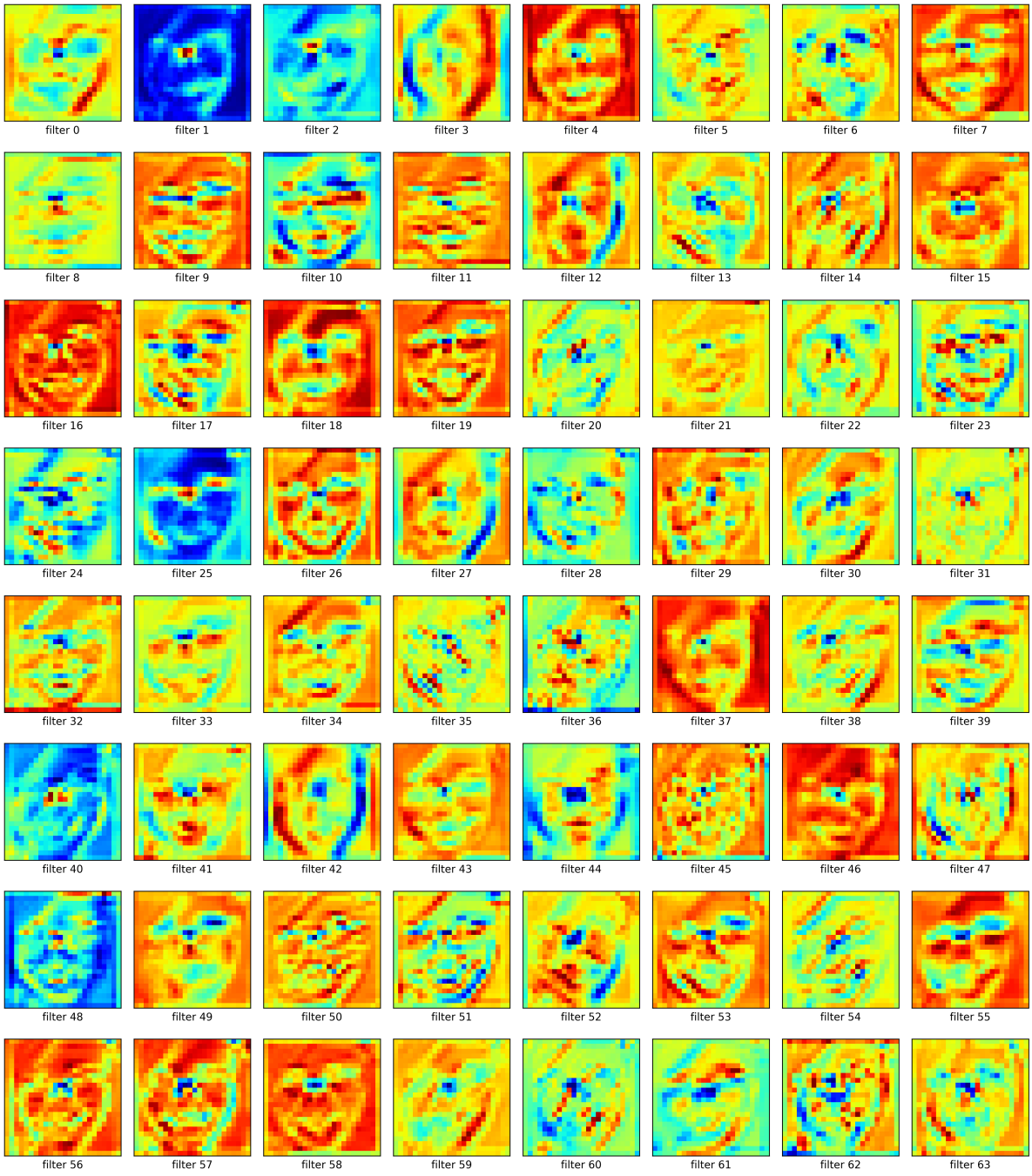



Figure 6: Output of filters in layer 2

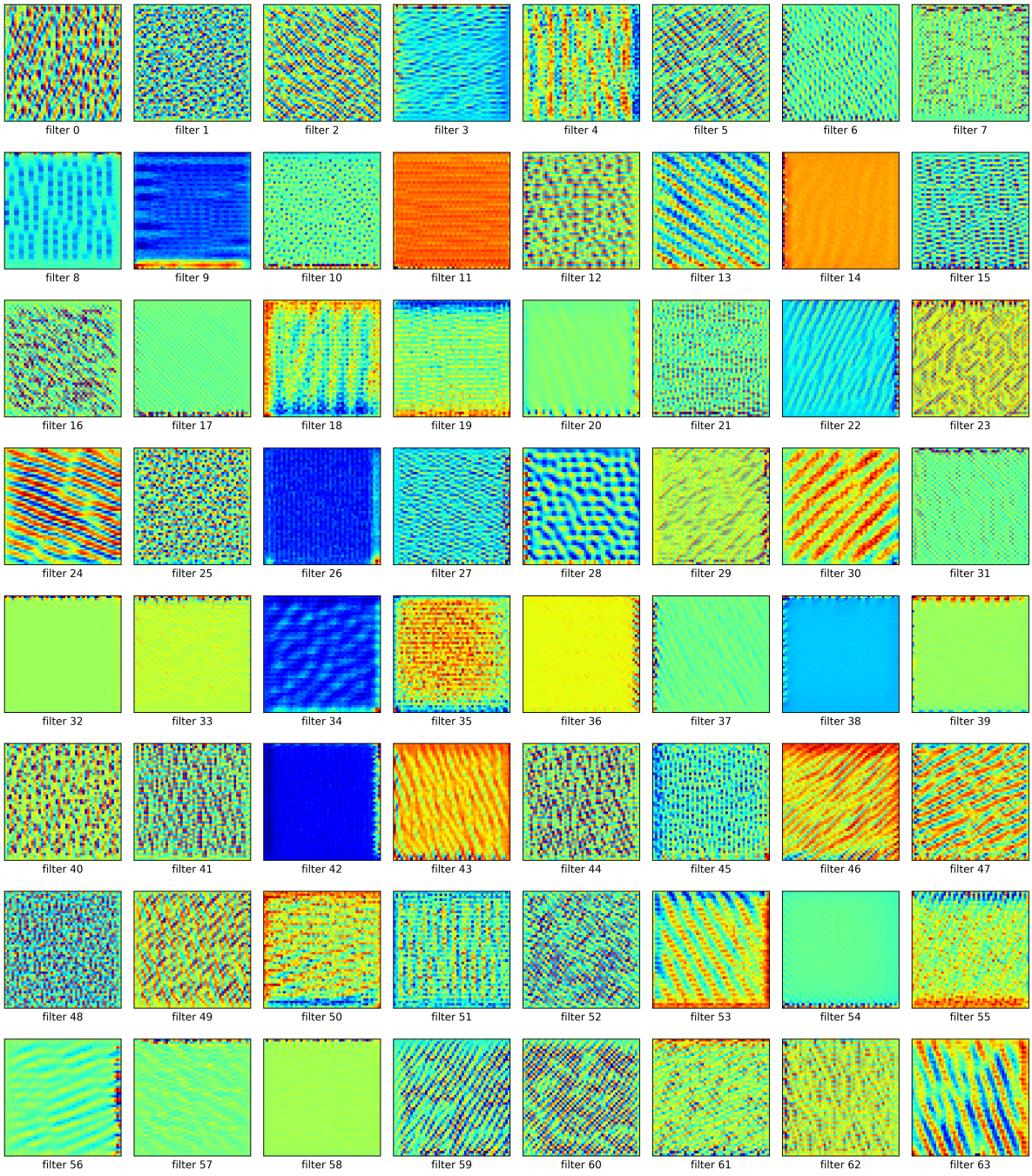



Figure 7: Visualization of filters in layer 2

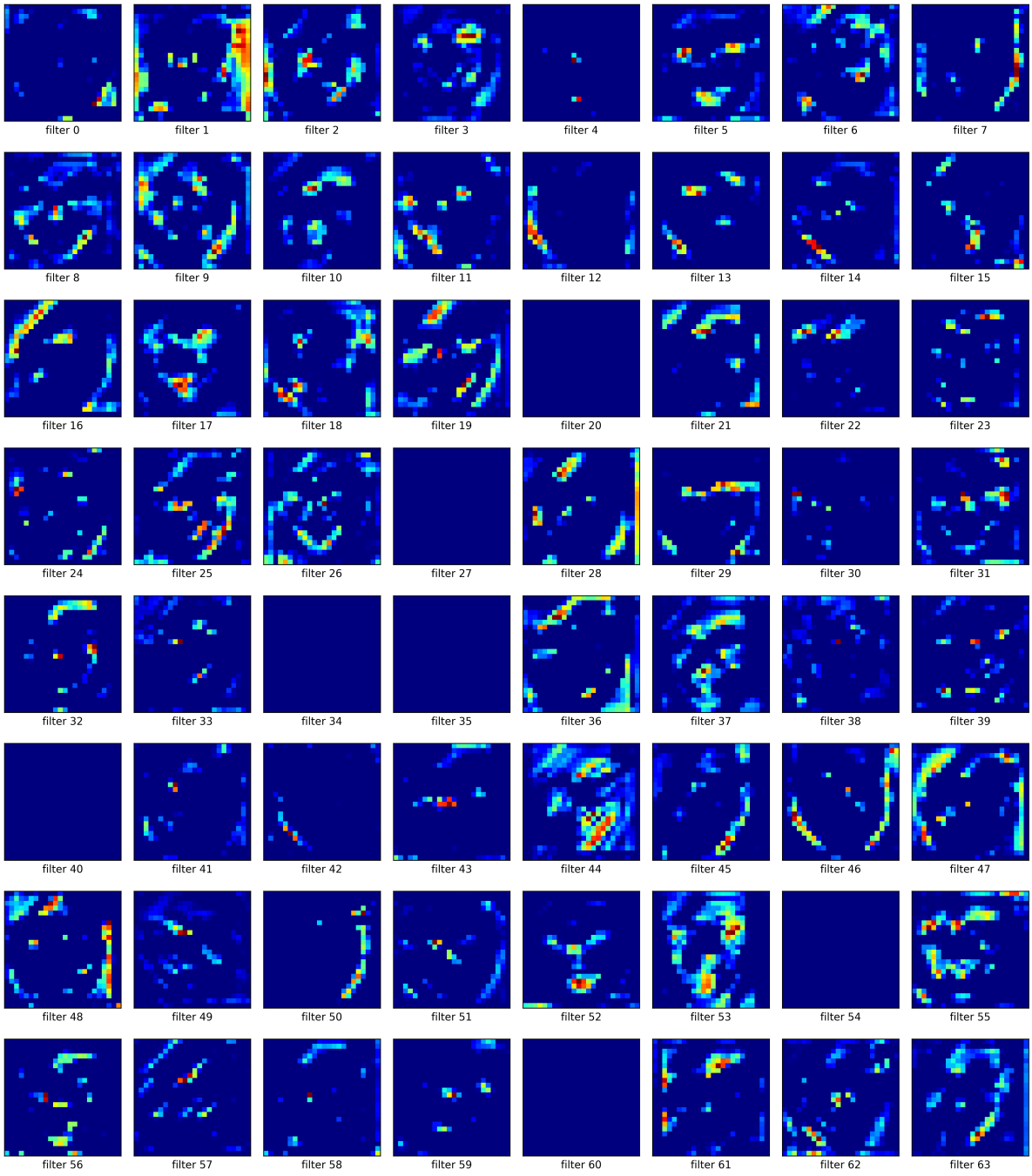



Figure 8: DLSVM: Output of filters in layer 2

# References

[1] Mo, Kaichun, Yang, Yuxin, Zhang, Yun. CS 221 Project Final Report: Facial Expression Automatic Recognition. http://www.cs.stanford.edu/ kaichun/resume/cs221_project_report.pdf

[2] Yichuan Tang. Deep Learning using Linear Support Vector Machines.