

Individual Assignment questions

Ioannis Konstantakopoulos 0960047

1) Describe the principles of overfitting and how dropout can reduce this (Question 1, 5 points)

Overfitting is the modeling error that happens when your model predictions are closely fit the limited training data. Thus, a future prediction to an unknown testing dataset is inaccurate with substantial errors reducing the power of the model. A lot of techniques there been created to stop the training as soon as the performance of the validation(training) set starts to get worse. One of these techniques is dropout.

With Dropout some units or neurons has been dropped out of the neural network. By dropping out we mean that temporarily remove a unit along training process with all its connection as we can see in figure 1.

Now Dropout most of the times is random. A fixed independent probability p is been retained with other units, where p can be chosen using a validation set or simply be 0.5. For input units the optimal value for p is 1.

Using dropout to a neural network means that a “thinned” sample is been created from it. Containing all the units that were left from the dropout. A neural network with n units can be seen as 2^n possible thinned neural networks. All these networks they share weights, so the overall calculation of the parameters is $O(n^2)$. At test time its not feasible to explicitly average the predictions from that many models. Just imagine the computation power that needed to address this really big number of epochs. However a simple approximate average method works well in the practice. The idea is to use only one neural network at test set without dropout. The weights of this network are scaled down versions of the trained weights. So, if the unit has probability p in training the outgoing weight are multiplied by p at test time. This ensures that all the units hidden or not they will all have the same actual output in the test time. Using this technique in training time and the approximate average method at test time leads to significantly lower generalization error and thus prevent overfitting as well.

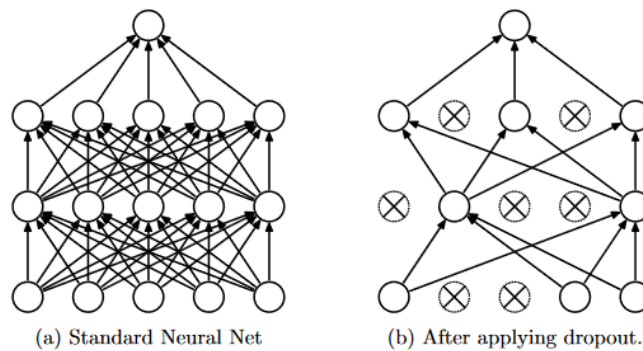


Figure 1: Dropout Neural Net Model. **Left:** A standard neural net with 2 hidden layers. **Right:** An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Source: <https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf>

2) Write a short (~500 word) summary of the experimental approach and results. (Question 2, 10 points)

The main research goal of the article is to create a model approach that can quantitatively accurately recognize objects, like humans. The human ability to recognize objects supported by a network of hierarchically stages starting from the lowest V1 to higher level like V4, IT.

Firstly, an image set containing natural categories with various object position, scale and pose has been created. To avoid correlation, the objects have been put in random natural scenes. They collected responses from 168 inferior temporal (IT) neurons to each image, using multiple electrode arrays. To measure the categorization performance and to assess the IT neural predictivity of each model using high-throughput computational methods. The validation of the models has been done using SVM through cross-validating to test accuracy.

Models were created from a large parameter space of convolutional neural networks (CNN) like the biological hierarchical process concept. Each convolution layer has been stuck up to construct deep neural networks.

Each model is defined by 57 parameters controlling the number, parameters, activation thresholds and pooling of each layer. Weight of each one to three ranged layers were selected randomly. There were three ways to evaluate and choose the best model. First, random sampling, second optimization for performance on the categorization task and finally by optimization directly for IT neural predictivity. To find the recognition difficulty they performed the evaluation from low to high in the three levels of object variation. As a comparison human performance was measured, the results were the V4 matches human IT performance at low levels but not closely to good at higher.

Although the results were good for the low level, the three layer hierarchical CNNs they are limited in higher variations tasks. One of the first thoughts is to expand the model and combine it with deeper CNN networks. To solve this, they used hierarchical modular optimization (HMO). The main advantage of this procedure is to create a simple hypothesis

for high performing combinations of operating hierarchical architectures can effectively create and hierarchically merge without specifying subtasks ahead of time. HMO algorithm is like boosting procedure with hyper-parameter optimization.

Given that HMO had better results the next step is to measure its IT predictivity from top and each of the three between layers. The results indicate that high IT predictivity not always acquired from category selectivity and it exists a the non-categorical structure. Cortical area V4 is the dominant cortical input to IT and V4 is less categorical than IT. The HMO models results showed that V4 contributes to the intermediate layer of the model and the top layer is the efficient level of IT.

The results indicates that the bottom-up methodology with a top-down aspect of describing IT as the result of developmental process using the correct model can be used to create quantitative predictive models of neural processing.

3) Play around with these settings and see how they affect your ability to learn classification of different data sets. Write down what you found and how you interpret the effects of these settings. This question is intentionally open to allow you to explore the process. (Question 3, 8 points)

The site is an interactive visualization the behavior of neural network. Having everything combined in a simple and understanding way helps the reader to deep deeper into the parameters and how they define and change the whole accuracy and epoch training. Only classification problems will be discussed.

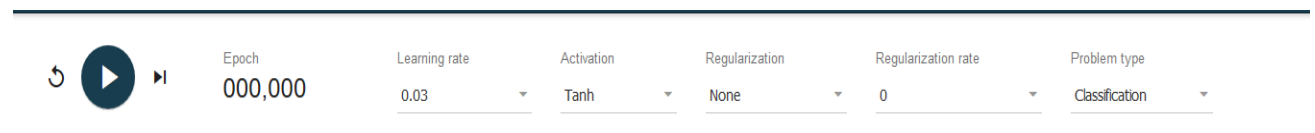


Figure 1

On the top we can see figure 1 is the menu from which we can select parameters such as

- Epoch: refers to one full iteration through all training set, thus the number of epoch increasing each time training is conducted.

- Learning rate indicated the speed of model adaptation to the problem usually range between 0-1 as learning rate bigger than 1 can make the steps large and the model unstable
- Activation functions are the mathematical equations the determine each neuron in the network if it should be activated or not. The most common one is the ReLU function which been activated for values $y \geq 0$ where y is output value.
- Regularization is used to prevent overfitting. *As regularization increases the weights of strong connections to make the pattern classification sharper. Here we have L1 which will make the weights of selected values bigger and the weights of non selected values smaller. Also L2 will control the weight values corresponding to the level of correlation
- Regularization Rate makes the weights limited in range

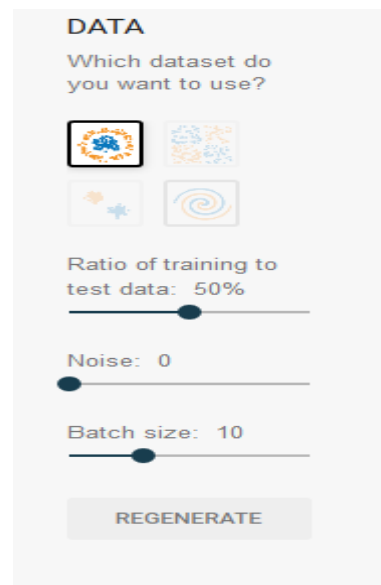
Left on the site as the Figure 2 shows, we can select the type of data circle, spiral ,gaussian and exclusive which will be seen in the output of the neural network.

The colors of the dots represent the impact Blue is positive up to +1 and orange is negative up to -1

Also the ratio of training to test data can be controlled 50% means more dots in the output and 10% means less dots or data are being used.

As the noise increases the data pattern becomes more irregular. Smaller noise means the output dots are clearly distinguished. Otherwise they mixed up making it very challenging to classify.

The batch size determines the data amount to use for each training iteration.



DATA

Which dataset do you want to use?

Four dataset icons are shown: a circle of dots, a spiral, a gaussian distribution, and an exclusive set. The circle of dots icon is selected with a black border.

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

Figure 2

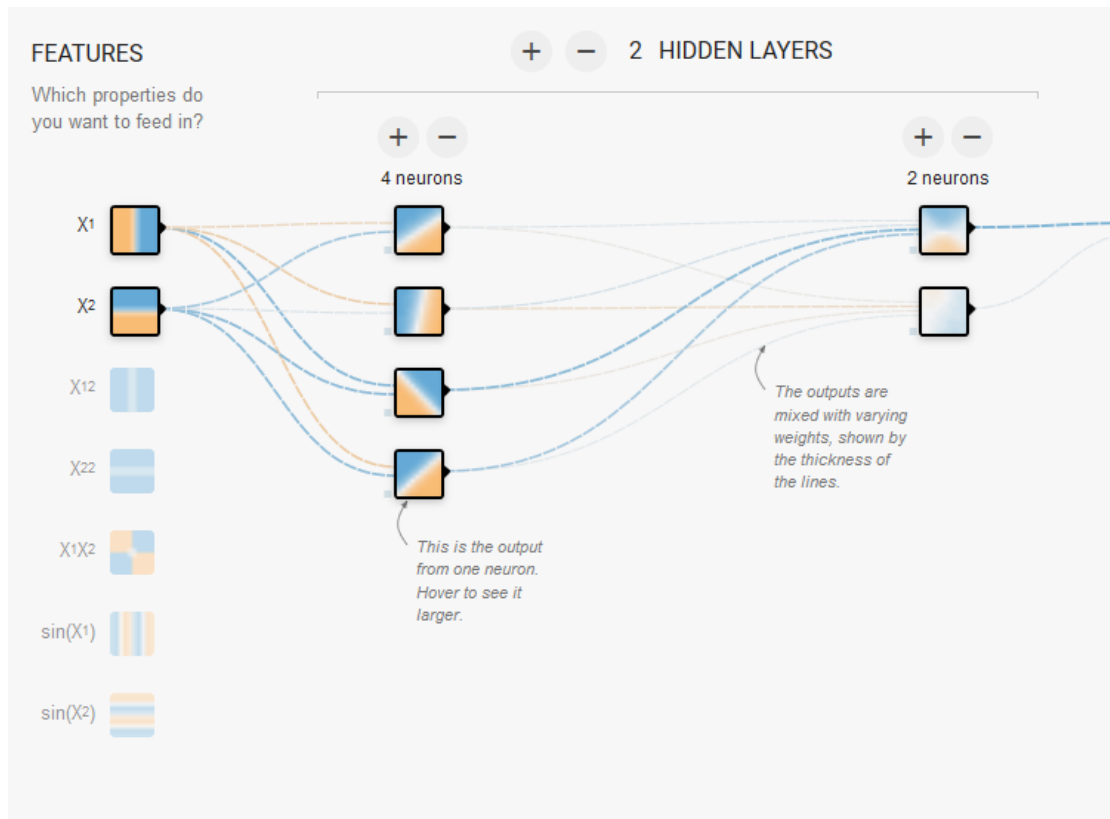


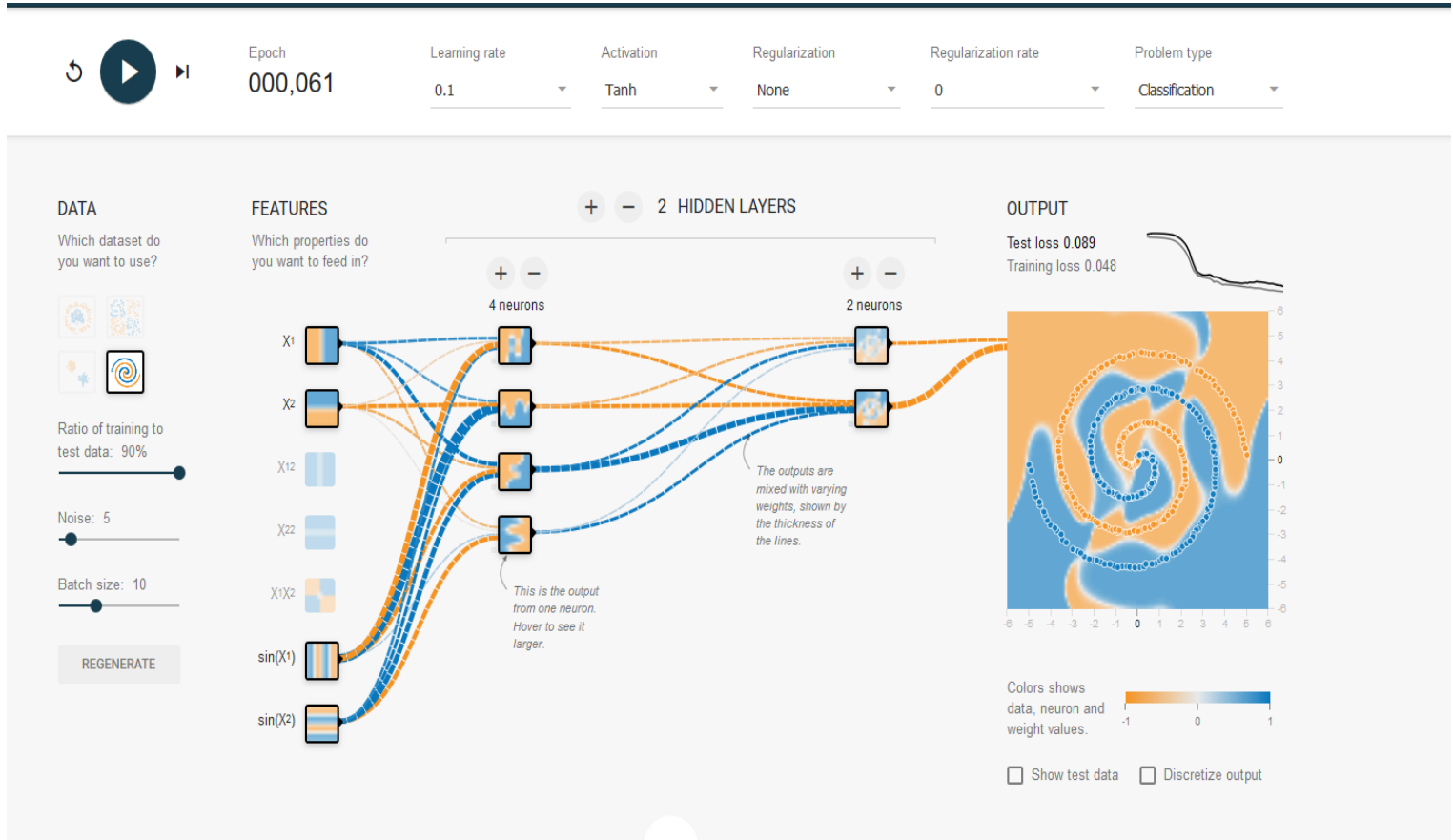
Figure 3

As Figure 3 indicates there are 7 features selection X^1 is the horizontal axis, X^2 is the vertical axis, squares of X^1 and X^2 also the product of X^1X^2 and the sin of X^1 and sin of X^2 . The structure of the hidden layers can be up to 6 and the neurons can be up to 8. The selection of the hidden layer is based on the input size. Adding many layers not always working better as the model need more computation power.

The neural network to minimizes the test loss and training loss as the epochs progress. Also in the lines are colored by the weights if they are blue close to 1 and positive weight, otherwise orange and negative weight.

4) What is the minimum you need in the network to classify the spiral shape with a test set loss of below 0.1? (Question 4, 7 points)

The data shape we need to classify is spiral shape, my first thought is to use except of the X_1 and X_2 the $\sin(X_1)$ and $\sin(X_2)$ because in mathematics that the way to transform from cartesian to spherical coordinates. Using only X_1 and X_2 will not work as good because we are using only cartesian lines. In my example I finalized the test loss below 0.1 around 0.089 also training loss is 0.048 and below after the 60th epoch and further also the model is not



overfitting. To achieve that I am going to make a step by step what settings and features I changed. Firstly, I choose learning rate of 0.1 if I went larger the model was fluctuating too much in each step and with smaller the training took way too many time. For activation function tanh had the best results, linear cant work as we can not classify spiral with lines. The ratio of training and test set was set at 90% it gave me the best accuracy faster. If I choose 80% it took much more epochs but here we cant to create the best possible. Noise can make the pattern irregular adding more noise than 5 my model needs way more time to converge. As I said before I used X_1, X_2 and $\sin(X_1)$ and $\sin(X_2)$ as features and 2 hidden layers the first one with 4 neurons and second one with 2 neurons. I choose 2 neurons for the second layer as it's a classification problem adding more neurons make the model more complex. For the first hidden layer the best converge had been achieved with 4 neurons. As final result the line in the output looks steady without any step fluctuation.

- 5) Explain the principle of backpropagation of error in plain English in about 500 words. This can be answered with minimal mathematical content and should be IN YOUR OWN WORDS. What is backpropagation trying to achieve, and how does it do so? (Question 5, 8 points)

Firstly, any neural network has been created by: neurons which are the circle shapes in Figure 1 and the Weights which are the rectangles in the Figure 1. Here I am going to create a neural network with three layers to justify what backpropagation does.

1) Input layer with two inputs neurons 2) One hidden layer with two neurons 3) Output layer with a single neuron.

Neural network starting weights are usually randomly generated, the weights (the lines connecting the neurons) show how important is each neuron to the output. Then the bias or intercept has been decided(b). The equation of calculate the input of h1 is $h1 = i1*w1+i2*w2+b1$ carrying the same process for h2. Then the procedure continues to $out = h1*w5+h2*w6 + b3$. Calculating the steps at the end we will have an output value, the next and final step of forward pass is to calculate the difference between our prediction and the actual output. The equation is $Error=1/2(prediction-actual)^2$, Error function is always positive and the $\frac{1}{2}$ is added to ease the derivative. Now the main goal is to minimize the error between prediction and actual output.

Furthermore, our main goal is to change the prediction value as the actual value is constant and we can not change it. The solution to the problem is changing or updating the weights values so the error is reduced. One way of doing it is by using backpropagation. The goal of backpropagation mechanism is to minimize the prediction error by updating the weights using gradient descent*. It calculates the gradient of the error function with respect of the weight of the each neuron.

*[Gradient descent is an algorithm for finding the minimum or the maximum of a function, in backpropagation we are trying to minimize the error function. To find a local minimum with gradient descent taking steps proportional to the negative of the gradient of the function at the current point.]

Backpropagation is the practice that starts from right and moves to the left and fine tuning the weights of a neuron based on the current error subtracted by the partial derivative of error function with respect to neuron weight. The function is: I will use as an example the w5,

$w5(new)=w1 -a(dError/dw5)$ where a is the learning rate and we using is to multiply the derivative of the error to make sure the new updated weight is minimizing the error function. The chain rule is been used to evaluate the derivation of the error as delta ($\delta\epsilon\lambda\tau\alpha$ in Greek)

$\delta h1 : dError/dw5 = Error/dprediction*dprediction/dw5$ (in the next question I will simplify the mathematical terms)

Similarly, all the weights are being updated, so we are moving back to the hidden layer weights and adjusting them using again the chain rule to each weight. The model generalization is been increased by using proper weight tuning and ensures lower error. Now using the new weights the whole procedure is been repeated starting with the forward phase. We now have a new prediction which is closer to the actual output. This procedure can be repeated as many times as an epoch number has been reached otherwise the error is close to zero or its been minimized to a threshold.

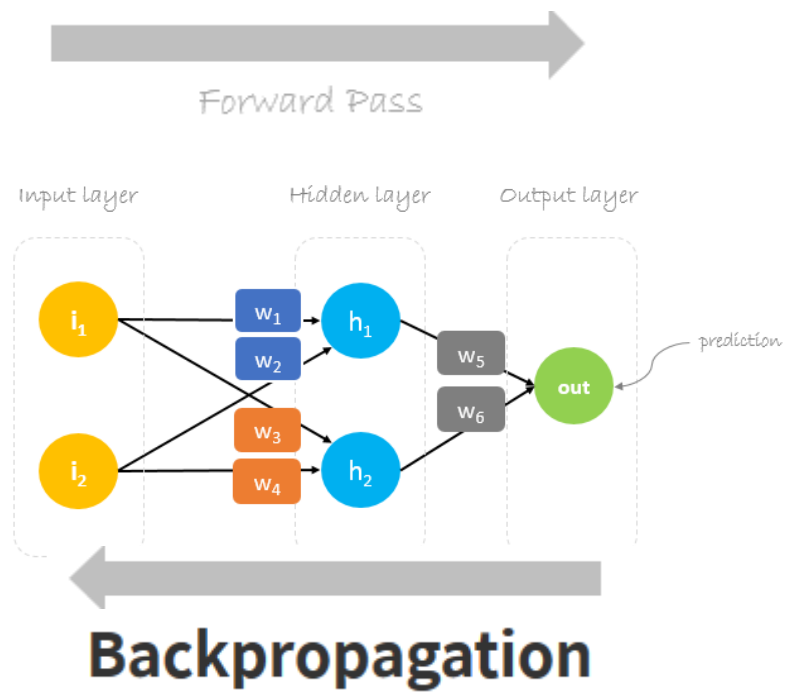


Figure 1: Visualization of our network

BONUS QUESTION:

- Describe the process of backpropagation in mathematical terms. Here, explain (in English, in about 500 words) what each equation you give does, and relate this to the answers given in Question 5. You are welcome to express equations in your own R or python code rather than using equation layout, but you need to make clear you understand what each line is doing. (Question 6, 5 points).

Here i am only write and explain the mathematical equations :

(1) I am starting with Forward Pass: My Neural Network will be like this

The goal is to optimize the weights so the NN can learn how to better predict the result.

(2) I will calculate the total net input of h_1 :

$$net_{h_1} = w_{11} * i_1 + w_{12} * i_2 + b_1 * 1 = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

The equation here shows how to calculate the input value of the neuron in hidden layer.

The output is been calculated by the logistic function:

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}} = 0.5932 \text{ equally } out_{h_2} = 0.5968$$

After that we go to the next layer the output layer.

$$net_{o_1} = w_{31} * out_{h_1} + w_{32} * out_{h_2} + b_3 * 1 = 1.105$$

using again the logistic function $out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} = 0.75$

and $out_{o_2} = 0.772$

We finish the forward Pass by calculating the Total Error

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

The \sum means the sum of the 2 errors from each neuron.

$$E_{total} = \frac{1}{2} (0.01 - 0.75)^2 + \frac{1}{2} (0.99 - 0.772)^2 = 0.99$$

the number 0.01 and 0.99 are the target we have from the beginning.

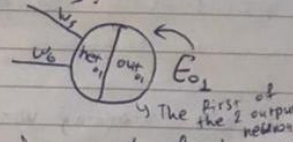
Now the Backward Pass

Let's take w_5 we want to know how much a change to w_5 affects the total error $\frac{dE_{total}}{dw_5}$ is the partial derivative of E_{total} with respect to w_5 .

We have to apply the chain rule

$$\frac{dE_{total}}{dw_5} = \frac{dE_{total}}{dout_{o_1}} \times \frac{dout_{o_1}}{dnet_{o_1}} \times \frac{dnet_{o_1}}{dw_5}$$

(2) (3) (4)



And we need to use the partial derivatives and find the solution

$$(2) \frac{dE_{total}}{dout_{o_1}} = \left(\frac{1}{2} (target_{o_1} - out_{o_1})^2 \right)' + 0$$

is zero because we don't have out_{o_1} so the partial derivative is 0

$$\Rightarrow \frac{dE_{total}}{dout_{o_1}} = 2 \cdot \frac{1}{2} (target_{o_1} - out_{o_1})^{2-1} \times - \left(\frac{out_{o_1}}{out_{o_1}} \right) = - (target_{o_1} - out_{o_1}) =$$

$$= 0.741$$

$$(3) \Rightarrow \frac{dout_{o_1}}{dnet_{o_1}} = out_{o_1} (1 - out_{o_1}) = 0.1868$$

That's why the

partial derivative of $out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}}$ $\Rightarrow \frac{dout_{o_1}}{dnet_{o_1}} = \frac{d(1 + e^{-net_{o_1}})^{-1}}{dnet_{o_1}}$

$$= \frac{-(-e^{-net_{o_1}})}{(1 + e^{-net_{o_1}})^2} = \frac{+e^{-net_{o_1}}}{(1 + e^{-net_{o_1}})^2}$$

is the identity $(a+b)^2 = a^2 + 2ab + b^2 \Rightarrow$

$$= \frac{e^{-net_{o_1}}}{1 + e^{-net_{o_1}}} \text{ And this can be written } out_{o_1} (1 - out_{o_1})$$

Now the final part

$$(4) \frac{dnet_{o_1}}{dw_5} = \frac{dw_5 \times out_{h_1} + w_6 \times out_{h_2} + b_1}{dw_5} = 1 \times out_{h_1} \times w_5^{(1-1)} + 0 + 0 = out_{h_1} = 0.593$$

Now we calculated the $\frac{dE_{total}}{dw_5} = 0.741 \times 0.1868 \times 0.593 = 0.08$

~~The Derivative~~ To decrease the error we subtract the value with the current weight

$$w_5^{(new)} = w_5 - \alpha \times \frac{dE_{total}}{dw_5} = 0.358 \rightarrow \text{The new weight}$$

α : learning rate

I thought it was easier for me (as a mathematician) to write my solution on the paper.