Neural Network

Lucas Geurtjens | s5132841 | 14/05/2019

Contents

[Manual Calculations 3](#_Toc8766449)

[Verifying Correctness of Manual Calculations 3](#_Toc8766450)

[Algorithm Overview 3](#_Toc8766451)

[Test Data Results 4](#_Toc8766452)

[Alternate cost function 9](#_Toc8766453)

[Appendix 10](#_Toc8766454)

# Manual Calculations

For raw manual calculations, please refer to appendix *Figure 1*. After a complete forward pass and backpropagation for 1 epoch in a mini batch of 2, the weights were updated to as seen in *Table 1*.

Table : Manual Calculation Weight Updates

|  |  |  |
| --- | --- | --- |
| W | Original Weight | New Weight |
| 1 | 0.1 | 0.099929 |
| 2 | 0.2 | 0.199997 |
| 3 | 0.1 | 0.099958 |
| 4 | 0.1 | 0.099972 |
| 5 | 0.1 | 0.99309 |
| 6 | 0.1 | 0.099171 |
| 7 | 0.1 | 0.099306 |
| 8 | 0.2 | 0.199167 |

\*Bias remained at 0.1 as it was not updated (hence weights 9, 10, 11 and 12 were excluded)

# Verifying Correctness of Manual Calculations

To ensure the neural network calculations are accurate, a cross reference has been made between the algorithm calculations and manual calculations as seen in *Table 2*. It was observed that the manual and algorithmic calculations for the first epoch were very similar (with some small differences possibly due to rounding or very minor errors in the handwritten calculations). This indicates that the algorithmic neural network should follow a valid process in its calculations for the future data sets to be used.

Table : Manual vs Algorithm New Weight Calculations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| W | Original Weights | Epoch 1 (Manual Calculation) | Epoch 1 (Algorithm) | Epoch 2 (Algorithm) | Epoch 3 (Algorithm) |
| 1 | 0.1 | 0.099929 | 0.09999292381459561 | 0.09998592491699507 | 0.09997900271500001 |
| 2 | 0.2 | 0.199997 | 0.1999892868242898 | 0.19997866970185157 | 0.19996814772105825 |
| 3 | 0.1 | 0.099958 | 0.09998906065180824 | 0.09997820950516348 | 0.09996744594356306 |
| 4 | 0.1 | 0.099972 | 0.09999859786088319 | 0.09999731132735014 | 0.09999613943115543 |
| 5 | 0.1 | 0.99309 | 0.09930886197388886 | 0.09862029820016521 | 0.09793430089602936 |
| 6 | 0.1 | 0.099171 | 0.09930570520953383 | 0.09861399492570555 | 0.09792486132556237 |
| 7 | 0.1 | 0.099306 | 0.09917137885954593 | 0.09834528380431987 | 0.09752170741326407 |
| 8 | 0.2 | 0.199167 | 0.19916741399056967 | 0.19833736693703355 | 0.19750985138826818 |

\*Bias remained at 0.1 for all epoch’s

# Algorithm Overview

The program uses the principles of forward pass and backpropagation (as used in the manual calculations) and applies them algorithmically. Firstly, all data is read in from the flat files provided by the user. Here, the training data samples were shuffled, such that there we not clusters of samples with the same target. From this, a set of small random weights are initialised. Next, the program loops though for a certain number of epochs to and trains with a given number of samples. For each mini batch item, the program performs a forward pass, then calculates the derivative of E Total over the weights. From this, the E final is calculated, and the weights are updated. The next batch is then looked at, until all the samples are exhausted. After this, the program then looks at the accuracy of the outputs, using a quadratic cost function. Here, a function for creating a prediction was to be implemented, however, due to time constraints was not yet fully developed. Another issue incurred with the algorithm was its significantly slow speed for loading in data. This significantly restricted the sample size that could be used and caused for smaller sample sizes to be used during experimentation.

# Test Data Results

After training the data set, using the test data, a line graph of Epoch’s vs Accuracy was created. It should be note that due to significant time costs, while training the model, a default epoch of 10 and training data sample size of 500 was used instead of the recommended epoch of 30 and complete sample size of 50000. Hence, with a smaller sample size and epoch number, the accuracy of weight predictions may be smaller.

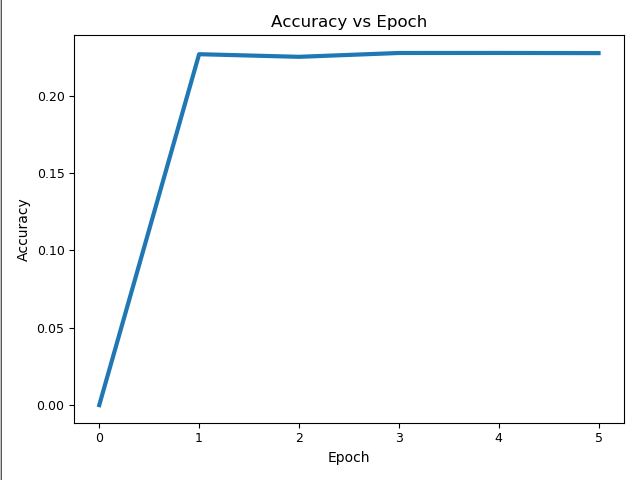
Epoch = 5

Sample size = 250

Batch Size = 20

Learning Rate = 3

Bias = 0.001



As seen above, the maximum accuracy achieved was approx. 0.25.

In making the predictions of the test samples, due to the prediction making functionality not being fully implemented, only a blank prediction document “PredictTestY.csv.gz” was created. Nevertheless, it’s contents would have contained integer values from 0 to 9 for each test sample. This would have been found by finding the output with the highest value and setting the index it was into its integer number equivalent.

Parameter Experimentation

Learning Rates

Firstly, the learning rates 0.001, 0.1, 1.0, 10 and 100 were experimented with:

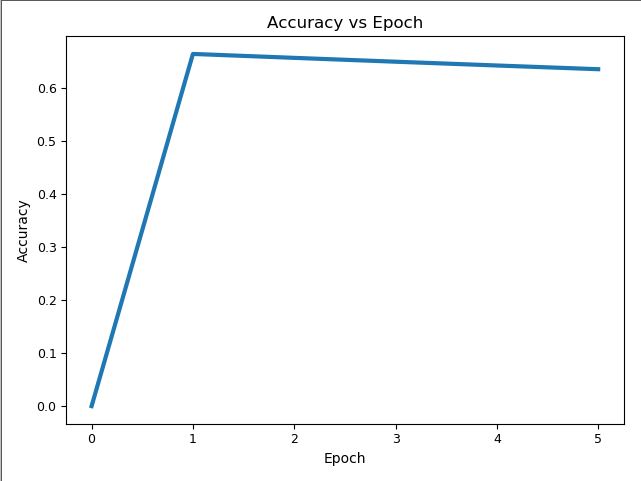


Figure : Learning Rate 0.001

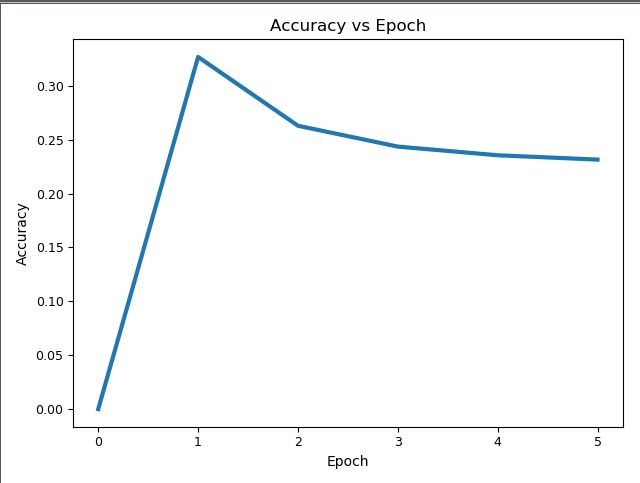


Figure : Learning Rate 0.1

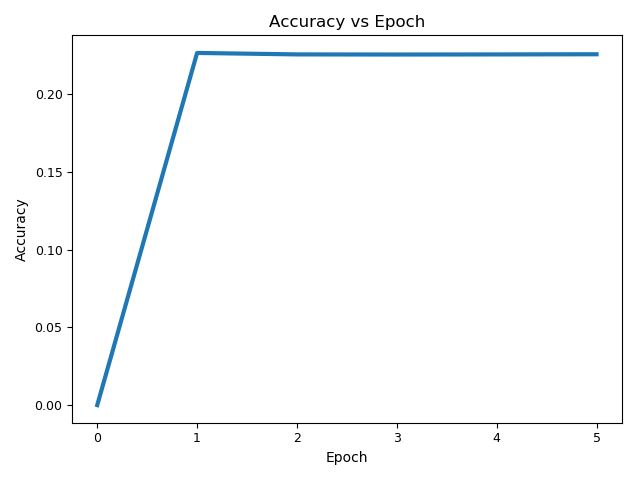


Figure : Learning Rate 1

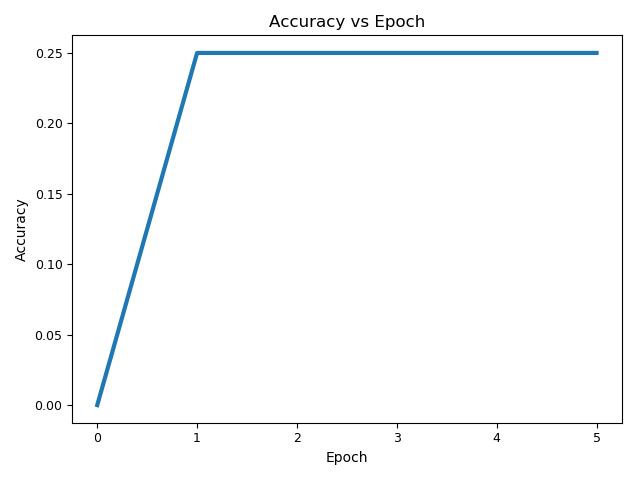


Figure : Learning Rate 10

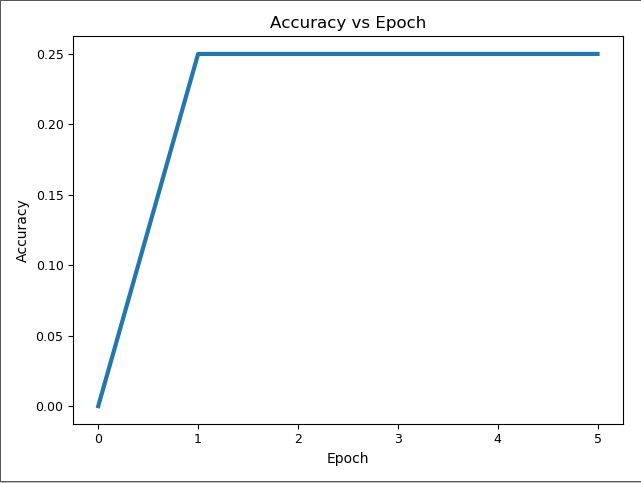


Figure : Learning Rate 100

It was observed that by using a higher learning rate, accuracy became stagnated. This suggests that by using a high learning rate, the gradient decent is too quick and not gradual enough, such that it is unable to meet the local minimum entirely. As well, it indicates that for our specific parameters, a lower learning rate of 0.001 may provide better accuracy. Here, the maximum accuracy attained was approx. 0.7 accuracy with the 0.001 learning rate.

Mini batches

Furthermore, the mini batch sizes 1, 5, 10, 20 and 100 were experimented with:

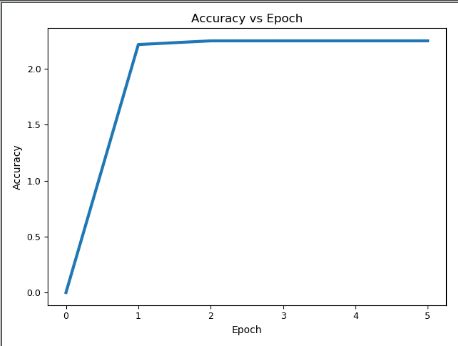


Figure : Batch Size 1

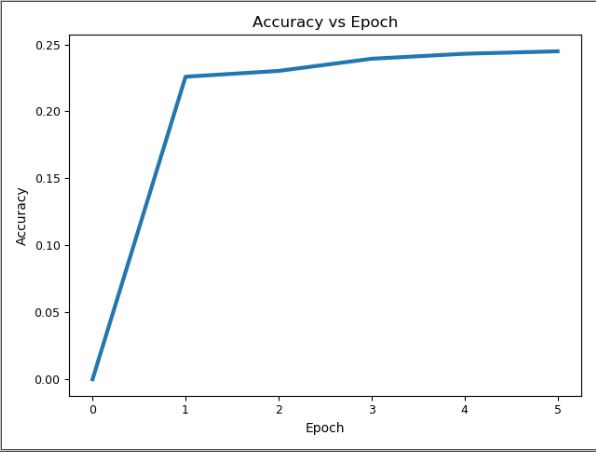


Figure : Batch Size 5

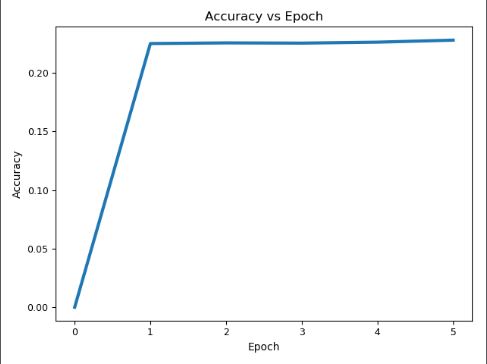


Figure : Batch Size 10

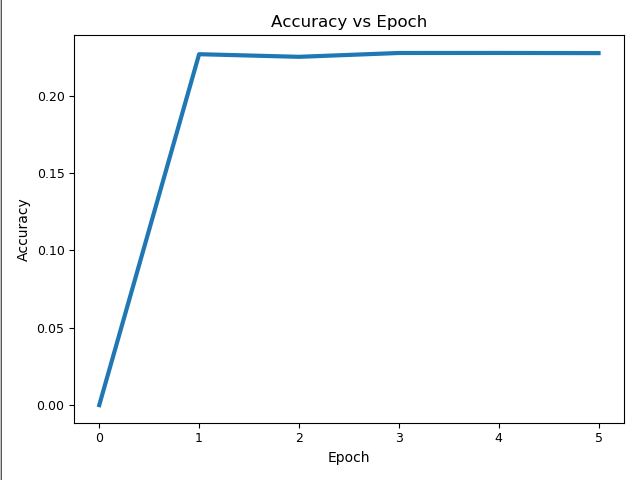


Figure : Batch Size 20

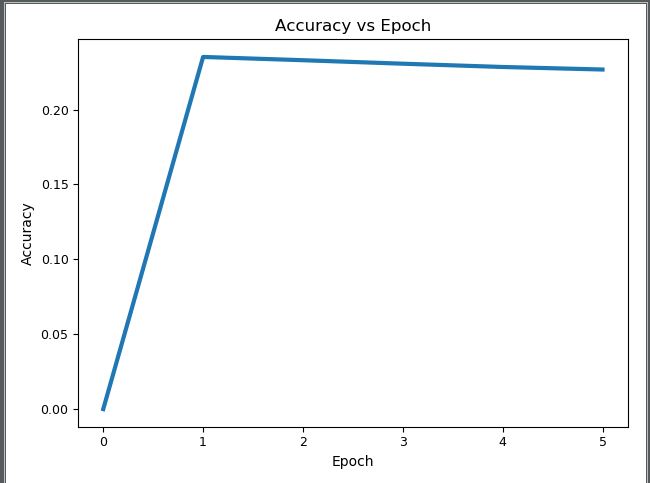


Figure : Batch Size 100

Here it was observed that smaller mini batch sizes produced higher accuracy. This can be seen in with the batch size of 1, in which a maximum accuracy of 2 was found. This may also indicate a possible flaw in the calculation of the accuracy since an accuracy of 2 is impossible theoretically.

Epoch

Finally, the epoch values 1, 3 and 5 were observed and compared:

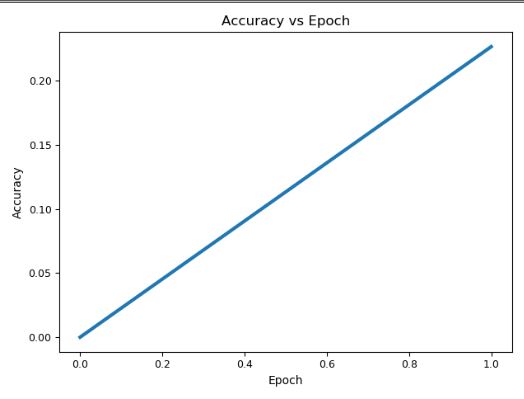
**

Figure : Epoch 1

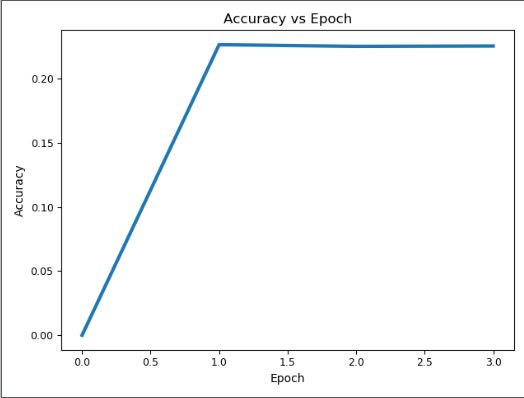


Figure : Epoch 3

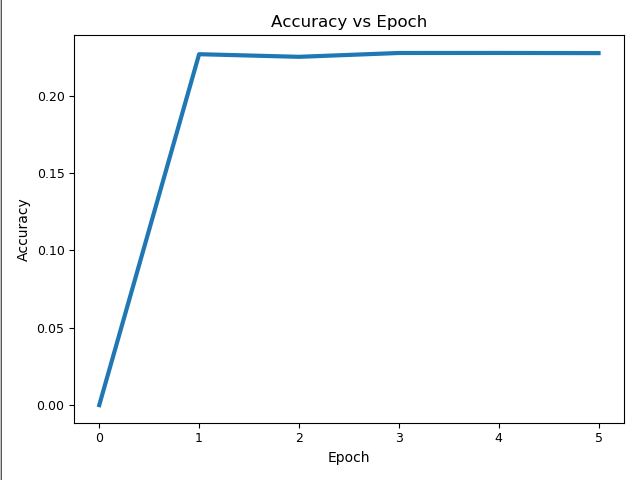


Figure : Epoch 5

By increasing the number of epochs, it was observed that the accuracy did not increase, giving a maximum accuracy of 0.25. This may be due to a relatively low number of epochs being compared, as more training should increase the accuracy (unless overfitting has occurred).

# Alternate cost function

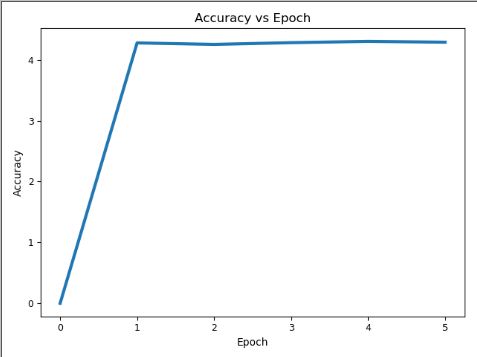
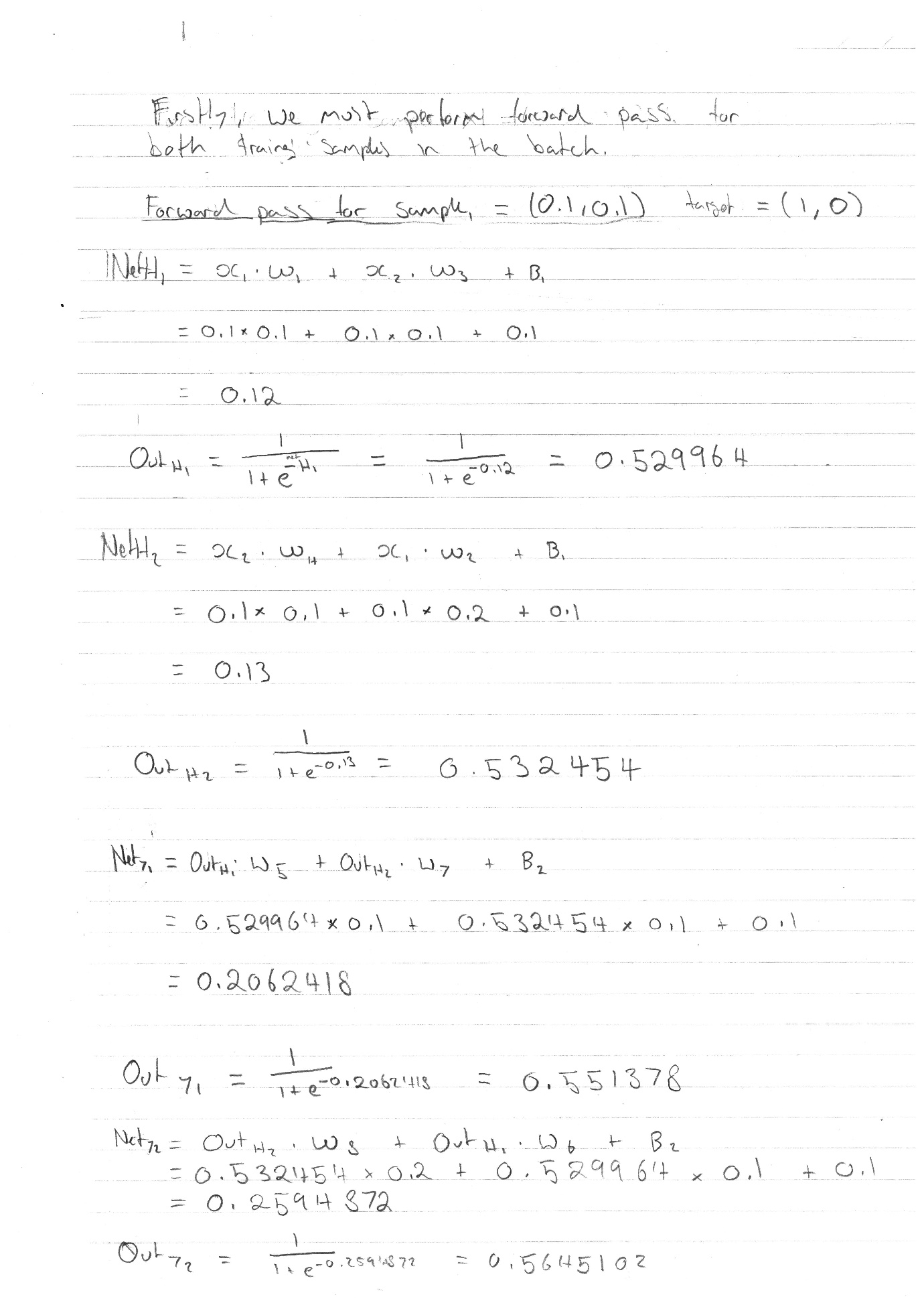


Figure : Cross Entropy Cost

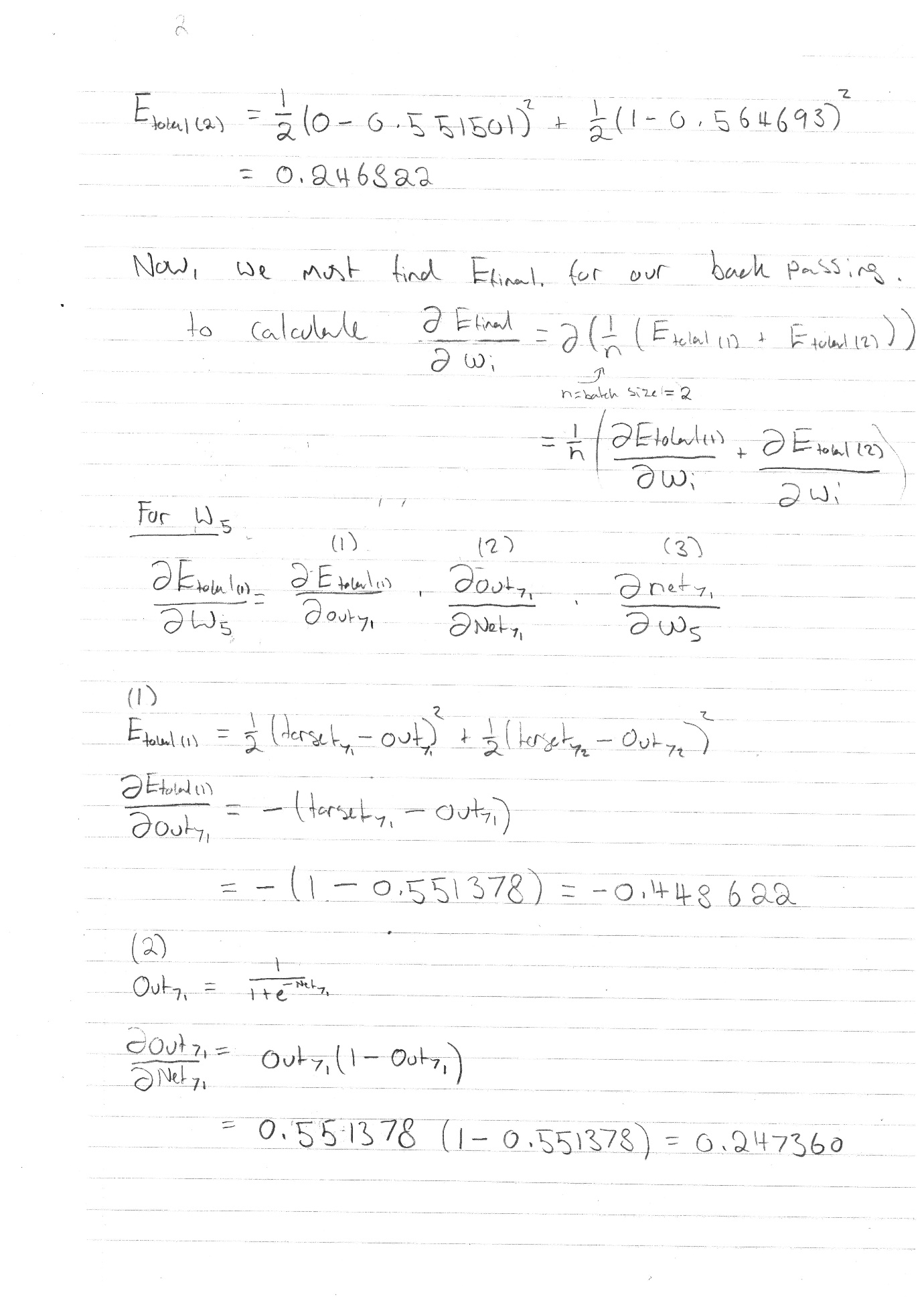
After implementing the cross-entropy cost function in place of the quadratic cost function, it was observed that accuracy significantly increased (despite indicating possible error in the calculation). Here a maximum accuracy of 4 was attained, compared to the maximum of 0.25 attained by the quadratic cost function using the same parameters. This indicates that the cross-entropy cost function provides a less conservative way of calculating accuracy.

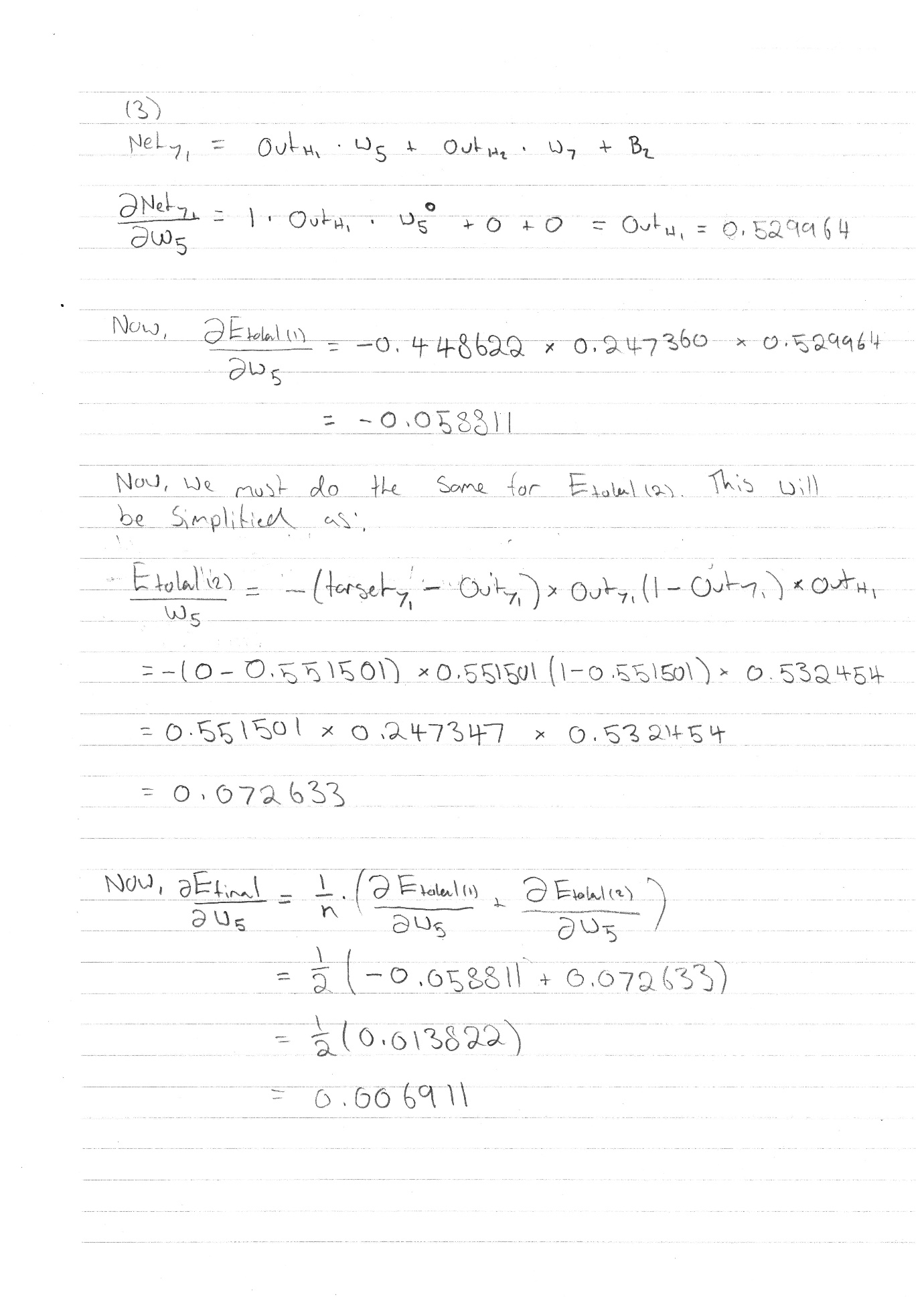
# Appendix

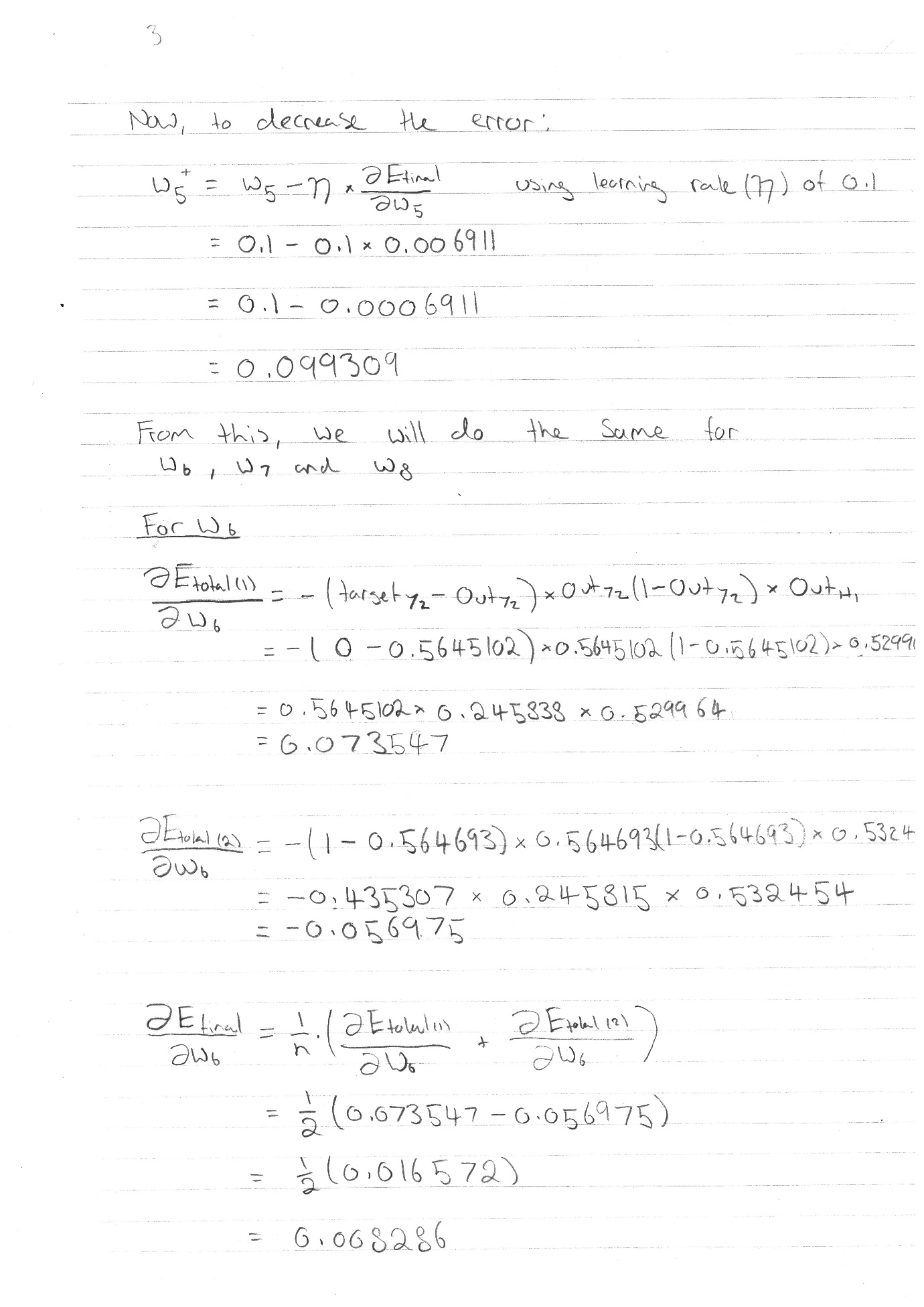
*Figure 1: Manual Neural Network Calculations*

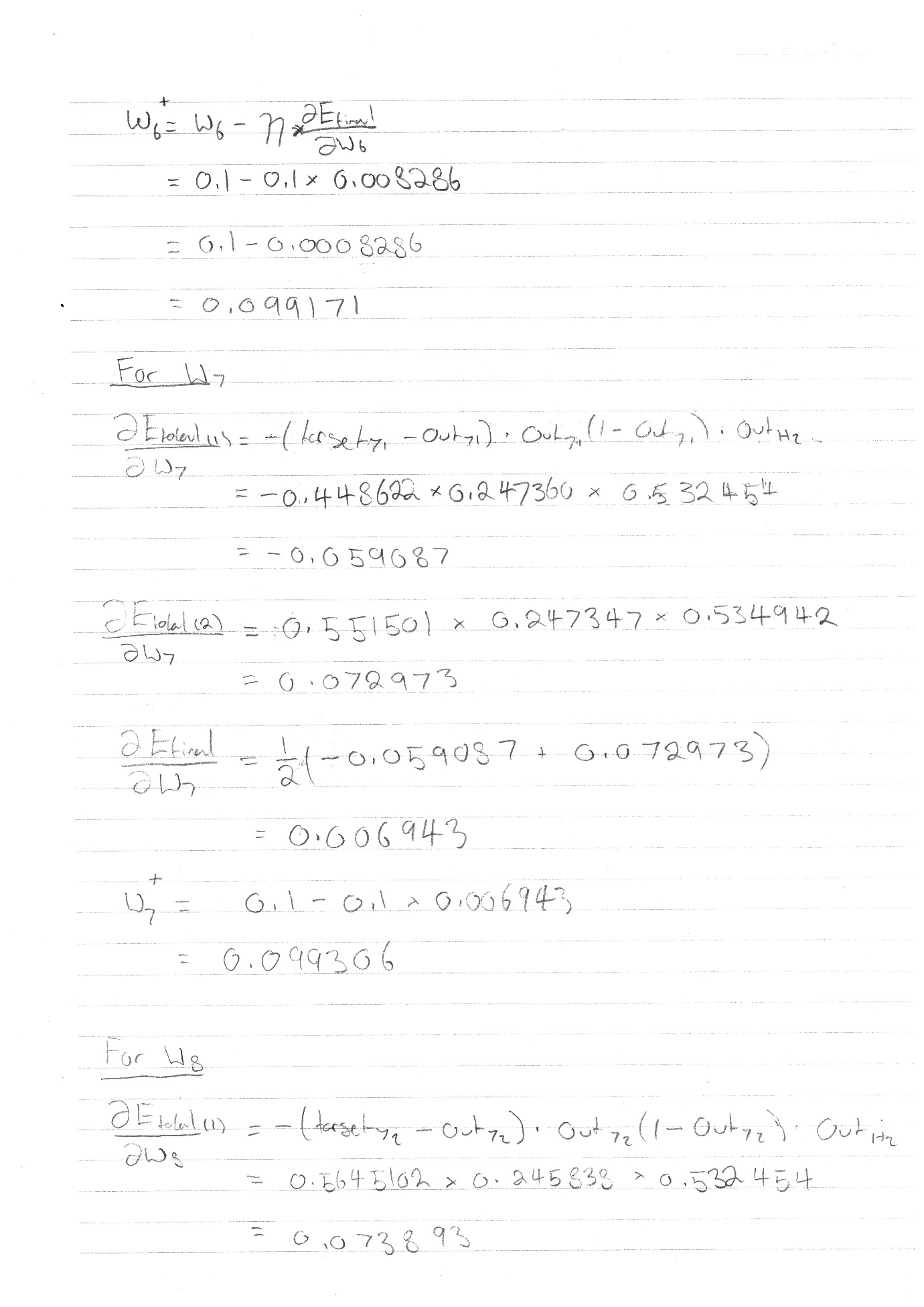
**

**

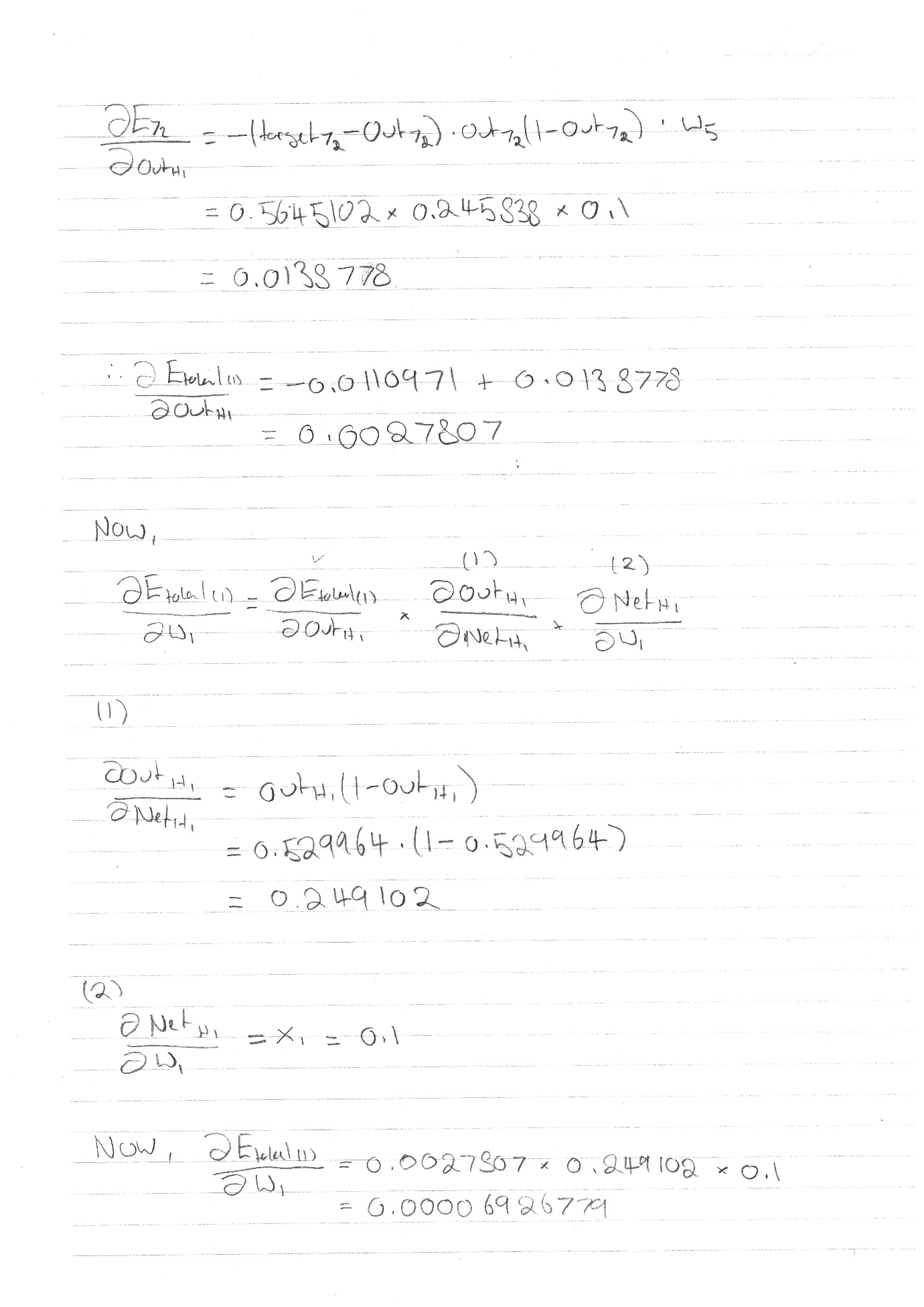
**

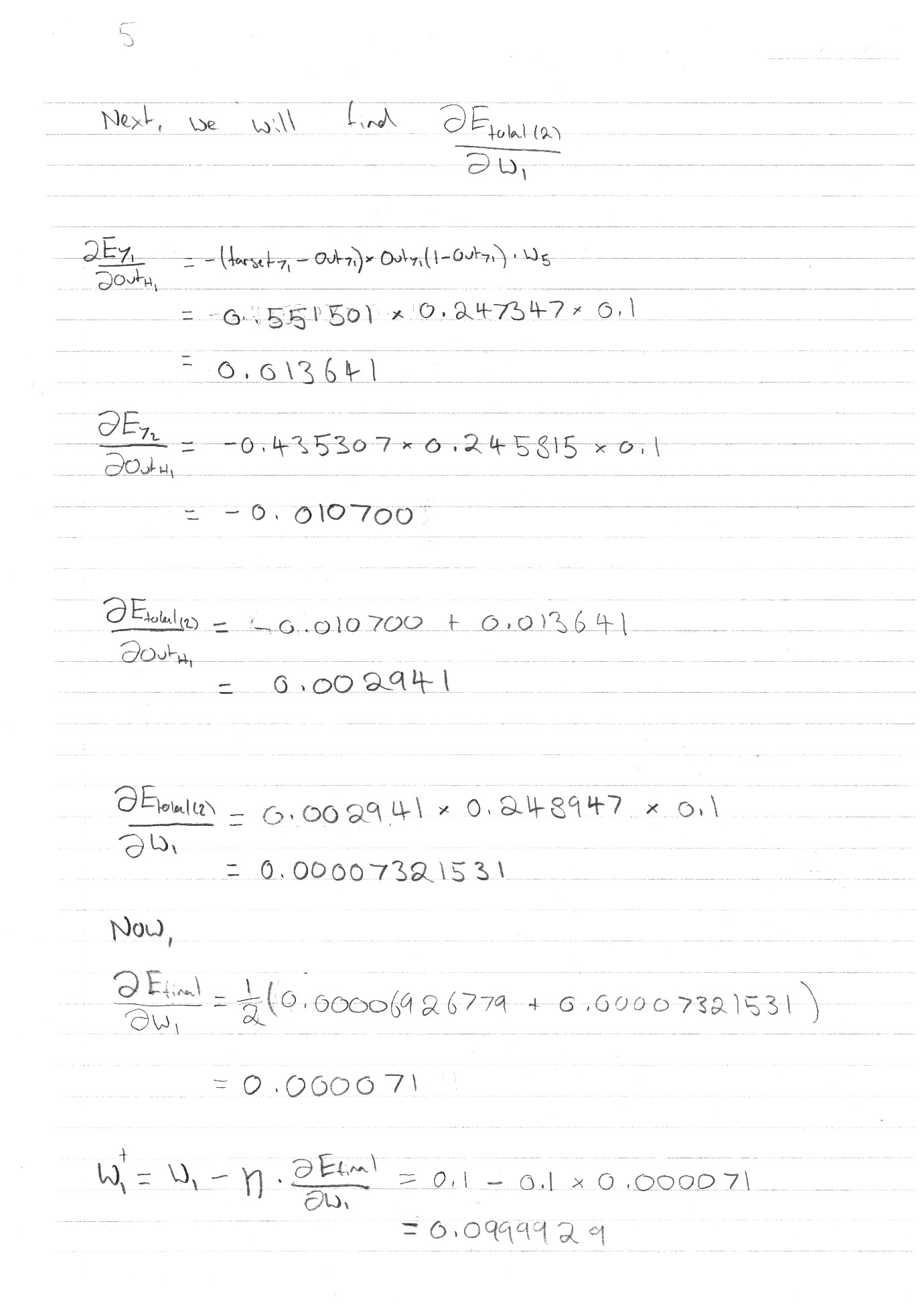
**

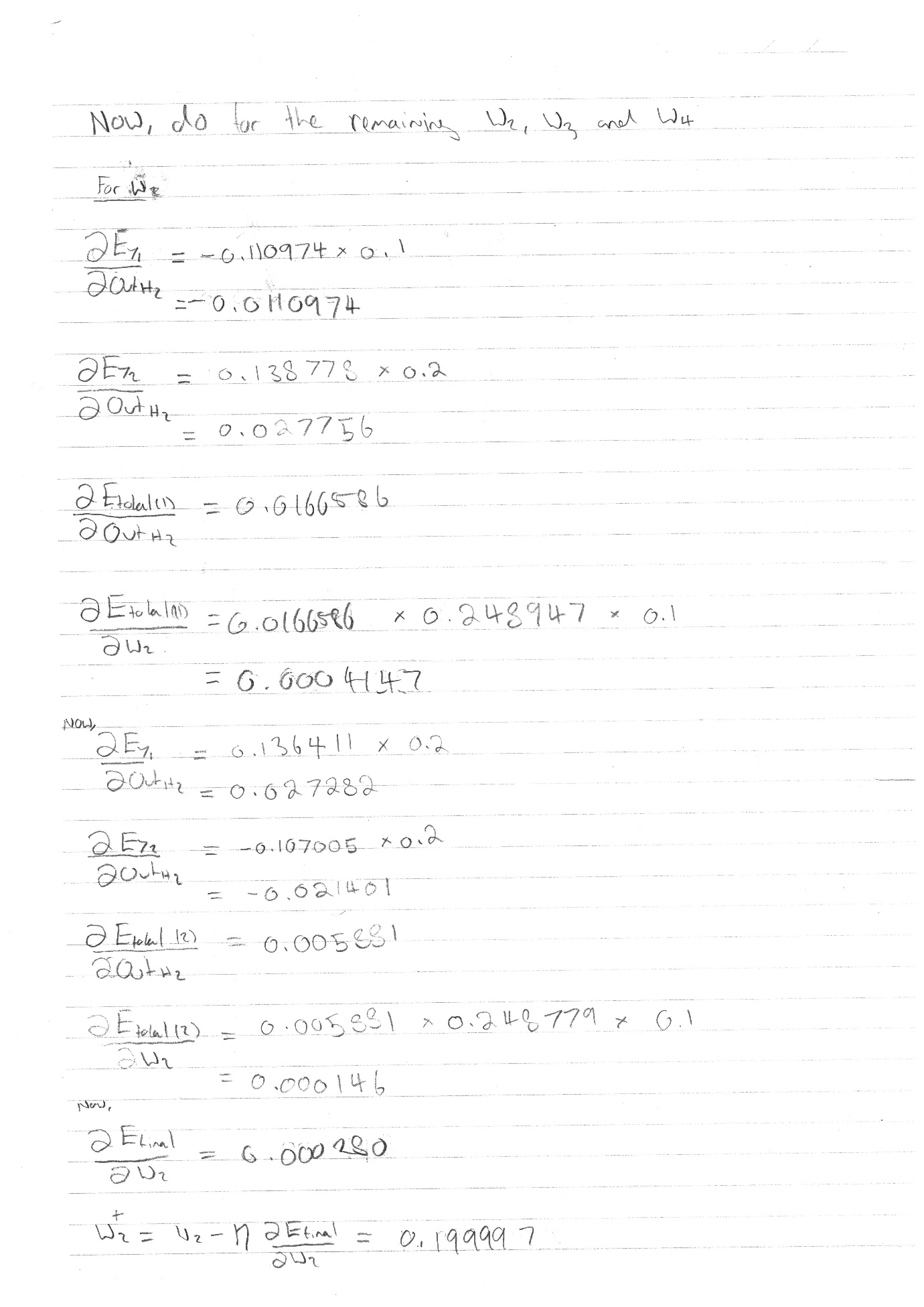
**

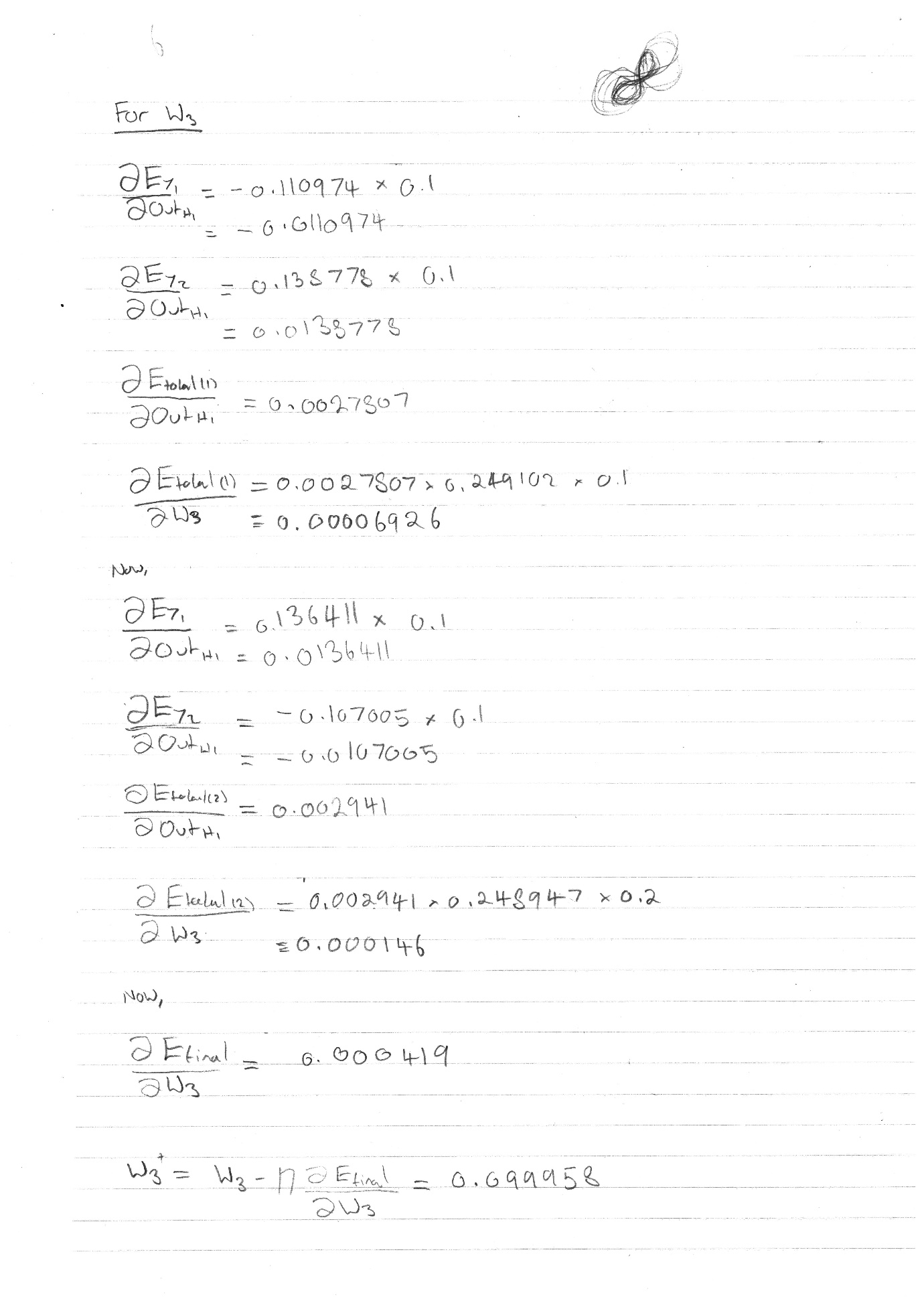
**

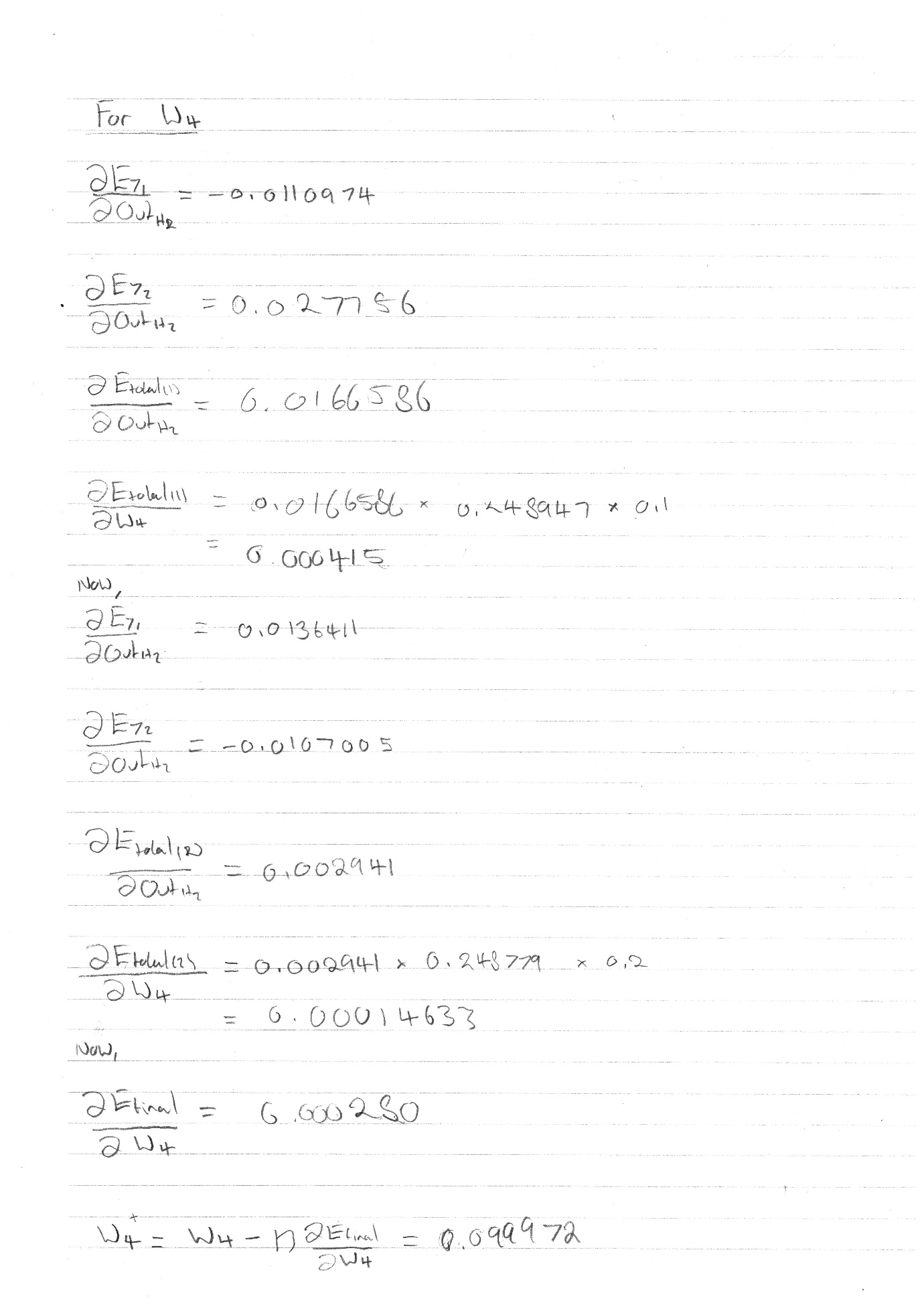
**

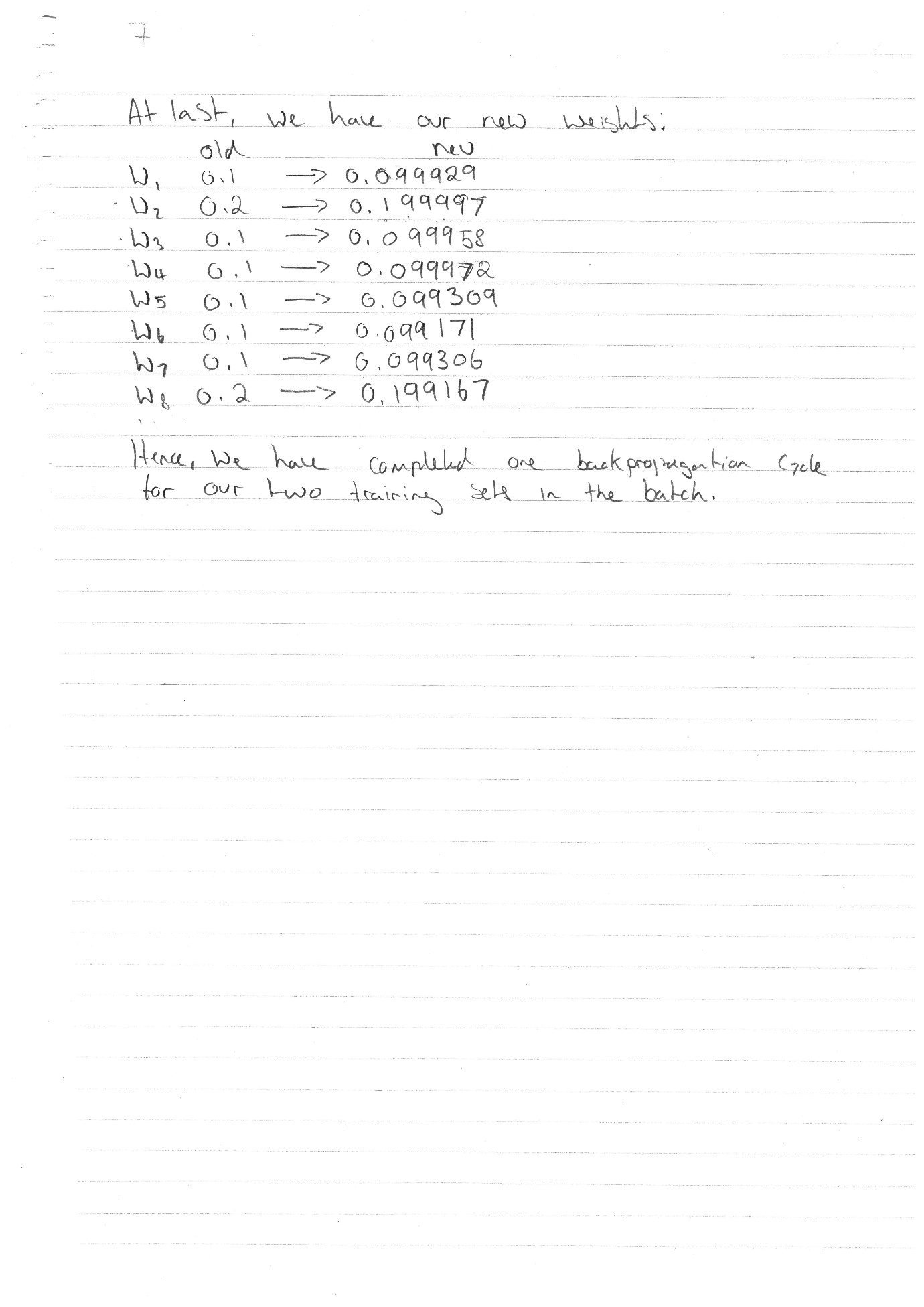
**

**

**

**

**

**