

Assignment 2 Report

CZ4042 Neural Network & Deep Learning

Name: He Yuhao

Matric. Number: U1722945E

Name: Li Guanlong

Matric. Number: U1722033H

**Introduction**

This project aims to apply Convolution Neural Network (CNN) and Recurrent Neural Network (RNN) to address real case classification problems. Different hyper-parameters and different ways of learning are applied, and by analysing the training costs as well as test accuracies, the differences among the models are discussed. In part A, CNN is utilized for the object recognition problem, different models are compared based on their accuracies. In part B, both CNN and RNN are used to classify text with different embedding. The training time and accuracies of different combinations of embeddings and models are used.

**Methods**

For both parts, the programming language used is Python, the framework used is TensorFlow.

In Part A, the neuron network consists of two convolutional layers, two max pooling layers, one fully connected layer and one softmax output layer. Cross entropy is used as the lost function. The effectiveness of different numbers of feature maps in the convolution layers are analysed. Besides, different ways of learning including gradient descent (GD) learning, GD with momentum, RMSProp algorithm, Adam optimizer and GD with dropout are explored.

In Part B, first the text is basically preprocessed by removing punctuations, numbers, stop words, and new lines. Then two different text embedding methods, including character embedding and word embedding are introduced. Based on different embeddings, different models also introduced, such as CNN and RNN. For CNN, it contains filters and pooling layer, ends up with a fully connected linear layer, with cross entropy as lost function. For RNN, vanilla rnn, gru, lstm are tried out.

**Experiments and Results**

**Part A:** **Object Recognition**

1. After configuring the CNN properly, with GD learning using batch size of 128, the 10000 training samples are used to train the CNN.

a. The training cost and test accuracy against learning epochs is plotted, as shown in Figure 1 and Figure 2 respectively.

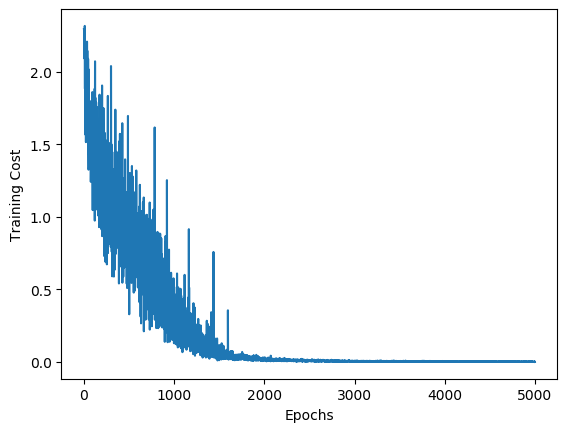
**** ****

Figure 1: Training cost Figure 2: Test accuracy

b. Two test images are randomly selected, the two images and their feature maps at both convolution layers and pooling layers are shown in Figure 3, 4, 5, 6, 7, 8, 9, 10, 11,12. Figure 3 is the first image, Figure 4, 5, 6, 7 are the first image’s feature maps at the first convolution layer, the first pooling layer, the second convolution layer and the second pooling layer respectively. Figure 8 is the second image, Figure 9, 10, 11, 12 are the second image’s feature maps at the first convolution layer, the first pooling layer, the second convolution layer and the second pooling layer respectively.

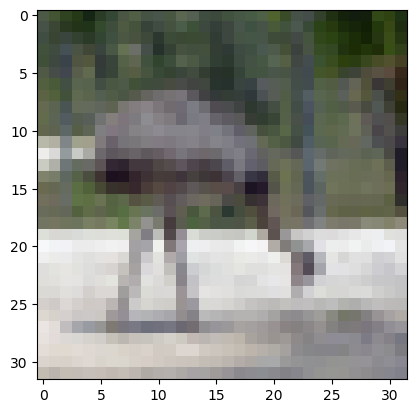


Figure 3: The first image

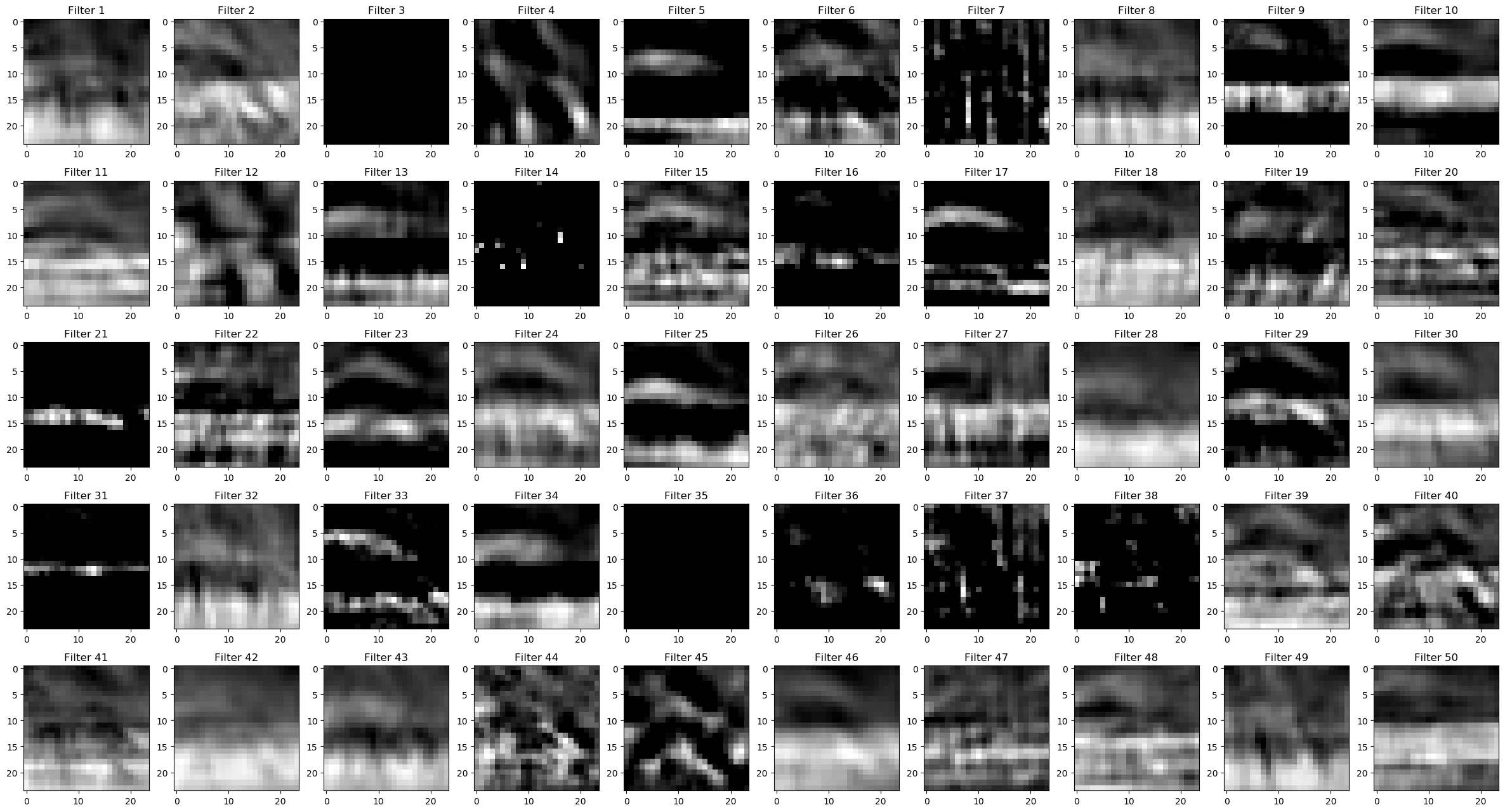


Figure 4: The first image’s feature maps at the first convolution layer

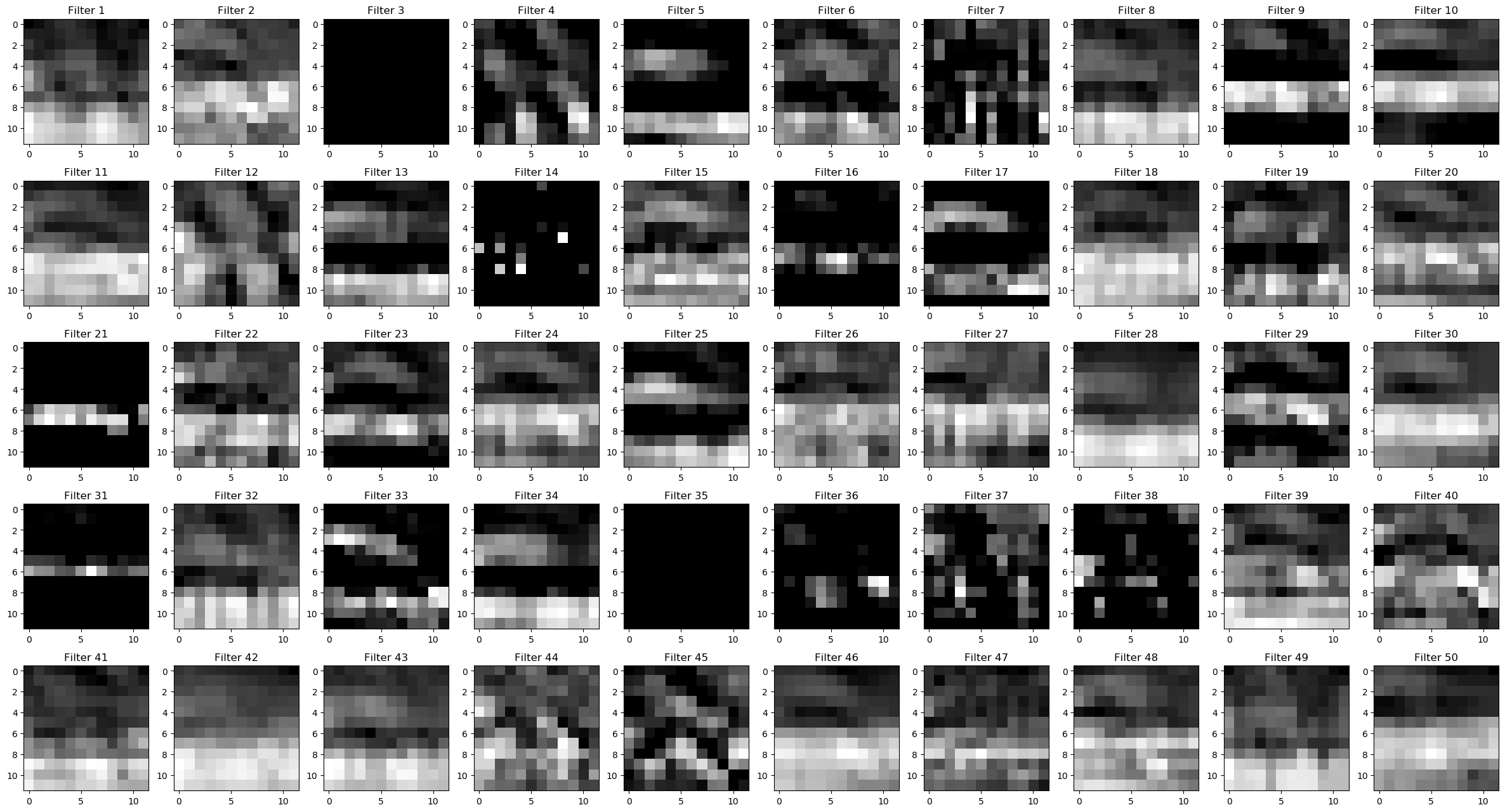


Figure 5: The first image’s feature maps at the first pooling layer

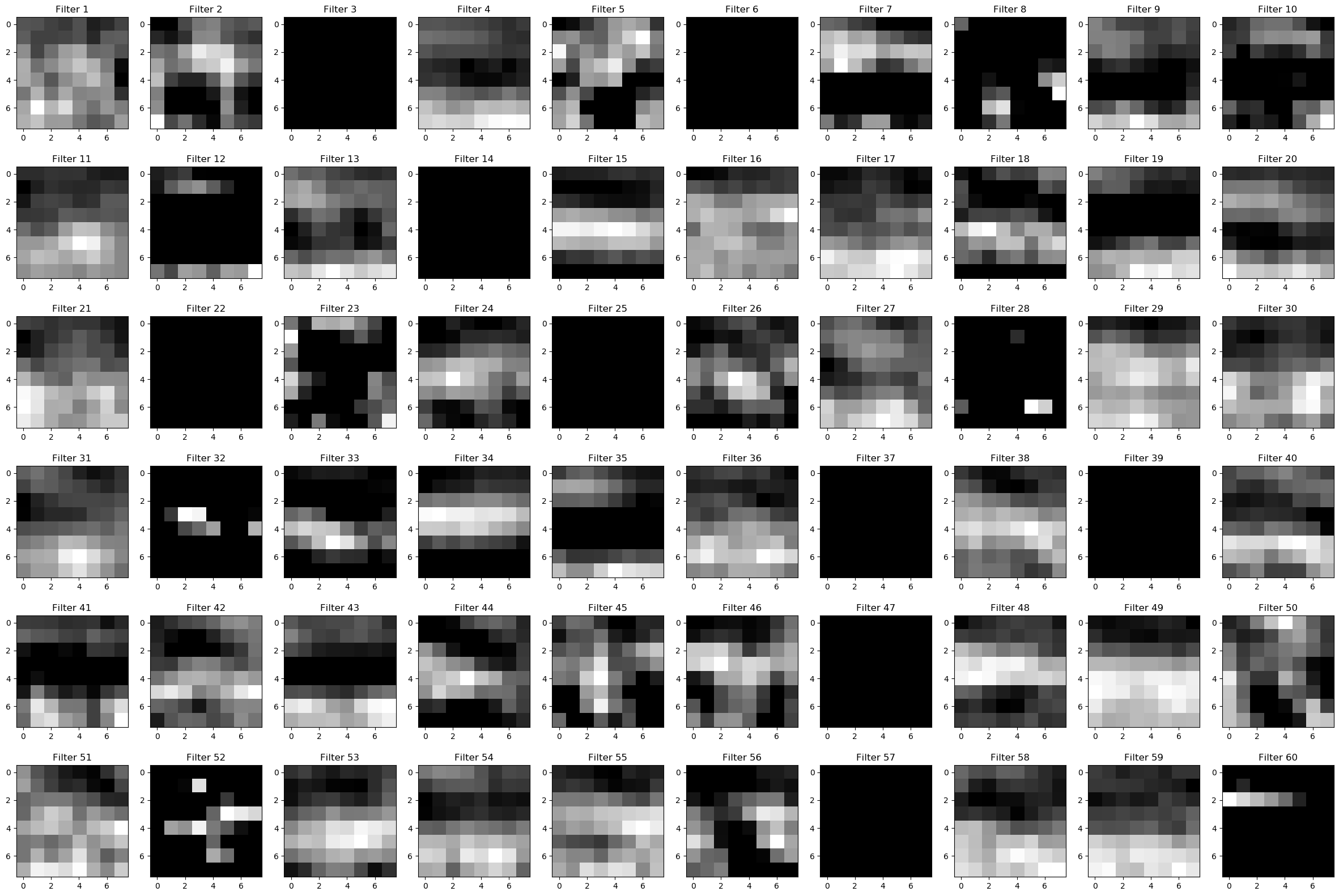


Figure 6: The first image’s feature maps at the second convolution layer

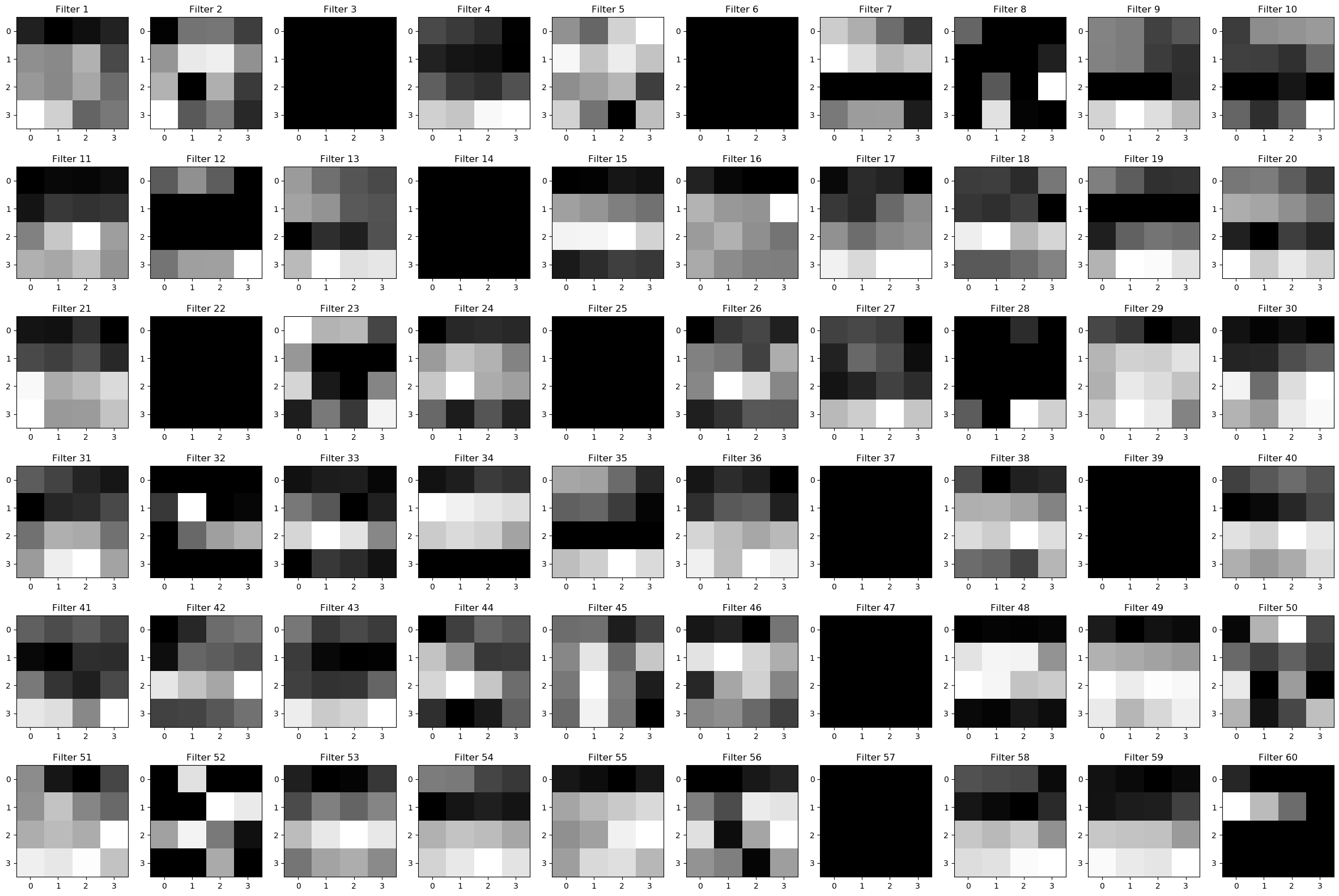


Figure 7: The first image’s feature maps at the second pooling layer

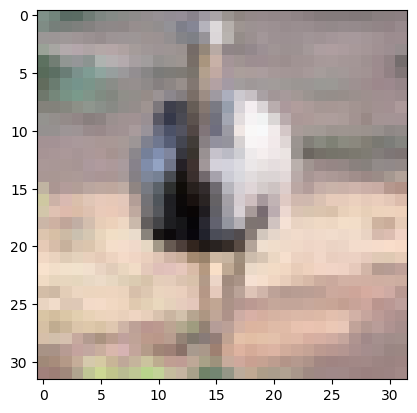


Figure 8: The second image

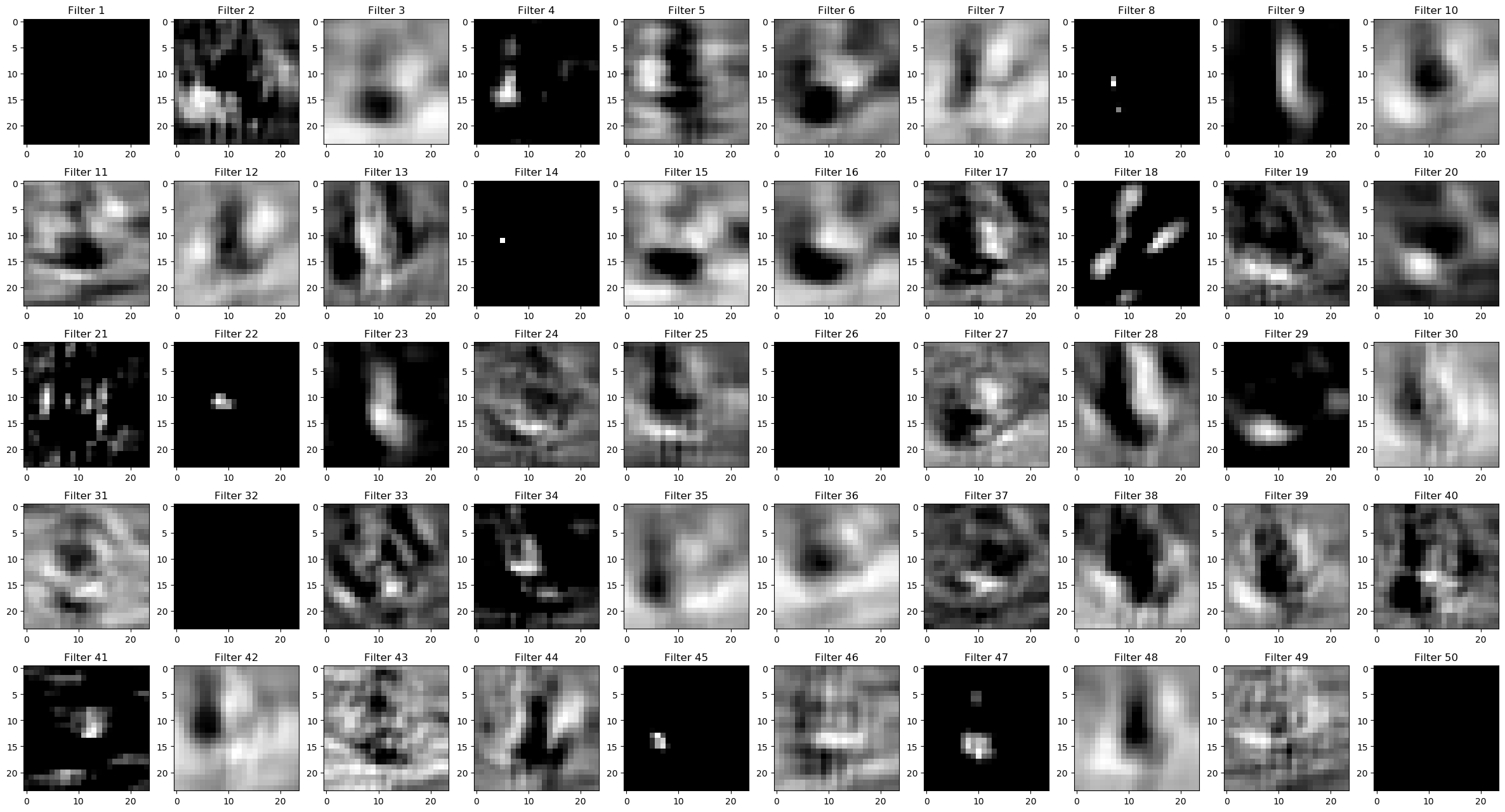


Figure 9: The second image’s feature maps at the first convolution layer

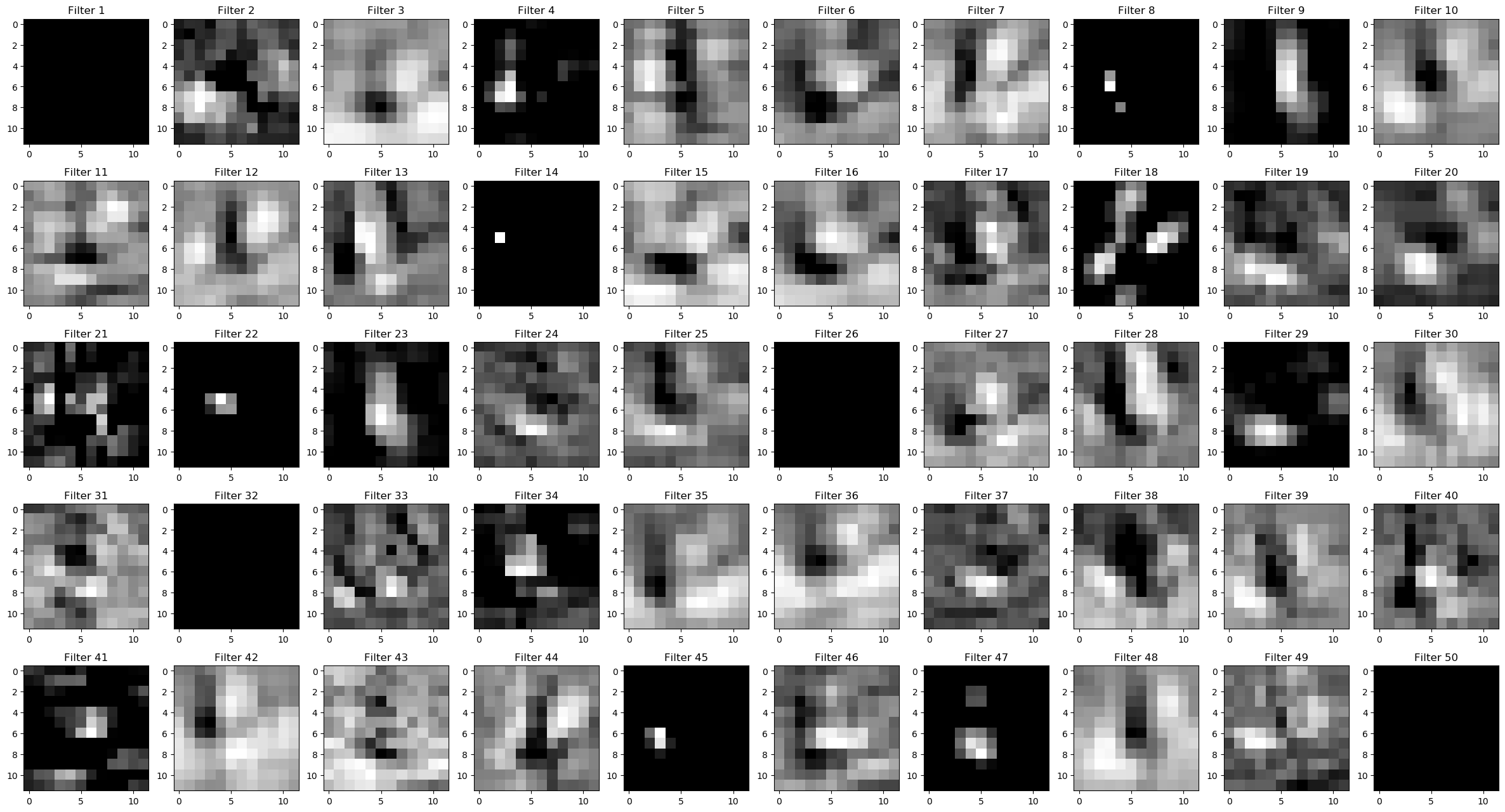


Figure 10: The second image’s feature maps at the first pooling layer

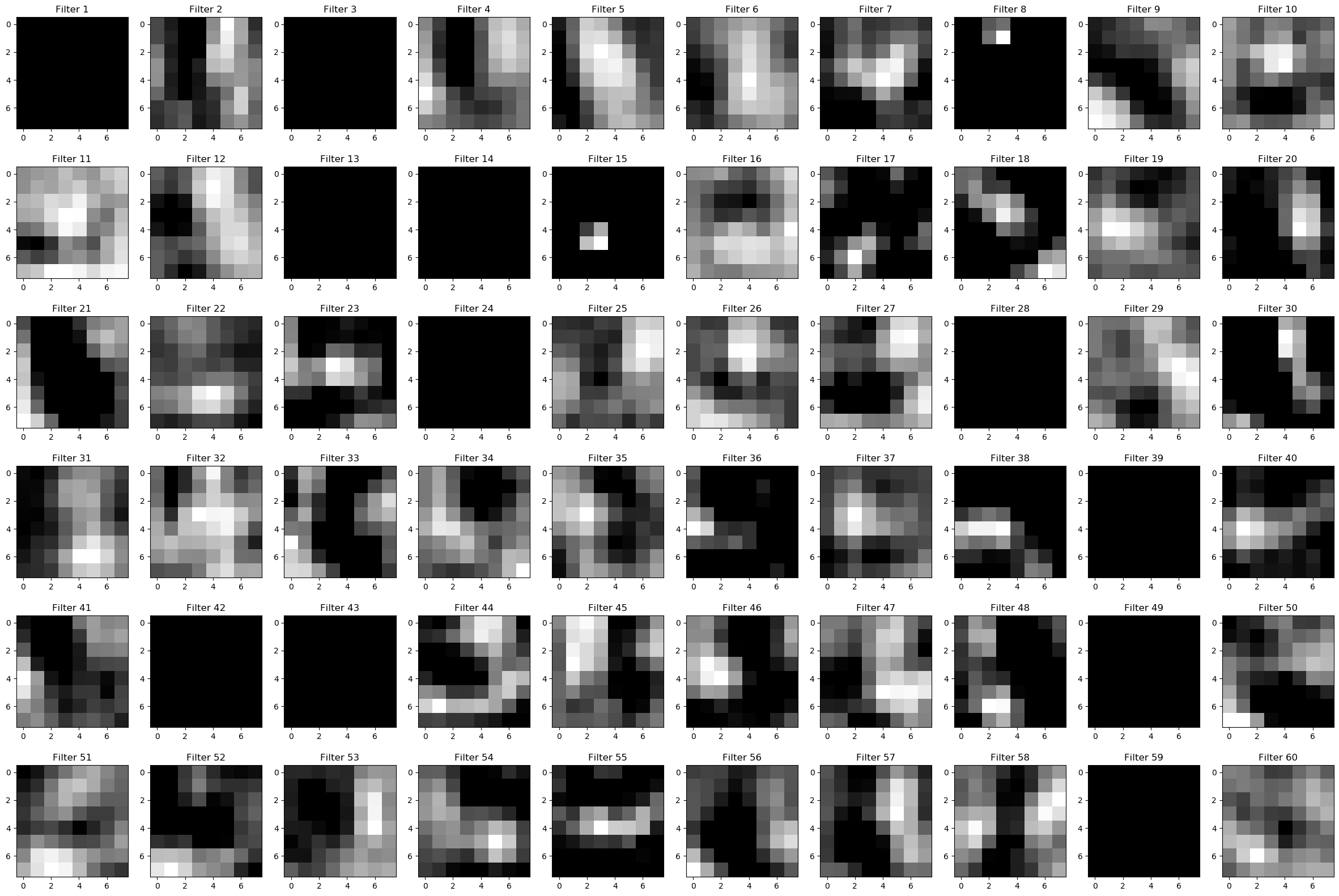


Figure 11: The second image’s feature maps at the second convolution layer

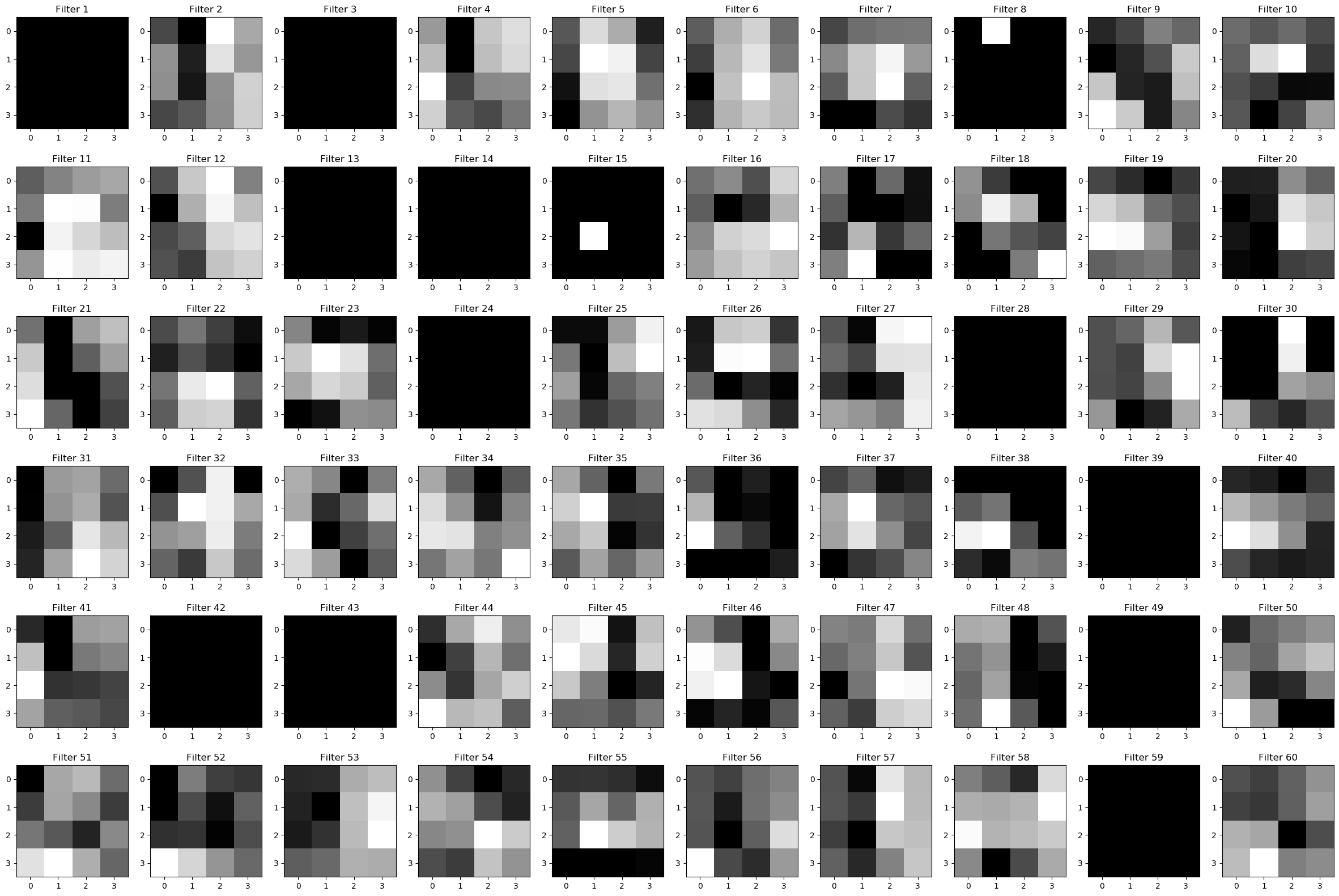


Figure 12: The second image’s feature maps at the second pooling layer

2. With a grid search, the [number of feature maps in first convolution layer, number of feature maps in second convolution layer] combinations of [40, 40], [40,50], [40, 60], [40, 70], [50, 40], [50,50], [50, 60], [50, 70], [60, 40], [60,50], [60, 60], [60, 70], [70, 40], [70,50], [70, 60], [70, 70] are examined one by one. The test accuracy against training epoch diagram for each of the combinations is shown in Figure 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 respectively.

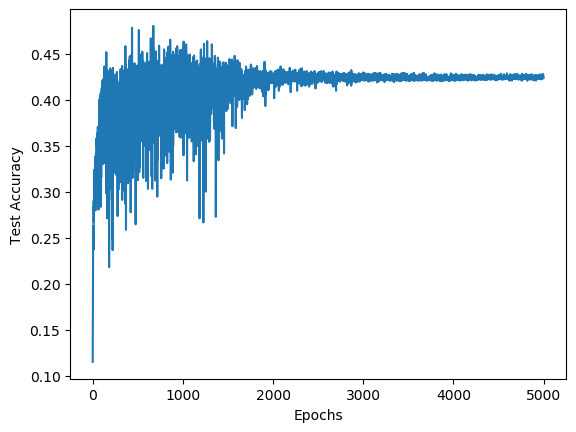
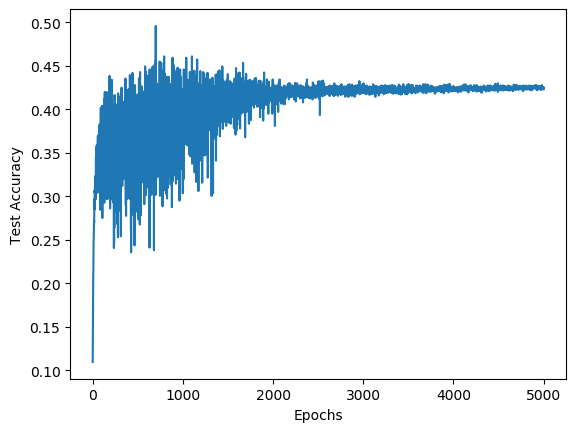
** **

Figure 13: Test accuracy for [40, 40] Figure 14: Test accuracy for [40, 50]

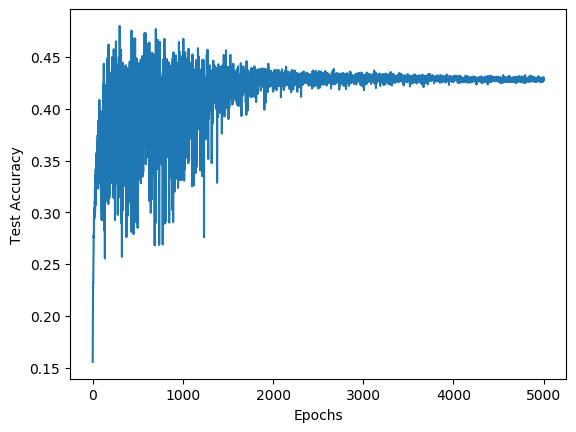
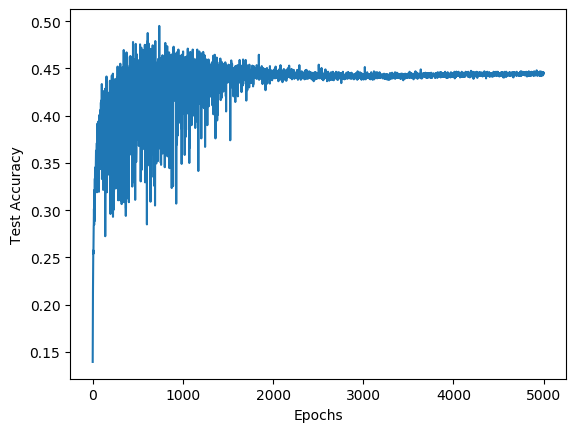
 

Figure 15: Test accuracy for [40, 60] Figure 16: Test accuracy for [40, 70]

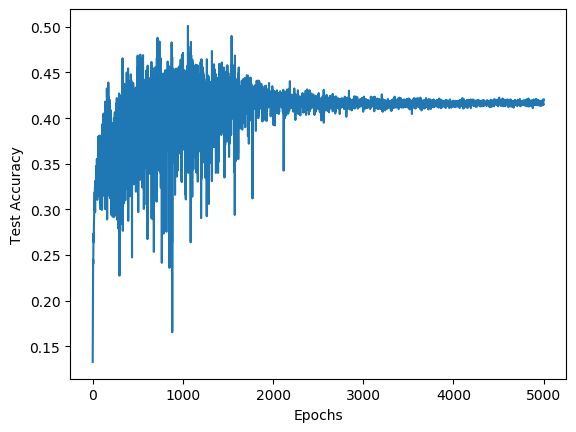
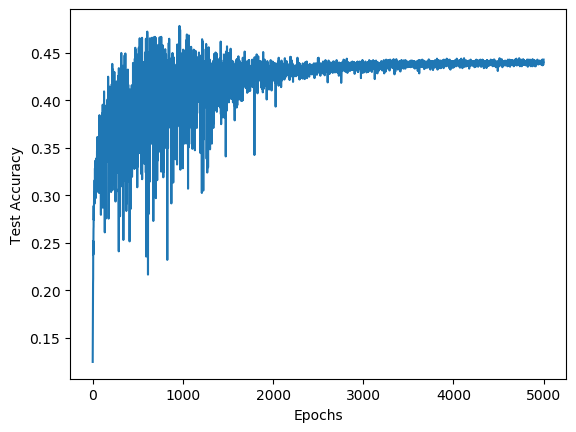
 

Figure 17: Test accuracy for [50, 40] Figure 18: Test accuracy for [50, 50]

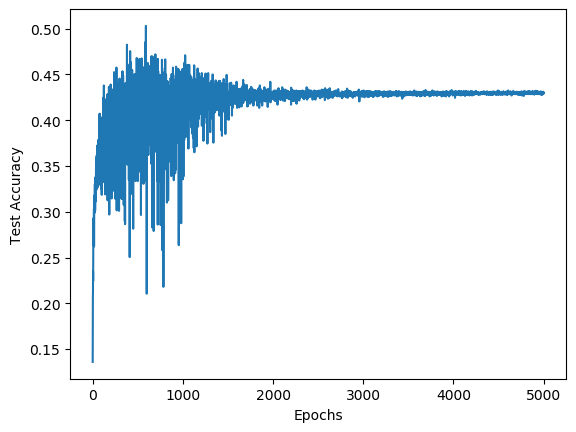
** **

Figure 19: Test accuracy for [50, 60] Figure 20: Test accuracy for [50, 70]

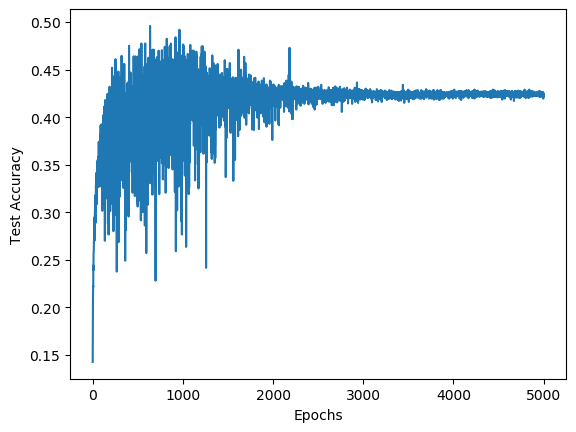
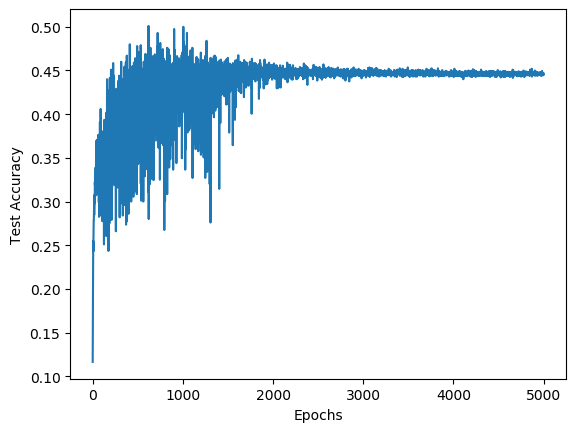
** **

Figure 21: Test accuracy for [60, 40] Figure 22: Test accuracy for [60, 50]

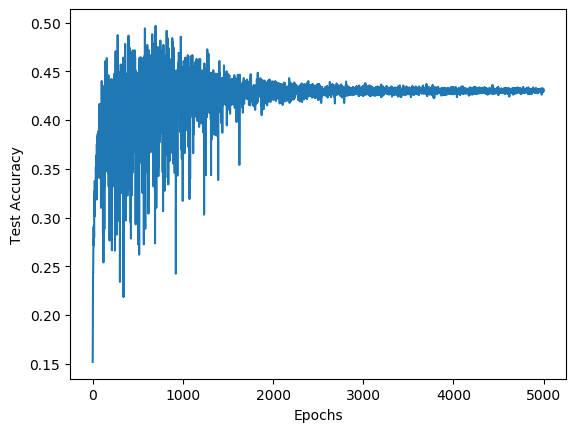
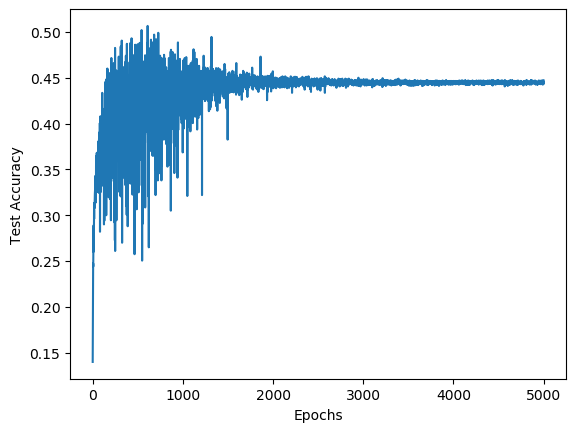
 ****

Figure 23: Test accuracy for [60, 60] Figure 24: Test accuracy for [60, 70]

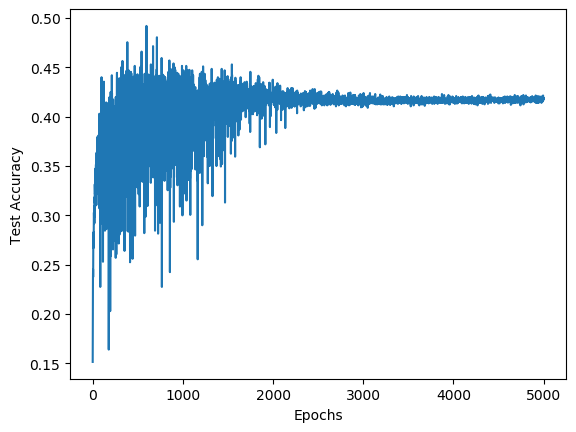
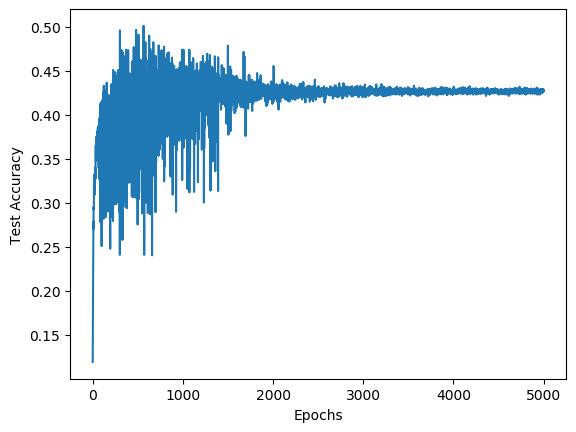
 

Figure 25: Test accuracy for [70, 40] Figure 26: Test accuracy for [70, 50]

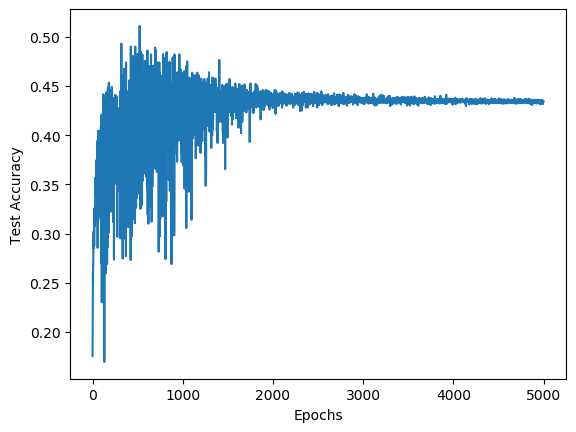
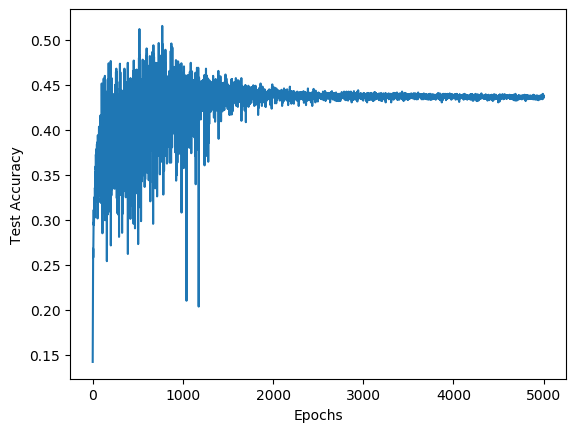
 

Figure 27: Test accuracy for [70, 60] Figure 28: Test accuracy for [70, 70]

From the figures above, and the data obtained by the grid search, it can be found that the combinations [40, 70], [60, 50] and [60, 70] can achieve the highest test accuracy, their test accuracies approximately converge to 0.445.

Since a smaller number of convolution layer feature maps will lead to a smaller usage of computational resources, the combination [40, 70] is selected to be the optimal number of feature maps. In other words, the optimal number of feature maps in the first convolution layer is 40, and the optimal number of feature maps in the second convolution layer is 70.

3. Using the optimal number of filters [40, 70], the CNN is trained by four different methods one at a time.

a. A momentum term 𝛾 = 0.1 is added to the network, with the momentum term, the training costs and test accuracies against epochs is plotted, as shown in Figure 29 and Figure 30.

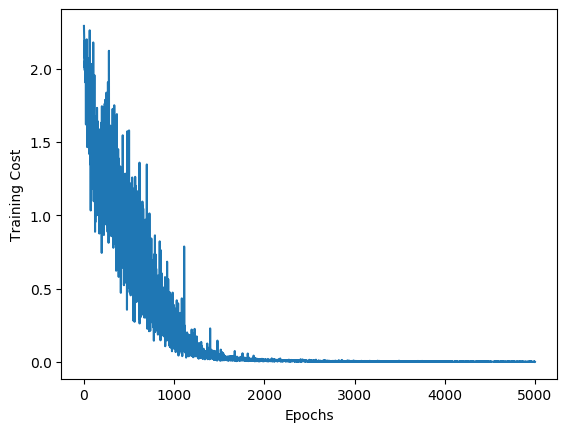
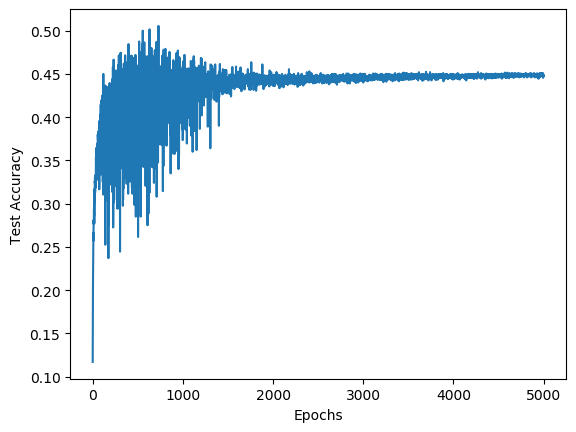
 

Figure 29: Training cost for momentum Figure 30: Test accuracy for momentum

b. When RMSProp algorithm is utilized for learning, the training costs and test accuracies against epochs is plotted, as shown in Figure 31 and Figure 32.

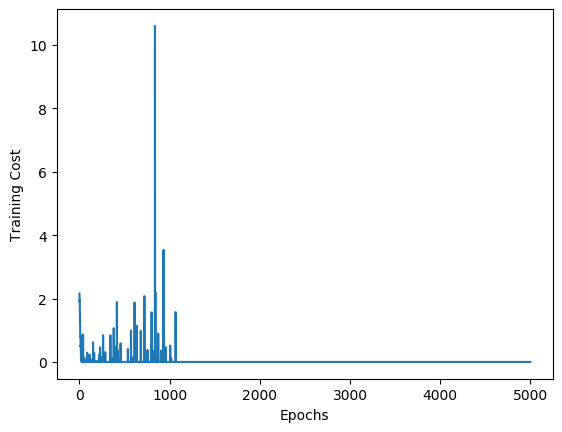
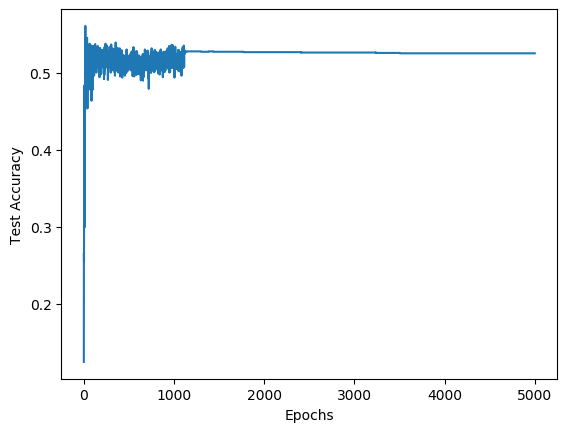
 

Figure 31: Training cost for RMSProp Figure 32: Test accuracy for RMSProp

c. When Adam optimizer is used for learning, the training costs and test accuracies against epochs is plotted, as shown in Figure 33 and Figure 34.

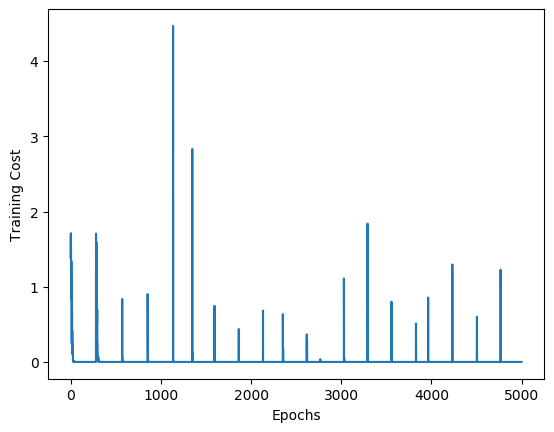
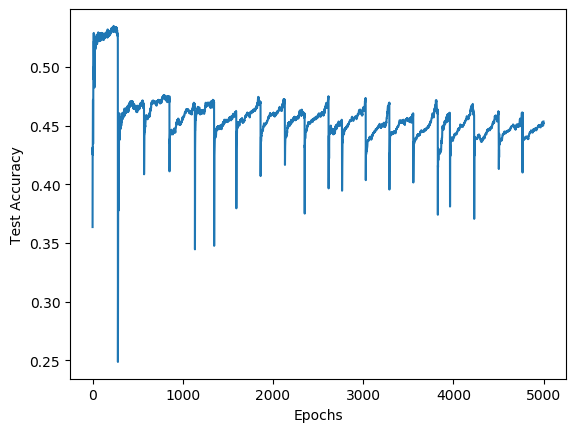
 

Figure 33: Training cost for Adam Figure 34: Test accuracy for Adam

d. When dropout with a rate of 0.5 is applied to the fully connected layer, the training costs and test accuracies against epochs is plotted, as shown in Figure 35 and Figure 36.

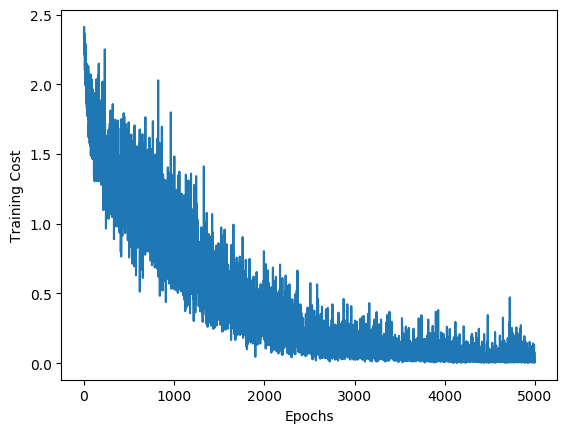
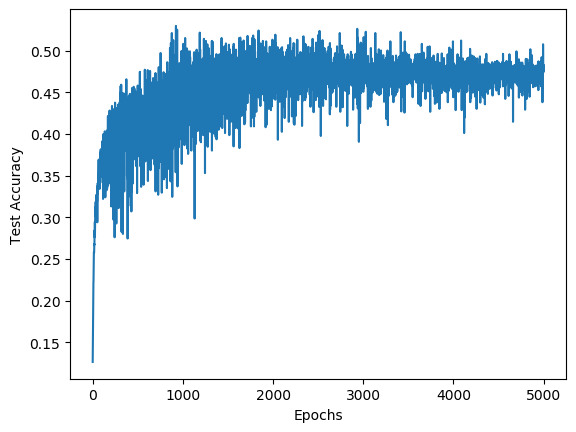
 

Figure 35: Training cost for dropout Figure 36: Test accuracy for dropout

4. From the accuracies of all the models from the previous parts, specifically Figure 16, 30, 32, 34, 46. It can be observed that:

* Among all the training methods, Adam optimizer and RMSProp algorithm can achieve the highest test accuracy, which approximately converges to 0.53. The test accuracies of GD with momentum and GD without any additional methods both converge to around 0.445. The test accuracies of GD with dropout has a fairly high tendency to fluctuate, by taking the average of the fluctuating accuracies, it can be concluded that the accuracies converge to around 0.47.
* Between Adam optimizer and RMSProp algorithm, it can be observed that Adam optimizer takes a shorter time to converge, but it has a periodic behaviour. That is to say, the test accuracy for the Adam optimizer can converge in less than 100 epochs, but if the training continues, the test accuracy will drop after around 200 epochs and rise to convergence again after less than 100 epochs. Therefore, from Figure 14, it can be concluded that for Adam optimizer, the training needs to be stopped after the accuracy converges for the first time, and this will yield a test accuracy of around 0.53. Meanwhile, it can be observed that for RMSProp algorithm, the test accuracy converges after around 1100 epochs, and tends to remain at around 0.53 even if the training continues. In general, both Adam optimizer and RMSProp algorithm can lead to a higher accuracy.
* In this particular CNN, the momentum does not have an obvious impact on the test accuracy and the epochs needed for the accuracy to converge.
* Adding a dropout to the fully connected layer is able to improve the test accuracy. However, it also makes the test accuracy prone to fluctuate.

**Part B:** **Text classification**

Many different models and embedding combination are tested out:

|  |  |  |
| --- | --- | --- |
| Model | Text embedding | With dropout |
| CNN | Character embedding | No |
| 1 layer Vanilla RNN | Word embedding | Yes |
| 1 layer GRU |  |  |
| 1 layer GRU with gradient clipping |  |  |
| 2 layer GRU |  |  |
| 1 layer LSTM |  |  |

There are 6\*2\*2 = 24 combinations, we will use MED to refer to the combination.

Eg: M1E1D1 🡺 Model1 + Embedding1 + Dropout1 🡺 CNN + Character embedding + No dropout

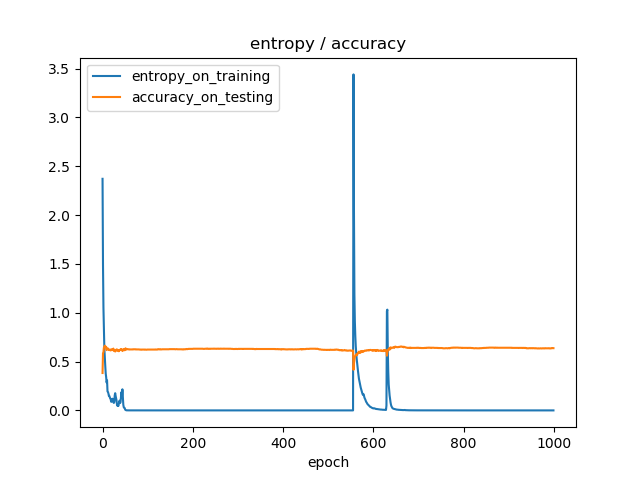
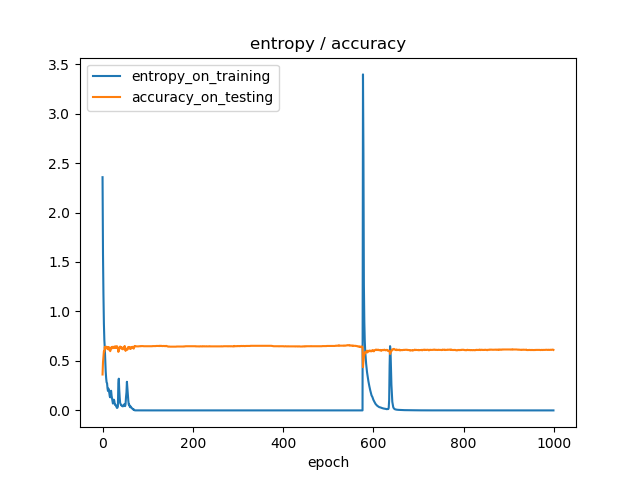
 

Figure 37: CNN char with dropout Figure 38: CNN char without dropout

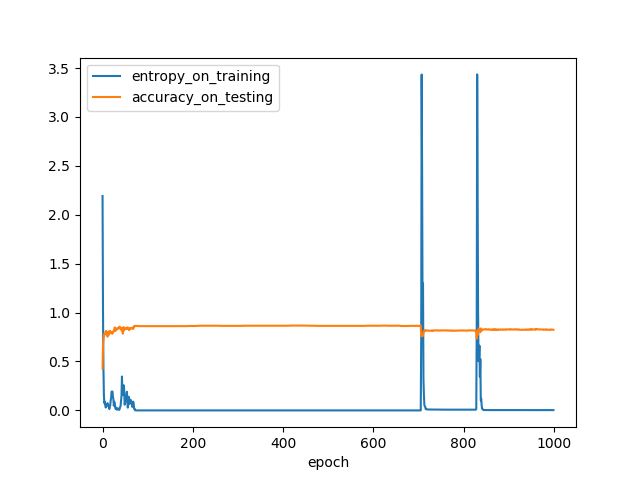
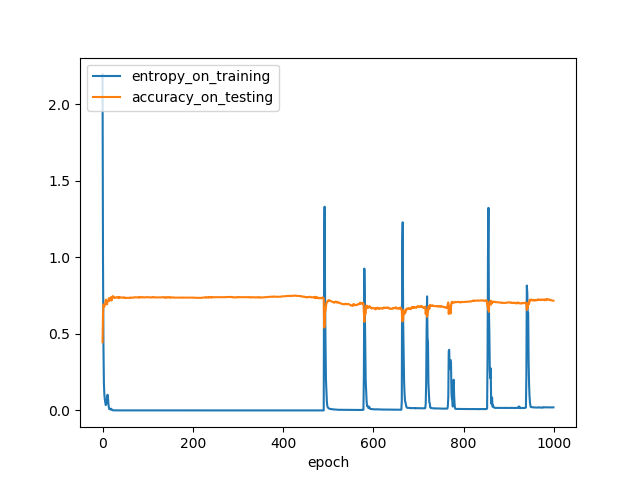
 

Figure 39: CNN word with dropout Figure 40: CNN word without dropout

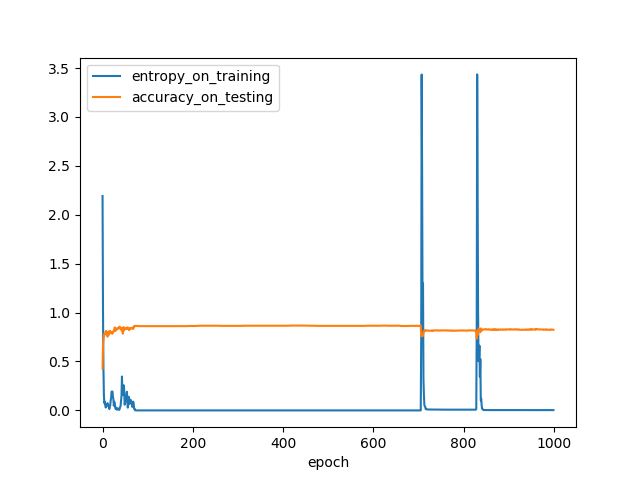
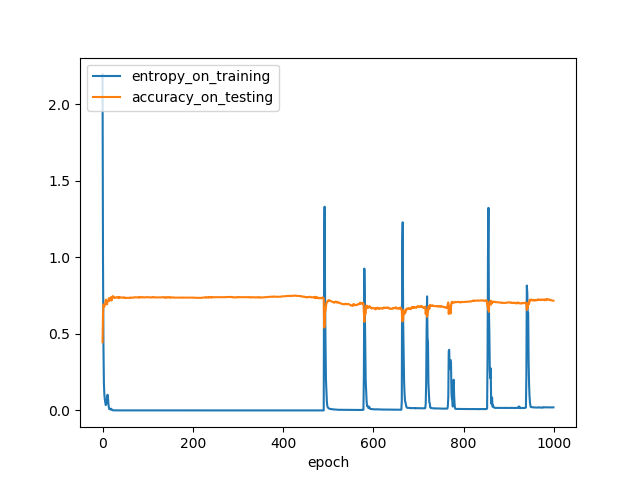
 

Figure 41: GRU char with dropout Figure 42: GRU char without dropout

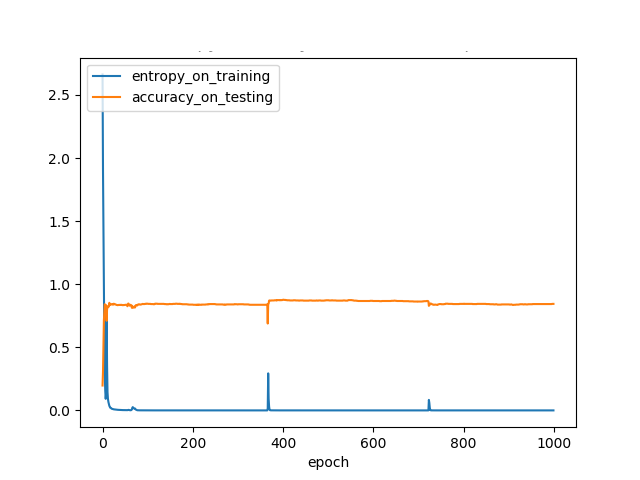
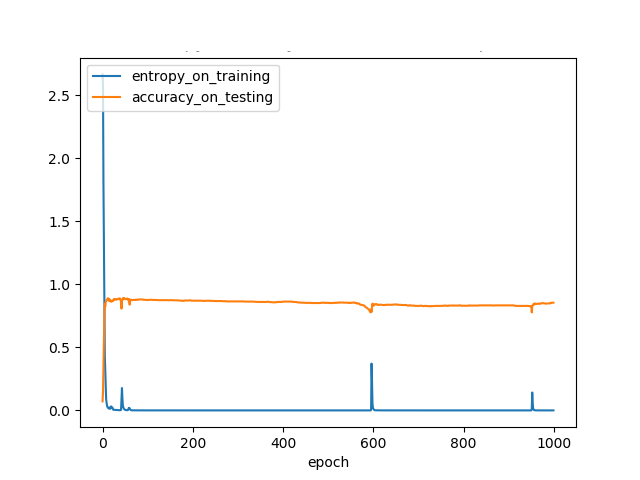
 

Figure 43: GRU word with dropout Figure 44: GRU word without dropout

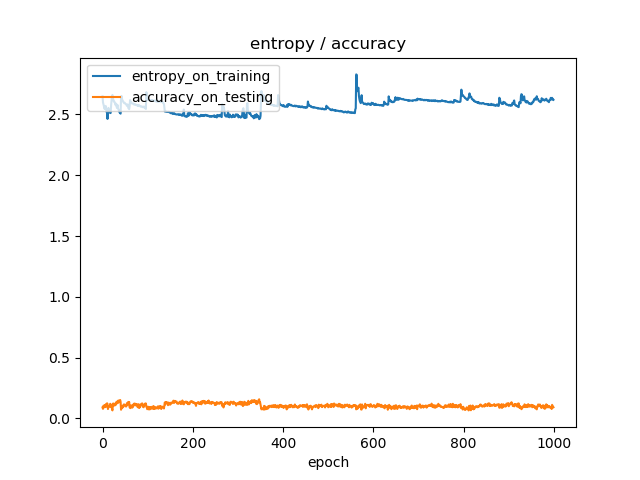
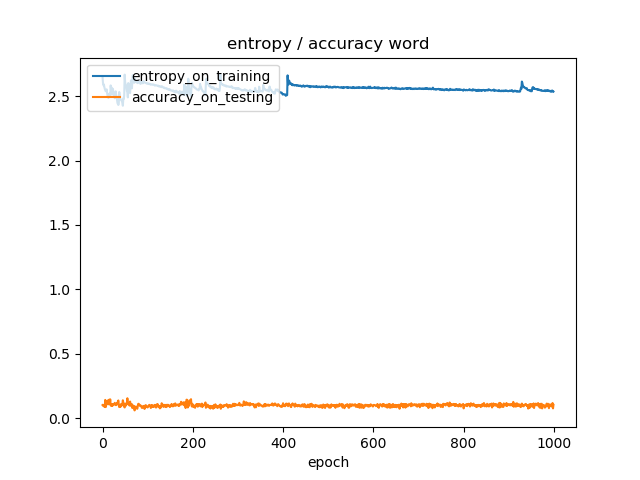
 

Figure 45: RNN char with dropout Figure 46: RNN char without dropout

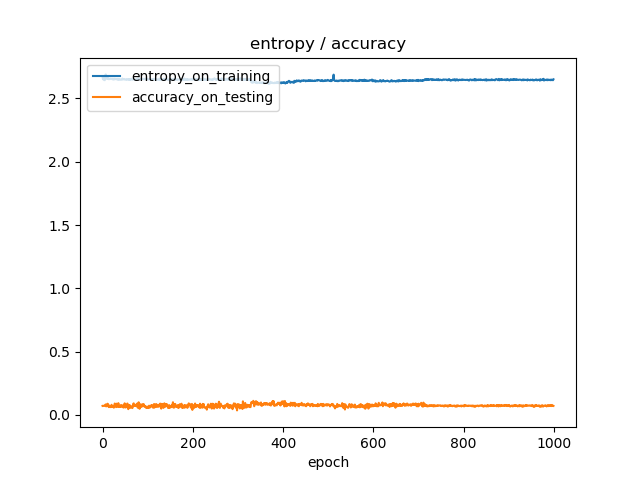
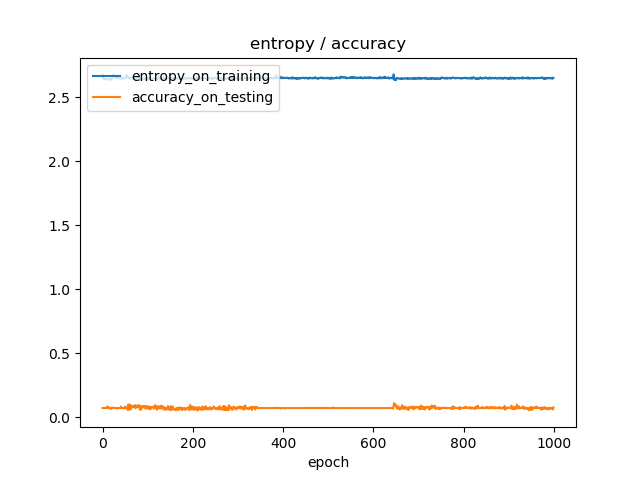
 

Figure 47: RNN word with dropout Figure 48: RNN word without dropout

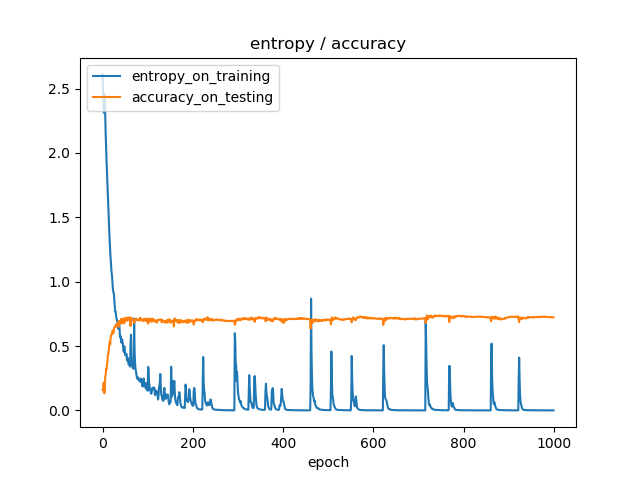
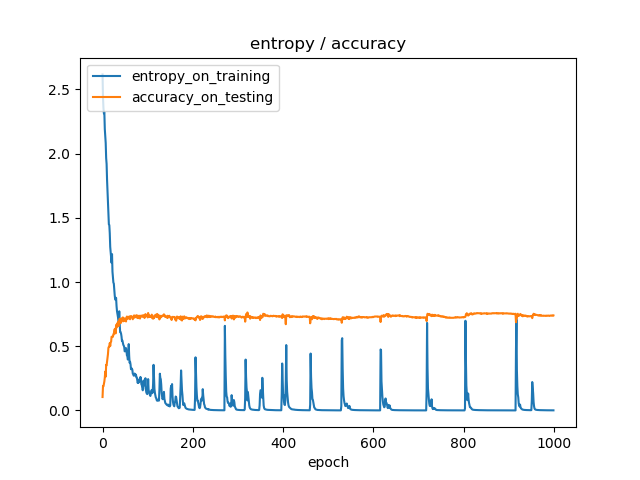
 

Figure 49: LSTM char with dropout Figure 50: LSTM char without dropout

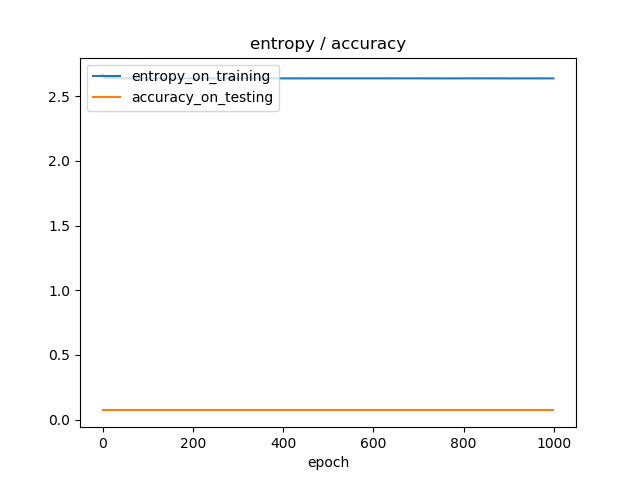
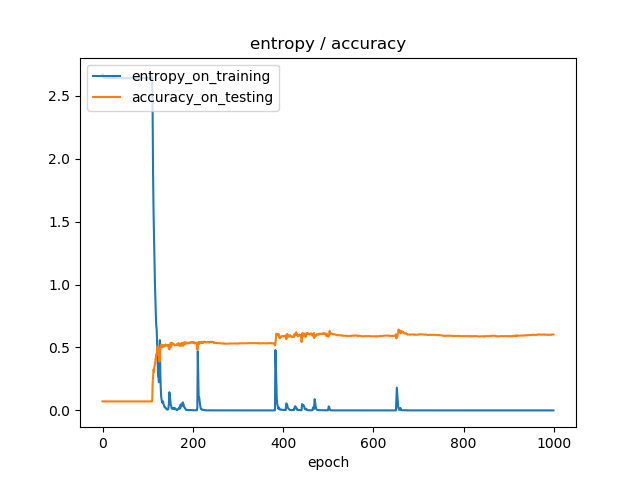
 

Figure 51: LSTM word with dropout Figure 52: LSTM word without dropout

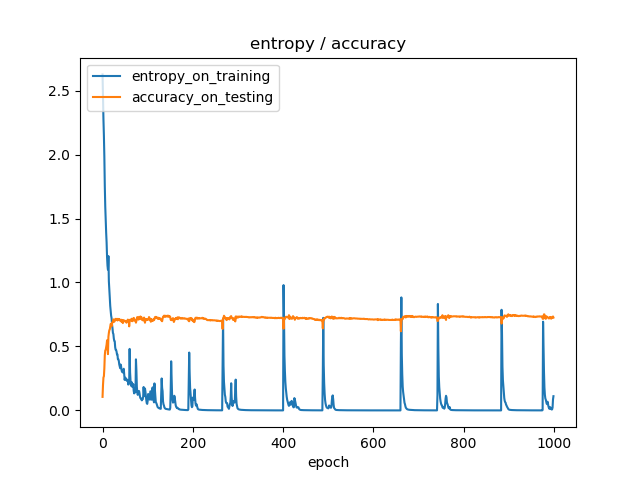
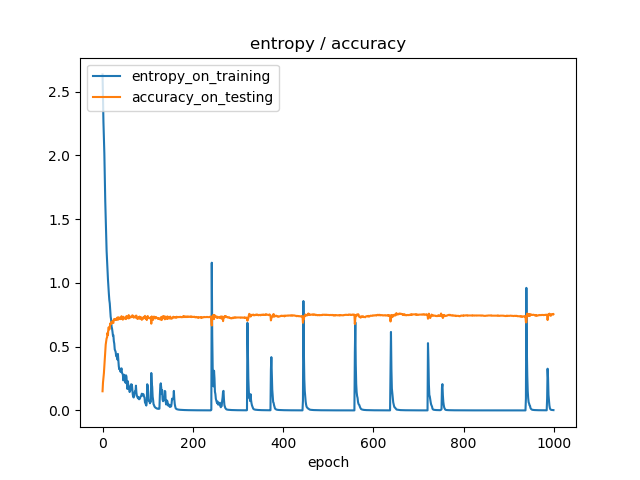
 

Figure 53: 2GRU char with dropout Figure 54: 2GRU char without dropout

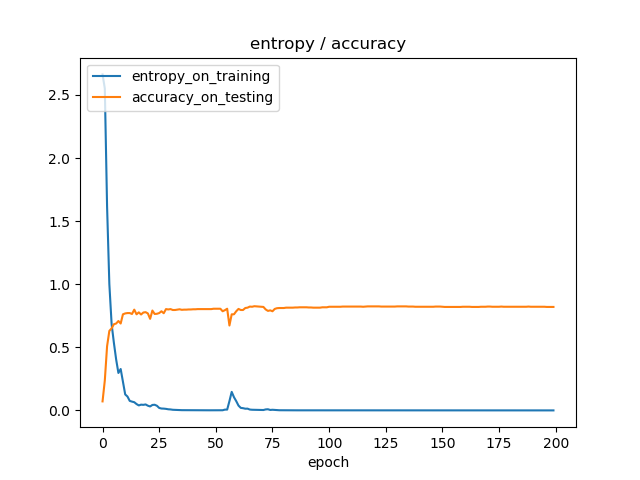
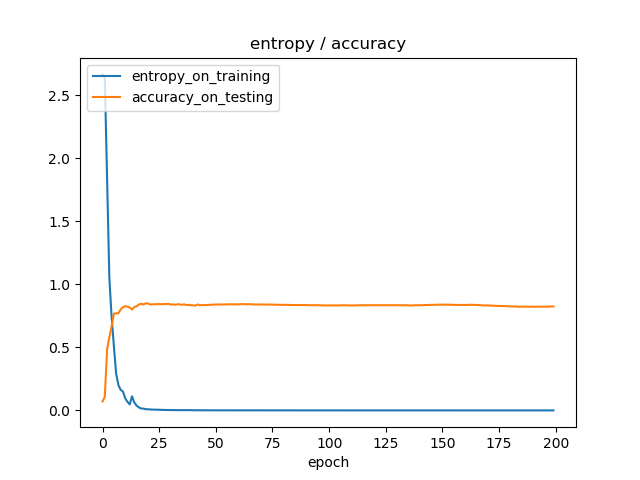
 

Figure 55: 2GRU word with dropout Figure 56: 2GRU word without dropout

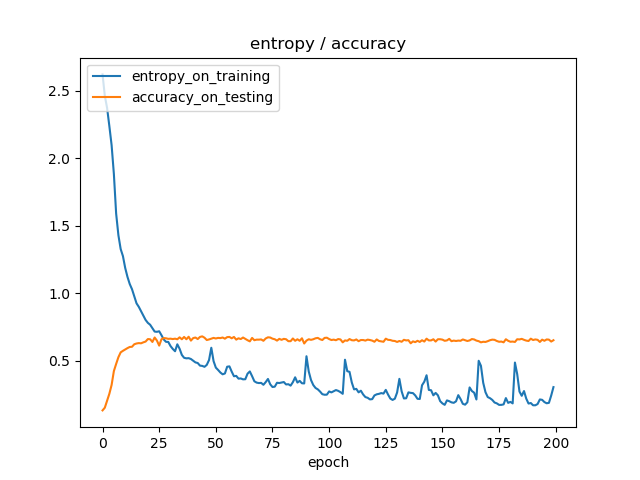
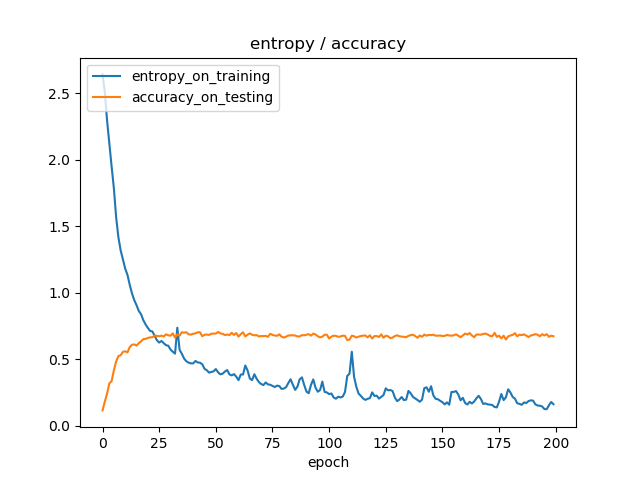
 

Figure 57: GRU clip char with dropout Figure 58: GRU clip char without dropout

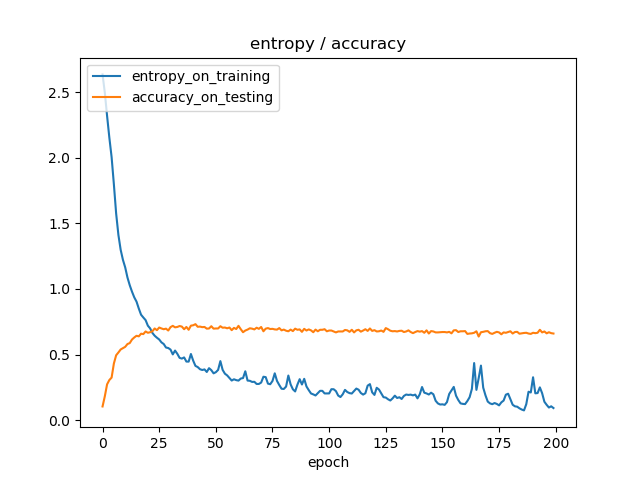
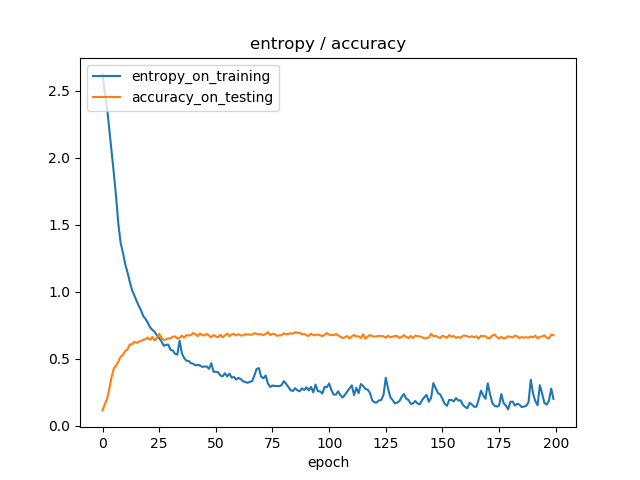
 

Figure 59: GRU clip word with dropout Figure60: GRU clip word without dropout

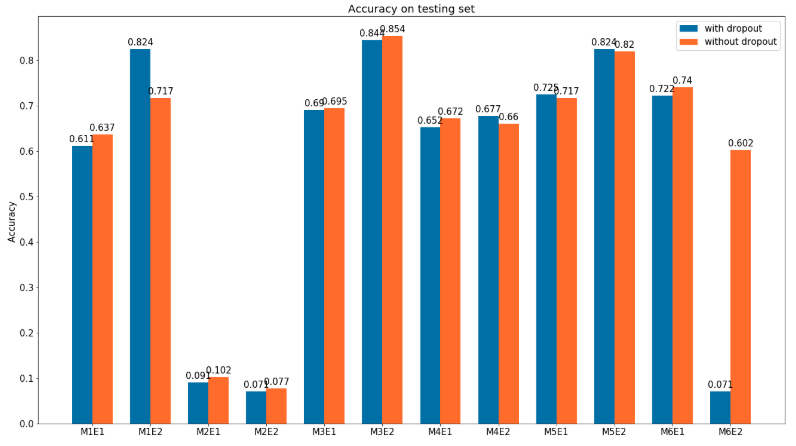


Figure 61: Accuracy on testing set for all models

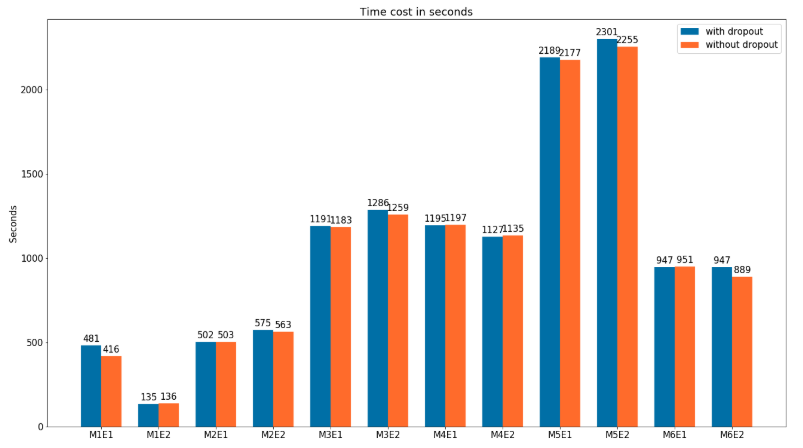


Figure 62: Time of training for all models

Testing Accuracy Results:

GRU+word (0.85) > CNN+word (0.76) > GRU+char (0.69) > GRU+char (0.62)

Word embedding makes more sense than character embedding since it contains the word meaning, leading to higher accuracy on both CNN and GRU model.

Running Time:

CNN+word (2:15) < CNN+char (7:35) > GRU+char (19:47) > GRU+word (21:12)

CNN generally runs faster than GRU since there is no relationship between pixels and they all be in parallel, which GRU are more dependent on previous data and thus slower.

Dropout:

By adding the dropout layer, not much accuracy and time difference are shown.

Replace GRU with vanilla RNN:

In this case, the vanilla RNN accuracy is always low. Gradient vanishing might be one of the reasons because entropy is large but no accuracy improvements.

Replace GRU with LSTM:

In this case, GRU has better accuracy than LSTM overall but cost more time.

Replace GRU with two layers:

In this case, two layers of GRU has better accuracy than one layer overall but cost more time. Actually many modern NLP architecture gets benefits from stacking multiple RNN above, which is easier to catch the previous information.

Add gradient clipping on GRU:

In this case, the clipping does not help on the accuracy. The one without clipping has better accuracy than the one with, and share the similar time.

**Conclusion**

For image problem, CNN is one of the common ways to handle it. By testing the combination of different number and size of filters and pooling, and different gradient optimizer, the hyperparameters get tuned. For text problem, CNN and RNN are both popular. CNN performs like a sliding window and capture neighbour information which RNN takes directional information. Different combination of embeddings and models are tested out to find the accurate and timeless one.