ECE 457B Design Project Report

Emotion Classification of Headshots using Neural Networks

Group Number - ?

Aaron Morais
20413440
aemorais@uwaterloo.ca

Adrian Cheung
20421743
a32cheun@uwaterloo.ca

Darren Mo
20436931
darren.mo@uwaterloo.ca

Rahim Mitha
20427143
rkmitha@uwaterloo.ca

Abstract

The problem being solved is the classification of faces into emotions. Specifically, we will classify faces into 5 basic emotions (Sad, Angry, Happy, Fear, Disgust). The ability to classify emotions would be useful for human-computer interactions, being able to adapt the system based on the user's emotions.

Since this is a classification problem, the proposed solution is to use a neural network. The traditional approach to image-based classification is to use a convolutional neural network (CNN) based on LeCunn and Bengio's proposal. A CNN is a technique for processing images into a neural network by creating a sliding window across the image and processing and evaluating the image sections at a time. CNNs have been used to classify facial expressions before. We will apply a CNN (implemented using TensorFlow) to still images of headshots that show distinct facial expressions.

The training dataset will come from the Japanese Female Facial Expression (JAFFE) database. The dataset consists of 10 people, 5 emotions each, and approximately 3 variations per emotion, resulting in 153 images to train and test on. We may use more images if necessary. We will test our implementation using 4-fold cross-validation, aiming for at least 70% accuracy.

1. Background

Emotions are a mental state or process in response to an external eliciting condition (Oatley, 1992) [1]. Emotions are typically accompanied by a conscious preoccupation, bodily disturbance and expressions. A conscious preoccupation means that emotions draw a lot of attention of the receiver. A bodily disturbance is referring to autonomic nervous system and other physiological processes. Finally expressions, the most important to our project, are recognizable facial gestures, bodily postures and tones of voice that are not entirely voluntary.

The problem of classifying the emotion that a particular person is experiencing at a particular time is challenging. Expressions can be a reliable way of classifying a basic emotion with some certainty, since they are not entirely voluntary. Although expressions can be faked, there are particular patterns of muscle movements that are involuntary when experiencing emotions. For example, happiness typically involves smiling with particular patterns of muscle movements around the eyes, and a lightness and spontaneity of speaking (Oatley, 1992). The reliability of expressions is the reason why we use the Japanese Female Facial Expression (JAFFE) database in our system.

2. Motivation

Classifying emotions has applications in many fields from security and human-computer interaction to public speaking and cognitive science. Building a system to automate classifying emotions opens up possibilities as well as takes advantage of the benefits of automation. The benefits of automation include reducing human error, increased consistency and predictability of results.

Eliciting conditions of emotions (for example a loud sharp sound that causes fear), are experienced differently in every individual. The stimulus is appraised according to the internal goals and motivations of the receiver. Thus classifying emotions can give us insights into the internal goals and motivations of an individual.

The core of an emotional experience is a concept called *Action Readiness*, which is a change in the readiness to perform an action that is typically specified to a range of options (Frijda, 1986) [2]. For example, when frightened we become ready to freeze, fight or flee. This leads to an important motivation for classifying emotion, which is providing the ability to gain insight on the future actions of an entity that is experiencing an emotion.

3. Applications

A system to classify emotion could potentially have numerous applications. Classifying the emotion of individuals is difficult even when conducted manually. A reliable system would be applicable for use in many real-world scenarios. The following are some potential applications of an emotion classifier that has facial expressions as input data.

3.1 Security

A real-time emotion classifier can use data from facial expressions to detect when someone is experiencing an out of place emotion. For example, if a camera on a subway platform detects that a large percentage of bystanders are experiencing fear, when normally they would be neutral, then it can raise a warning for that subway platform to be investigated by a human.

An emotion classifier can also be used in interrogations. Even though interrogations are typically performed by humans who can classify emotions on their own, there are often micro-expressions that are too fast for the human eye to perceive. An automated emotion classifier would be able to catch these micro-expressions and use data from the micro-expressions to classify an emotion that may be missed by the human interrogator.

3.2 Human Computer Interaction

With robotics becoming more and more advanced, and interactions becoming both complex and natural (human-like), emotion classification can play an important role. Computer interfaces that interact with humans can use emotional data to dynamically react to situations. This can potentially improve the user experience such that tasks are completed more efficiently and with a higher degree of satisfaction.

One specific use case for augmenting Human Interaction is recognizing user frustration. When a user is classified as angry or frustrated in some sense, the interface can adapt to provide more guidance to help the user accomplish their desired task. Getting user feedback about features will also be greatly simplified. Without requiring a survey, user sentiment can be recorded. By using emotional data, our models and motivations for the structure of our interactive systems can change in a positive manner.

3.3 AI and Computation

Emotions can be used as a heuristic for narrowing down alternatives, effectively reducing the total amount of computations needed to determine an action. A primitive example is a robot that detects that a person is angry. It would use this emotion classification as a heuristic to reduce its options for action to the set of *calming actions*. A robot perceptive of emotions could provide simple mental-health analysis, allowing users to more easily self regulate their health.

While there are many additional advancements necessary for true artificial intelligence to exist, demonstrating emotion classification is an important step forward. In order for an AI to communicate directly with humans, it must fully understand the context from which it is communicating. This includes understanding the emotional state of the humans that it is communicating with.

4. Problem

Given an image of a human face, we would like to classify the emotion expressed by the individuals face. We would like to select a particular emotion from a predetermined set of emotions to represent the dominant emotion expressed by the individual in the image. The scope of this problem is limited to just five emotions. The emotions of “Happy”, “Sad”, “Angry”, “Disgust” and “Fear” have been selected for classification. Due to the training set we have selected, the scope is also limited to just black and white images of female Japanese faces. The training set used is provided by the Japanese Female Facial Expression Database. The database provides images with dataset provides images prelabeled with emotions allowing for training off of accurate classifications.

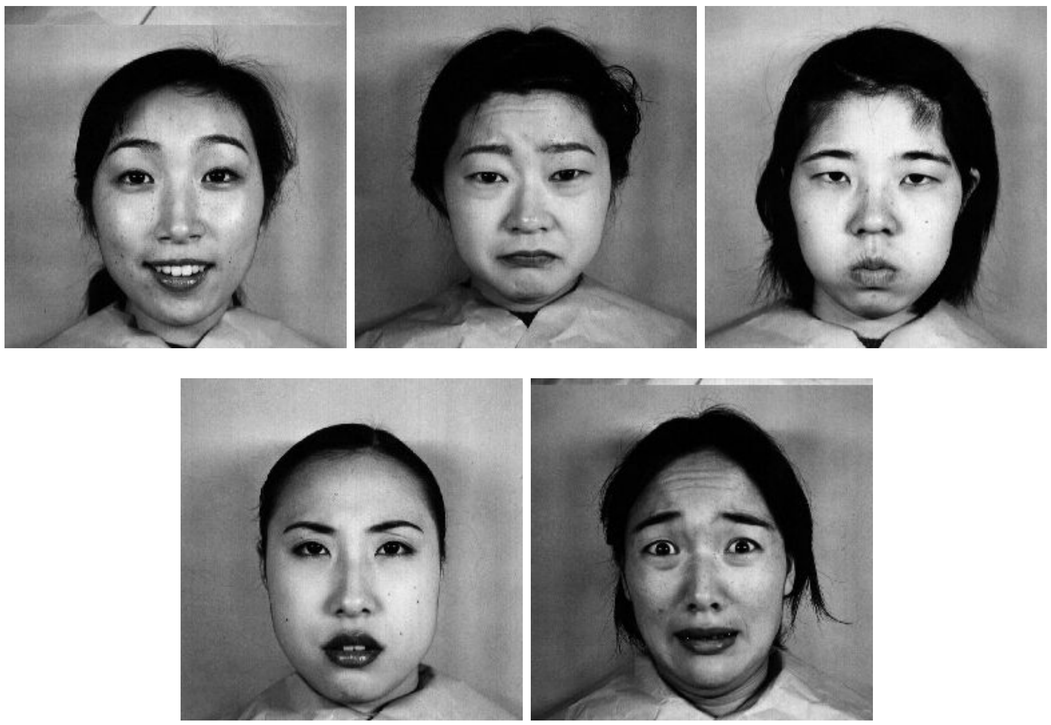


Figure 4.1 Sample Input Images. Clockwise from top left: Happy, Sad, Angry, Disgust, Fear

Classification of emotions must be conducted in a fast and efficient manner. The hardware available to us to perform classification is one GPU optimized machine. We would like any training to be conducted within a reasonable amount of time. After training has been conducted, we would like classification of a single new image to produce a result in a matter of seconds. Our solution should not rely on additional image processing of features. Image processing and facial feature detection is conducted at classification time. We would like a solution that does not have to rely on this. We ideally would like to leverage the behaviour of something like a neural network to learn about features about the human face without explicit definition of said features.

5. Proposed Solution

To solve the problem of emotional classification of images, we propose a convolutional neural network solution. Convolutional neural networks have already been successfully applied to similar problems in this space. Reliable image recognition, video analysis, and natural language processing systems exist today that are backed by convolutional neural network data. We can look at past research, such as LeCunn and Bengio's proposal [4], to find proven techniques for classifying facial expressions. We can apply these techniques to develop our own convolutional neural network solution to classify emotions in our images. We choose a convolutional neural network over a pure fully connected layer because our problem space deals with 2-D images. The use of convolutional layers is expected to be more successful than the fully connected layers because it is expected to be better at finding features and patterns of regional areas.

Our proposed solution involves the use of TensorFlow, an open source software library for machine learning. There are many libraries that enable the implementation of our solution. However, TensorFlow provides some of the best performance over its competitors. In addition, the library allows developers to write in Python, a programming language that our team has a lot of past experience working with. For these reasons, TensorFlow allows us to implement our solution as quickly as possible.

Using TensorFlow, we develop our Python application to train on the Japanese Female Facial Expression database. The result is then tested using 4-fold cross validation. The structure of our network is a mix between ImageNet [4] and TensorFlow's tutorial for classification with MNIST [5]. ImageNet starts with 2 convolutional pooling layer pairs with 5x5 sliding windows, which we have adapted into our network. It then uses 3 more convolutional layers but without pooling layers, and finally ending with 2 fully connected layers of 2048 nodes each. Since we have significantly less labels to predict than ImageNet (1000), our network ends with two fully connected layers of 1024 nodes instead. TensorFlow has each convolutional layer extract twice as many features as the last, starting with 32 features in the first layer [6]. We chose the number of features to extract based on this doubling principle.

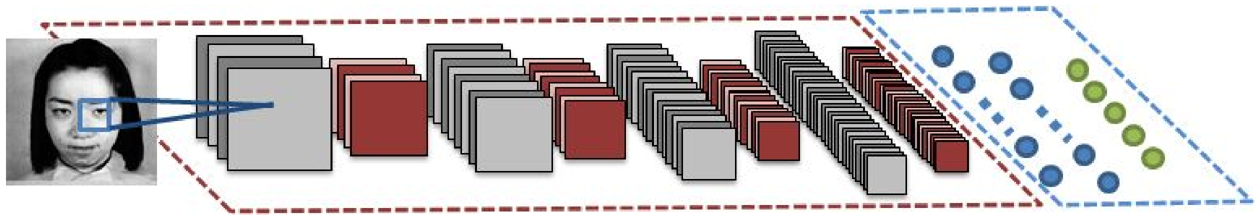


Figure 5.1 Layout of Neural Network Solution. The red box is for feature extraction and the blue box is classification

Our final convolutional neural network design, shown in Figure 5.1, has four layers of convolutional and pooling pairs of windows size 5x5 and 2x2 respectively. The features extracted by these layers are 32, 64, 128, and 256. Each convolutional layer uses a rectified linear unit (ReLU) which changes all negative values to 0 and keeps the positive values as is. The last pooling layer is flattened and connected to a hidden layer of 1024 nodes which in turn is fully connected to another hidden layer of 1024 nodes. Both hidden layers use a sigmoid activation function. The last hidden layer is connected to a dropout layer. The dropout rate is 50%. This final layer before the output layer is to provide randomness into the net. By adding a dropout layer there is a chance to minimize overfitting in the training [7]. The final output layer consists of 5 output nodes, corresponding to each possible label. The output layer applies a softmax over the 5 nodes such that the value of each node is between 0 and 1 and the sum of all output nodes is 1. The output of each node is the confidence the network has in its prediction of each label. The node with the highest probability is selected as the predicted label.

In order to train this network a loss function needs to be defined. In our case we use the cross entropy function defined in Figure 5.2. We define $\epsilon = 10 * -9$ for instances where yi equals

1 or 0. In this case, t is the target label and y is the label produced by the neural network. The cross entropy function is large when the labels are far from being correct. The goal of the training is to minimize the output of this function.

$$-\sum_{i=1}^{5} t_i \log(y_i + \epsilon) + (1 - t_i) \log((1 - y_i) + \epsilon)$$

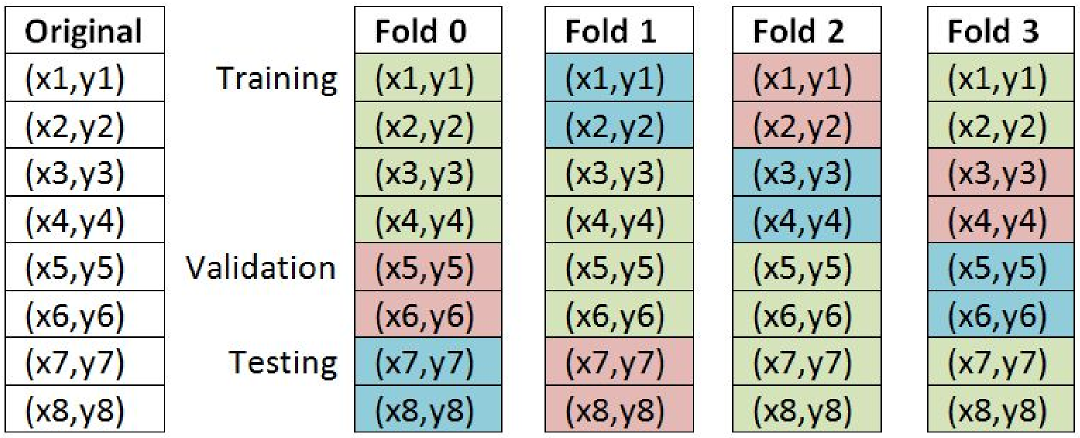
Figure 5.2 Cross entropy function

The learning algorithm used is the Adam Optimizer [8]. This is a variation on Stochastic Gradient Descent (SGD), which is a very common optimization algorithm. This method finds individual learning rates for each parameter, unlike traditional SGD which has a fixed size learning rate. Adam is an extension to the Adaptive gradient (AdaGrad) method, another variation on SGD. Both techniques have the advantage over SGD in that they calculate the moment of the gradient. However, Adam requires less memory to perform than AdaGrad. It is also a provided optimization method within TensorFlow. Adam requires an initial stepsize hyperparameter which we have selected as 10^-5. This value was selected because stepsizes of smaller or larger magnitude either took longer to converge, diverged, or got caught in local minima.

6. Training and Evaluation

6.1 Data partitioning

A common bottleneck to the neural network approach is having too few data points to work with. Even if a network can successfully train on the limited amount of data and achieve a strong training and test accuracy, due to the low sample size it does not provided high confidence. The solution to this problem is to use a 4-fold Cross-Validation method. In 4-fold Cross Validation, the neural network is trained four separate times with randomly selected training, validation, and testing sets. Where each validation and testing set is mutually exclusive to each respective set in the other training iterations. The advantage of the 4-fold Cross Validation method is that each training and testing set pair is independent and random, lowering the possibility for biased distribution of features found in either the testing or training set. For example, in Figure 6.1, a 4-Fold Cross Validation partitioning is applied to 8 data points. If only Fold 0 is used and (x7, y7) and (x8, y8) happen to be outliers then the neural network will unfairly be considered bad. Conversely, if (x7, y7) and (x8, y8) happen to be prototypical examples then the neural network will unfairly be considered to do very well.



Original		Fold 0	Fold 1	Fold 2	Fold 3
(x1,y1)	Training	🟩 (x1,y1)	🟦 (x1,y1)	🟥 (x1,y1)	🟩 (x1,y1)
(x2,y2)		🟩 (x2,y2)	🟦 (x2,y2)	🟥 (x2,y2)	🟩 (x2,y2)
(x3,y3)		🟩 (x3,y3)	🟩 (x3,y3)	🟦 (x3,y3)	🟥 (x3,y3)
(x4,y4)		🟩 (x4,y4)	🟩 (x4,y4)	🟦 (x4,y4)	🟥 (x4,y4)
(x5,y5)	Validation	🟥 (x5,y5)	🟩 (x5,y5)	🟩 (x5,y5)	🟦 (x5,y5)
(x6,y6)		🟥 (x6,y6)	🟩 (x6,y6)	🟩 (x6,y6)	🟦 (x6,y6)
(x7,y7)	Testing	🟦 (x7,y7)	🟥 (x7,y7)	🟩 (x7,y7)	🟩 (x7,y7)
(x8,y8)		🟦 (x8,y8)	🟥 (x8,y8)	🟩 (x8,y8)	🟩 (x8,y8)

Figure 6.1 An example of 4-Fold Cross Validation on a data set of size 8

The JAFFE dataset provided 153 images for the 5 emotions. The data is shuffled to ensure a random distribution of images to be evaluated. In each iteration of the 4-fold Cross Validation, 113 images are set aside for training. The remaining 40 are split evenly between a validation set and a testing set. In traditional training, the training ends after a given number of training steps have been achieved. In our case we set this value to be 50,000 steps. However, training for this long has two potential problems. The first is that high number of training steps takes a high amount of computing time. The second problem is that training for too long may lead to overfitting. The compromise is to use a validation set. A validation set holds images that are independent of those being trained and of those being tested. At set intervals of 500 training steps, the network can check its performance against the validation set to validate whether or not it is generalizing to unseen data. If the validation set is consistently being evaluated well then the network can decide to end training prematurely instead of having to train for the full 50,000 steps.. For our experimentation we define this consistency as the validation set having greater than 85% accuracy for five intervals in a row.

6.2 Evaluation Metric

The performance of the validation set accuracy is an important output of the neural network. However, these values alone are not entirely sufficient to determine the accuracy of the network. It is not sufficient because the performance of the validation set accuracy is considered during the training of the neural network. Although the validation does not directly influence it, it does have some affect on the parameters learned by the neural network. If the validation set is not considered, then the end result of the training will be be a different network that has a chance of overfitting. As well it is possible that when the validation set reaches a good accuracy, the training has only generalized enough for this particular set and not for all unseen data. The actual performance of the neural network is evaluated based off the average accuracy across each of the four folds on their respective testing sets.

The overall accuracy of the neural network is determined by comparing data from the neural network result to data in the testing set. The labels with the largest predicted value by the network is compared with the actual classification labels of each image in the testing set. This comparison is evaluated as a percentage accuracy. Since the testing set of each of the four folds is the same size, the overall average can be obtained by taking the average accuracy of each fold. The result gives a general impression of how well the network structure and training procedure would generalize if all 153 images were used for training.

7. Results

After implementing our system, we ran our training and validation against our folds in order to use a 4-fold cross validation approach. Each of the four folds displayed different levels of validation confidence and resulted in different levels of test accuracy. Fortunately, as the training steps increased, the validation accuracy followed a similar trend to the training accuracy.

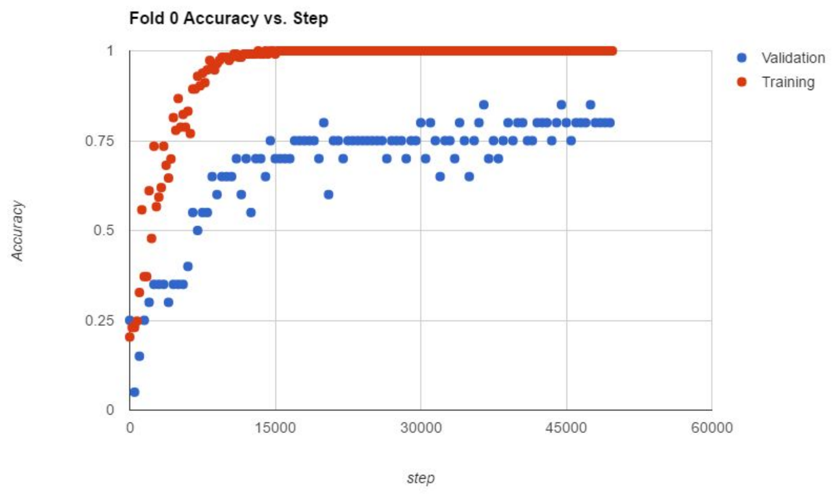


Figure 7.1a Fold 0 Test Accuracy: 85%

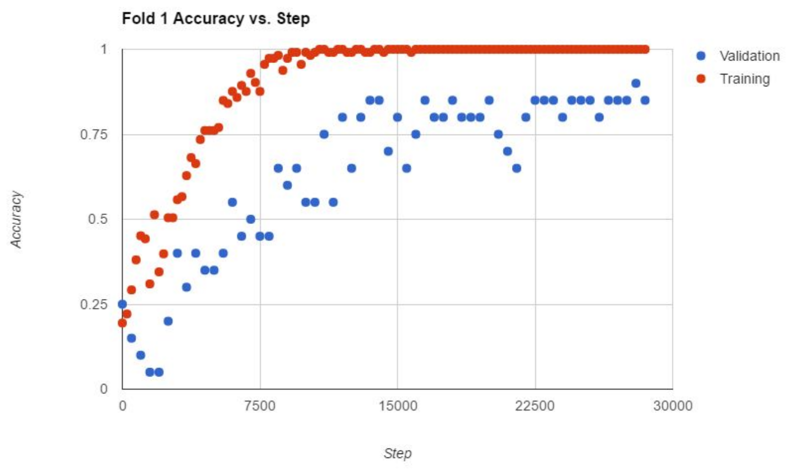


Figure 7.1b Fold 1 Test Accuracy: 90%

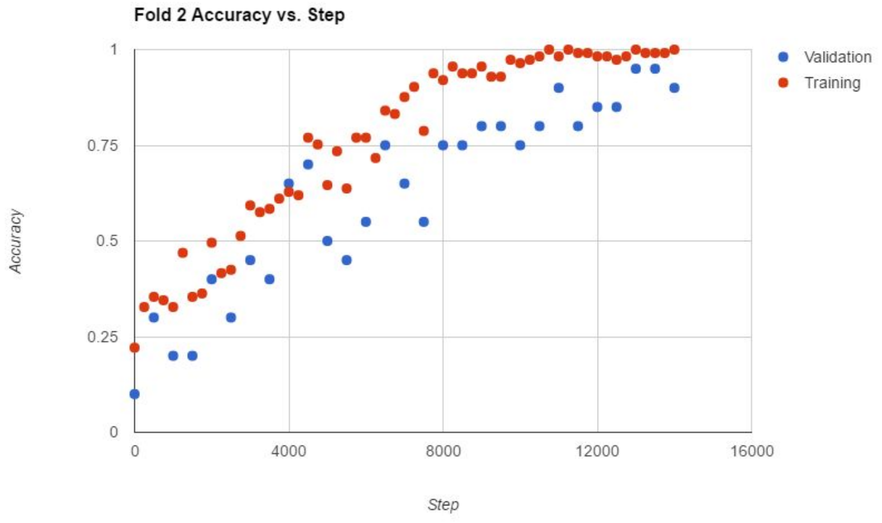


Figure 7.1c Fold 2 Test Accuracy: 80%

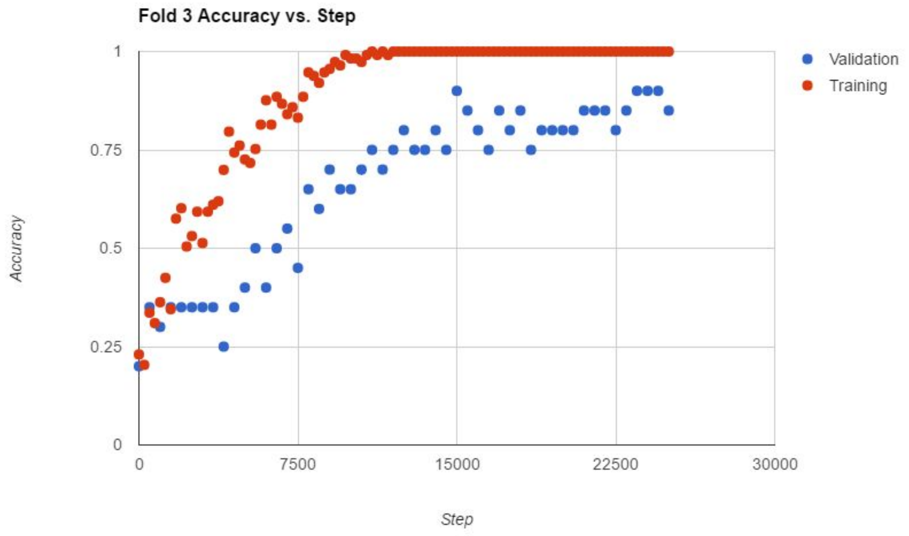


Figure 7.1d Fold 3 Test Accuracy:100%

Of the four folds, three maintained a steady perfect training accuracy and were able to take advantage of steady validation in order to end training early. The fourth fold, shown in Figure 7.1c, which did not reach this steady state continued to train until all 50,000 steps were finished. This fold resulted in the lowest test accuracy. Which is to be expected from the imperfect training. The average accuracy across all folds is 88.75%.

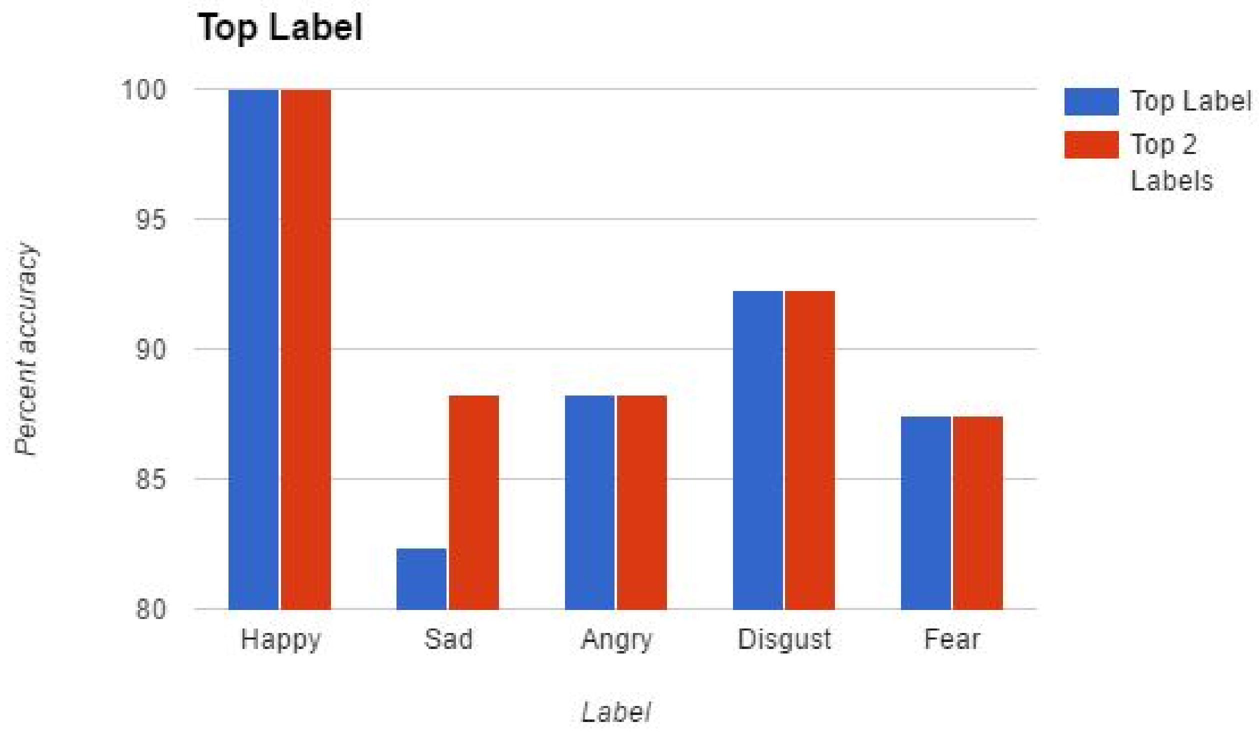


Figure 7.2 Label accuracy for Top prediction and Top-2 prediction

Each emotion label achieved different levels of accuracy as shown in Figure 7.2. The ‘Happy’ label performed the best and was correctly predicted in 100% of cases. Followed by “Disgust” which achieved 92% accuracy. The “Sad” images were the least effective. However, when we consider the second highest scoring prediction in addition to just the top scoring prediction, the “Sad” image accuracy increases to the same level as “Angry”. Although “Sad” increased, the Top-2 prediction analysis provided no increase in accuracy for the other labels. This means that the features learned by the neural network were not applicable to the misclassified instances and the training had missed on some important features. It is likely that the reason “Happy” has a much higher accuracy range is because the mouth shapes, namely a smile, in a “Happy” expression are clearly different from the mouth shapes of the other emotions. This allows for less ambiguity. A frown can be found in both “Sad” and “Angry” expressions. The remaining labels however often depends on more facial features to be certain; mainly eyes and mouth together.

Table 7.1 The labels on the left are the correct labels. Each column represents the predicted label counts for the respective labels

	Happy	Sad	Angry	Disgust	Fear
Happy	17	0	0	0	0
Sad	2	14	0	1	0
Angry	0	0	15	2	0
Disgust	0	0	1	12	0
Fear	0	1	0	1	14

From Table 7.1 we can see that the most common mistakes come from incorrectly labelling the image as “Disgust”. This happens for each image type except for “Happy” which is correctly labelled each time. It is possible that the range of features to display disgust are so great that other emotions are often confused for it. This also shows that the high accuracy for labeling “Disgust” may not be due to understanding what features best correlate with that emotion but because the neural network is trained to more commonly predict “Disgust” during uncertain cases. Although happy and sad are considered to be opposites and clearly distinguishable, the classifier misclassified a sad image as “Happy” twice.

For each prediction, each label is given a confidence between 0 and 1, totaling 1 for the sum of all confidences. The largest confidence is selected to be the label. In Figure 7.3 we compare the average confidence for labels that were correctly predicted and labels that were incorrectly predicted. When the label is correctly predicted there is a high confidence. Incorrect labels have low confidence. This is important because if the incorrect labels had high confidence, then that implies there are certain features that are incorrectly influencing the neural network.

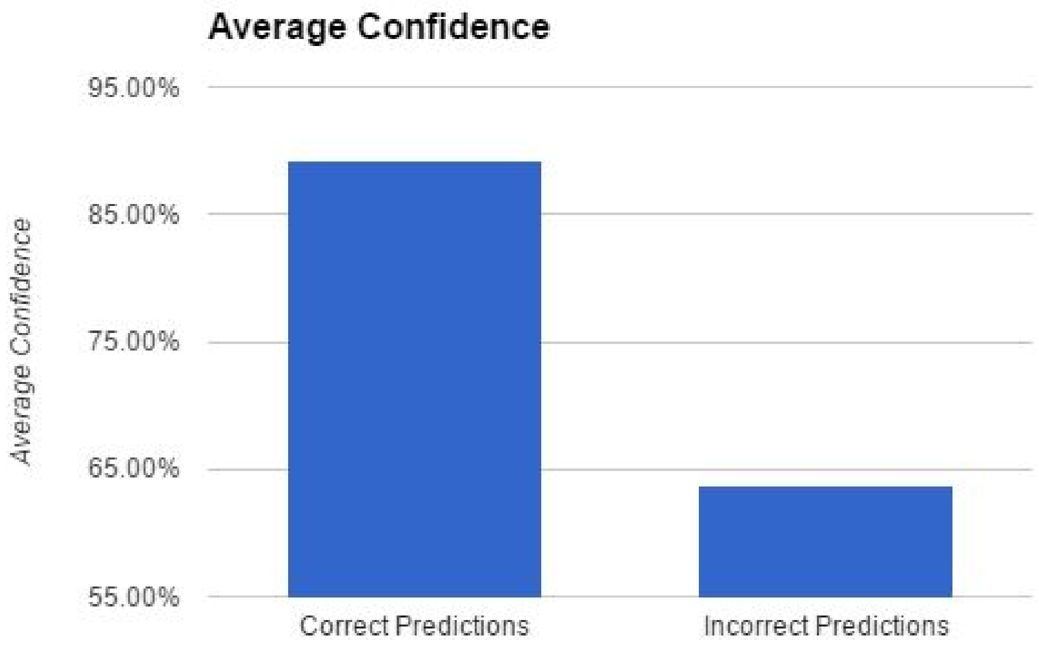


Figure 7.3 Confidence in predictions

We compared the performance of the neural network results to that of humans performing the same classification. This allowed us to get a better sense for how well a machine should be expected to perform these classifications. To collect this human data, human participants were asked to classify the emotion depicted by a face for an image without a labelled emotion. The images used were from the same database and the sample size of the images was similar to that of the tests performed on our neural network. The resulting human performance was very similar to that of our neural network. Just like the neural network, images that had faces expression the emotion of “Happy” were correctly labeled 100% of the time. For the other emotion labels, the human performance was poorer than that of the network performance at correctly predicting the emotion. While the performances were similar, this highlights the future potential of the neural network system. If further improvements are made, the potential applications for this software can be realized as the machine will be able to classify emotions more reliably than humans.

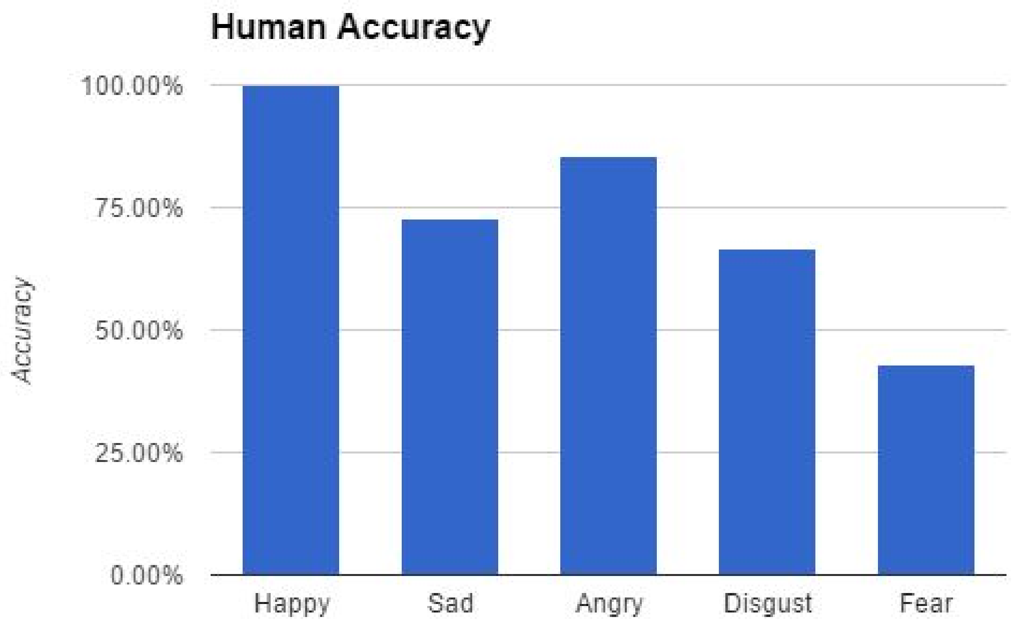


Figure 7.4 Human rating accuracy

The testing accuracy obtained by each fold fluctuated a large amount. This shows that the different partitioning has a large influence on the training behaviour. The partitioning method chosen is to distribute the data between training, validation, and testing is a pure random shuffle. This leads to the possibility of a bad distribution of the data. Bad distributions can include too many or too few of a particular label or too many of the same face used. The JAFFE dataset has each women displaying each label multiple times. If the distribution has a disproportionate amount of one woman for a particular label then the features learnt might be for the woman herself and not of the expression she is making. Better results might be obtained if each fold had an even distribution of labels and ensured that for each label there was a distribution of the women used for each emotion. This gives the neural network a better chance at finding the relevant features.

8. Conclusion

Using our Convolutional Neural Network, we were able to successfully distinguish faces based on their facial expressions with great accuracy. Which far exceeded the goal of 75% accuracy. While there are improvements that can be made, emotions were predicted by the network with a greater level of accuracy and confidence that our human raters are capable of.

There are many aspects of the network that can be improved upon in the future. Although our network employed the use of a dropout layer, it did not explore different levels of dropout. The use of dropout was fixed at 50%. Future work can experiment with smaller or larger dropout rates. In addition, it is probable that modifying the number of convolutional layers or modifying the number of nodes for the fully connected layers would allow the network to produce better results.

In order to increase the scope of our solution for real-world scenarios, the data used for the training set needs to be improved. The current limitations of the training set are multi-faceted. The images are black and white and of low resolution. Increasing the level of detail in the images will provide more data points from which the neural network can perform analysis, potentially improving classification performance. The training set also has a limited demographic of humans represented. All of the subjects photographed are female and Japanese. Having more equal racial and gender representation will allow the neural network to perform well on more diverse types of inputs. By improving the training set, the neural network will perform well in more situations, allowing it to be applied in many practical scenarios.

Works Cited

1. Oatley, K., “The Structure of Emotions”. Best Laid Schemes, pp. 14-32. Cambridge University Press, 1992. Document from paperback (3. Mind Introduction to Cognitive Science by Paul Thagard)
2. Frijda, N.H. “The Emotions”. Cambridge: Cambridge University Press, 1986. Document from paperback (3. Mind Introduction to Cognitive Science by Paul Thagard)
3. Thagard, Paul. *Mind: Introduction to cognitive science*. Vol. 4. Cambridge, MA: MIT press, 1996
4. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012. http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf.
5. LeCun, Yann, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits." (1998).
6. Abadi, Martın, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015." *Software available from tensorflow.org*.
7. Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.
8. Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

Appendix A

Raw Test Results

```
153
Validation Labels
[[ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]]
Testing labels
[[ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  0.]]
Training Size
113
1, 0, training accuracy, 0.153846
1, 0, validation accuracy, 0.25
1, 250, training accuracy, 0.174825
1, 500, training accuracy, 0.230769
1, 500, validation accuracy, 0.15
1, 750, training accuracy, 0.300699
1, 1000, training accuracy, 0.356643
1, 1000, validation accuracy, 0.1
1, 1250, training accuracy, 0.34965
1, 1500, training accuracy, 0.244755
1, 1500, validation accuracy, 0.05
1, 1750, training accuracy, 0.405594
1, 2000, training accuracy, 0.272727
1, 2000, validation accuracy, 0.05
1, 2250, training accuracy, 0.314685
1, 2500, training accuracy, 0.398601
1, 2500, validation accuracy, 0.2
1, 2750, training accuracy, 0.398601
1, 3000, training accuracy, 0.440559
1, 3000, validation accuracy, 0.4
1, 3250, training accuracy, 0.447552
1, 3500, training accuracy, 0.496503
1, 3500, validation accuracy, 0.3
1, 3750, training accuracy, 0.538462
1, 4000, training accuracy, 0.524476
1, 4000, validation accuracy, 0.4
1, 4250, training accuracy, 0.58042
1, 4500, training accuracy, 0.601399
1, 4500, validation accuracy, 0.35
1, 4750, training accuracy, 0.601399
1, 5000, training accuracy, 0.601399
1, 5000, validation accuracy, 0.35
1, 5250, training accuracy, 0.608392
1, 5500, training accuracy, 0.671329
1, 5500, validation accuracy, 0.4
```

```
1, 5750, training accuracy, 0.664336
1, 6000, training accuracy, 0.692308
1, 6000, validation accuracy, 0.55
1, 6250, training accuracy, 0.678322
1, 6500, training accuracy, 0.706294
1, 6500, validation accuracy, 0.45
1, 6750, training accuracy, 0.692308
1, 7000, training accuracy, 0.734266
1, 7000, validation accuracy, 0.5
1, 7250, training accuracy, 0.713287
1, 7500, training accuracy, 0.692308
1, 7500, validation accuracy, 0.45
1, 7750, training accuracy, 0.755245
1, 8000, training accuracy, 0.769231
1, 8000, validation accuracy, 0.45
1, 8250, training accuracy, 0.769231
1, 8500, training accuracy, 0.776224
1, 8500, validation accuracy, 0.65
1, 8750, training accuracy, 0.741259
1, 9000, training accuracy, 0.769231
1, 9000, validation accuracy, 0.6
1, 9250, training accuracy, 0.783217
1, 9500, training accuracy, 0.783217
1, 9500, validation accuracy, 0.65
1, 9750, training accuracy, 0.755245
1, 10000, training accuracy, 0.783217
1, 10000, validation accuracy, 0.55
1, 10250, training accuracy, 0.776224
1, 10500, training accuracy, 0.783217
1, 10500, validation accuracy, 0.55
1, 10750, training accuracy, 0.79021
1, 11000, training accuracy, 0.79021
1, 11000, validation accuracy, 0.75
1, 11250, training accuracy, 0.783217
1, 11500, training accuracy, 0.783217
1, 11500, validation accuracy, 0.55
1, 11750, training accuracy, 0.79021
1, 12000, training accuracy, 0.79021
1, 12000, validation accuracy, 0.8
1, 12250, training accuracy, 0.783217
1, 12500, training accuracy, 0.783217
1, 12500, validation accuracy, 0.65
1, 12750, training accuracy, 0.79021
```

```
1, 13000, training accuracy, 0.79021
1, 13000, validation accuracy, 0.8
1, 13250, training accuracy, 0.783217
1, 13500, training accuracy, 0.783217
1, 13500, validation accuracy, 0.85
1, 13750, training accuracy, 0.79021
1, 14000, training accuracy, 0.79021
1, 14000, validation accuracy, 0.85
1, 14250, training accuracy, 0.783217
1, 14500, training accuracy, 0.79021
1, 14500, validation accuracy, 0.7
1, 14750, training accuracy, 0.79021
1, 15000, training accuracy, 0.79021
1, 15000, validation accuracy, 0.8
1, 15250, training accuracy, 0.79021
1, 15500, training accuracy, 0.79021
1, 15500, validation accuracy, 0.65
1, 15750, training accuracy, 0.783217
1, 16000, training accuracy, 0.79021
1, 16000, validation accuracy, 0.75
1, 16250, training accuracy, 0.79021
1, 16500, training accuracy, 0.79021
1, 16500, validation accuracy, 0.85
1, 16750, training accuracy, 0.79021
1, 17000, training accuracy, 0.79021
1, 17000, validation accuracy, 0.8
1, 17250, training accuracy, 0.79021
1, 17500, training accuracy, 0.79021
1, 17500, validation accuracy, 0.8
1, 17750, training accuracy, 0.79021
1, 18000, training accuracy, 0.79021
1, 18000, validation accuracy, 0.85
1, 18250, training accuracy, 0.79021
1, 18500, training accuracy, 0.79021
1, 18500, validation accuracy, 0.8
1, 18750, training accuracy, 0.79021
1, 19000, training accuracy, 0.79021
1, 19000, validation accuracy, 0.8
1, 19250, training accuracy, 0.79021
1, 19500, training accuracy, 0.79021
1, 19500, validation accuracy, 0.8
1, 19750, training accuracy, 0.79021
1, 20000, training accuracy, 0.79021
```

```
1, 20000, validation accuracy, 0.85
1, 20250, training accuracy, 0.79021
1, 20500, training accuracy, 0.79021
1, 20500, validation accuracy, 0.75
1, 20750, training accuracy, 0.79021
1, 21000, training accuracy, 0.79021
1, 21000, validation accuracy, 0.7
1, 21250, training accuracy, 0.79021
1, 21500, training accuracy, 0.79021
1, 21500, validation accuracy, 0.65
1, 21750, training accuracy, 0.79021
1, 22000, training accuracy, 0.79021
1, 22000, validation accuracy, 0.8
1, 22250, training accuracy, 0.79021
1, 22500, training accuracy, 0.79021
1, 22500, validation accuracy, 0.85
1, 22750, training accuracy, 0.79021
1, 23000, training accuracy, 0.79021
1, 23000, validation accuracy, 0.85
1, 23250, training accuracy, 0.79021
1, 23500, training accuracy, 0.79021
1, 23500, validation accuracy, 0.85
1, 23750, training accuracy, 0.79021
1, 24000, training accuracy, 0.79021
1, 24000, validation accuracy, 0.8
1, 24250, training accuracy, 0.79021
1, 24500, training accuracy, 0.79021
1, 24500, validation accuracy, 0.85
1, 24750, training accuracy, 0.79021
1, 25000, training accuracy, 0.79021
1, 25000, validation accuracy, 0.85
1, 25250, training accuracy, 0.79021
1, 25500, training accuracy, 0.79021
1, 25500, validation accuracy, 0.85
1, 25750, training accuracy, 0.79021
1, 26000, training accuracy, 0.79021
1, 26000, validation accuracy, 0.8
1, 26250, training accuracy, 0.79021
1, 26500, training accuracy, 0.79021
1, 26500, validation accuracy, 0.85
1, 26750, training accuracy, 0.79021
1, 27000, training accuracy, 0.79021
1, 27000, validation accuracy, 0.85
```

```
1, 27250, training accuracy, 0.79021
1, 27500, training accuracy, 0.79021
1, 27500, validation accuracy, 0.85
1, 27750, training accuracy, 0.79021
1, 28000, training accuracy, 0.79021
1, 28000, validation accuracy, 0.9
1, 28250, training accuracy, 0.79021
1, 28500, training accuracy, 0.79021
1, 28500, validation accuracy, 0.85
[[ 0.0032953   0.59580112  0.02624728  0.00199216  0.37266415]]
[[ 0.  0.  0.  0.  1.]]
[[  9.80947852e-01   1.45935016e-02   3.16959864e-04   2.12240251e-04
    3.92951118e-03]]
[[ 1.  0.  0.  0.  0.]]
[[  9.98135448e-01   1.24364335e-04   3.84689229e-05   1.26978644e-04
    1.57479465e-03]]
[[ 1.  0.  0.  0.  0.]]
[[  1.13477363e-04   8.32725037e-03   1.71770424e-01   8.12261820e-01
    7.52700213e-03]]
[[ 0.  0.  0.  1.  0.]]
[[ 0.01346512  0.00482106  0.02786706  0.30318892  0.65065783]]
[[ 0.  0.  0.  0.  1.]]
[[  1.83550263e-04   9.98913050e-01   6.23585482e-04   2.56606785e-04
    2.32584789e-05]]
[[ 0.  1.  0.  0.  0.]]
[[  9.98179793e-01   1.06608449e-03   1.30061220e-04   1.22108861e-04
    5.02010807e-04]]
[[ 1.  0.  0.  0.  0.]]
[[ 0.00109129  0.00509409  0.04846793  0.66919327  0.27615342]]
[[ 0.  0.  0.  1.  0.]]
[[ 0.0160301   0.0689323   0.00370105  0.73696715  0.17436945]]
[[ 0.  0.  0.  1.  0.]]
[[  6.94075352e-05   3.00642988e-03   9.96650279e-01   2.36131818e-04
    3.77785691e-05]]
[[ 0.  0.  1.  0.  0.]]
[[  4.41666663e-04   1.59004389e-03   3.52612074e-06   9.18744714e-04
    9.97046053e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  7.74431915e-04   9.15515065e-01   8.29509199e-02   6.75834948e-04
    8.38714768e-05]]
[[ 0.  1.  0.  0.  0.]]
[[  2.31089682e-04   9.98658895e-01   8.36185878e-04   7.02304533e-05
    2.03558957e-04]]
```

```
[[ 0.  1.  0.  0.  0.]]
[[  2.08272922e-06   6.78631041e-05   9.99868989e-01   5.66809285e-05
    4.27866189e-06]]
[[ 0.  0.  1.  0.  0.]]
[[  9.96447563e-01   1.44234928e-03   4.64740064e-04   4.78826223e-05
    1.59758748e-03]]
[[ 1.  0.  0.  0.  0.]]
[[  1.77047477e-04   4.11449581e-01   9.60973557e-03   5.77574968e-01
    1.18865934e-03]]
[[ 0.  0.  0.  0.  1.]]
[[ 0.00116832  0.84791738  0.03655368  0.01673202  0.09762855]]
[[ 0.  1.  0.  0.  0.]]
[[  3.75912350e-04   8.32144811e-04   3.16391597e-05   1.97178415e-05
    9.98740613e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  1.50154941e-04   2.59387936e-03   1.43834273e-03   9.92980599e-01
    2.83701718e-03]]
[[ 0.  0.  0.  1.  0.]]
[[  5.44632203e-04   9.97674882e-01   1.16295298e-03   8.75169280e-05
    5.30044839e-04]]
[[ 0.  1.  0.  0.  0.]]
1, test accuracy, 0.9
Validation Labels
[[ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]]
```

```
Testing labels
[[ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]]
Training Size
113
2, 0, training accuracy, 0.174825
2, 0, validation accuracy, 0.1
2, 250, training accuracy, 0.258741
2, 500, training accuracy, 0.27972
2, 500, validation accuracy, 0.3
2, 750, training accuracy, 0.272727
2, 1000, training accuracy, 0.258741
2, 1000, validation accuracy, 0.2
2, 1250, training accuracy, 0.370629
2, 1500, training accuracy, 0.27972
2, 1500, validation accuracy, 0.2
2, 1750, training accuracy, 0.286713
2, 2000, training accuracy, 0.391608
2, 2000, validation accuracy, 0.4
2, 2250, training accuracy, 0.328671
2, 2500, training accuracy, 0.335664
2, 2500, validation accuracy, 0.3
2, 2750, training accuracy, 0.405594
2, 3000, training accuracy, 0.468531
2, 3000, validation accuracy, 0.45
```

```
2, 3250, training accuracy, 0.454545
2, 3500, training accuracy, 0.461538
2, 3500, validation accuracy, 0.4
2, 3750, training accuracy, 0.482517
2, 4000, training accuracy, 0.496503
2, 4000, validation accuracy, 0.65
2, 4250, training accuracy, 0.48951
2, 4500, training accuracy, 0.608392
2, 4500, validation accuracy, 0.7
2, 4750, training accuracy, 0.594406
2, 5000, training accuracy, 0.51049
2, 5000, validation accuracy, 0.5
2, 5250, training accuracy, 0.58042
2, 5500, training accuracy, 0.503497
2, 5500, validation accuracy, 0.45
2, 5750, training accuracy, 0.608392
2, 6000, training accuracy, 0.608392
2, 6000, validation accuracy, 0.55
2, 6250, training accuracy, 0.566434
2, 6500, training accuracy, 0.664336
2, 6500, validation accuracy, 0.75
2, 6750, training accuracy, 0.657343
2, 7000, training accuracy, 0.692308
2, 7000, validation accuracy, 0.65
2, 7250, training accuracy, 0.713287
2, 7500, training accuracy, 0.622378
2, 7500, validation accuracy, 0.55
2, 7750, training accuracy, 0.741259
2, 8000, training accuracy, 0.727273
2, 8000, validation accuracy, 0.75
2, 8250, training accuracy, 0.755245
2, 8500, training accuracy, 0.741259
2, 8500, validation accuracy, 0.75
2, 8750, training accuracy, 0.741259
2, 9000, training accuracy, 0.755245
2, 9000, validation accuracy, 0.8
2, 9250, training accuracy, 0.734266
2, 9500, training accuracy, 0.734266
2, 9500, validation accuracy, 0.8
2, 9750, training accuracy, 0.769231
2, 10000, training accuracy, 0.762238
2, 10000, validation accuracy, 0.75
2, 10250, training accuracy, 0.769231
```

```
2, 10500, training accuracy, 0.776224
2, 10500, validation accuracy, 0.8
2, 10750, training accuracy, 0.79021
2, 11000, training accuracy, 0.776224
2, 11000, validation accuracy, 0.9
2, 11250, training accuracy, 0.79021
2, 11500, training accuracy, 0.783217
2, 11500, validation accuracy, 0.8
2, 11750, training accuracy, 0.783217
2, 12000, training accuracy, 0.776224
2, 12000, validation accuracy, 0.85
2, 12250, training accuracy, 0.776224
2, 12500, training accuracy, 0.769231
2, 12500, validation accuracy, 0.85
2, 12750, training accuracy, 0.776224
2, 13000, training accuracy, 0.79021
2, 13000, validation accuracy, 0.95
2, 13250, training accuracy, 0.783217
2, 13500, training accuracy, 0.783217
2, 13500, validation accuracy, 0.95
2, 13750, training accuracy, 0.783217
2, 14000, training accuracy, 0.79021
2, 14000, validation accuracy, 0.9
[[ 0.02915692  0.00458804  0.02911018  0.38192961  0.55521524]]
[[ 0.  0.  0.  0.  1.]]
[[ 0.49971205  0.42125466  0.00060787  0.02089532  0.05753015]]
[[ 1.  0.  0.  0.  0.]]
[[ 0.04581391  0.28701091  0.03002202  0.58937049  0.0477826 ]]
[[ 0.  0.  1.  0.  0.]]
[[ 0.01591962  0.01648701  0.87737834  0.08056352  0.00965152]]
[[ 0.  0.  1.  0.  0.]]
[[ 0.05947873  0.35239774  0.008023    0.00705114  0.57304943]]
[[ 0.  0.  0.  0.  1.]]
[[ 0.41984269  0.0127723   0.00676084  0.53731847  0.02330573]]
[[ 1.  0.  0.  0.  0.]]
[[ 0.02856445  0.29978186  0.01025507  0.07892824  0.58247036]]
[[ 0.  0.  0.  0.  1.]]
[[  3.29660601e-04   2.49852473e-03   9.89391327e-01   4.94593522e-03
    2.83453031e-03]]
[[ 0.  0.  1.  0.  0.]]
[[ 0.80432385  0.1845447   0.00103633  0.00428464  0.00581047]]
[[ 1.  0.  0.  0.  0.]]
[[  5.35566476e-04   4.47991071e-04   9.26939309e-01   7.17335045e-02
```

```
   3.43638210e-04]]
[[ 0.  0.  1.  0.  0.]]
[[ 0.01417443  0.0468694   0.02275302  0.87241209  0.04379101]]
[[ 0.  0.  0.  1.  0.]]
[[  1.16229691e-02   6.84499694e-03   9.45971173e-04   2.71137757e-03
    9.77874696e-01]]
[[ 0.  0.  0.  0.  1.]]
[[ 0.0177418   0.9344787   0.03536082  0.01058269  0.00183609]]
[[ 0.  1.  0.  0.  0.]]
[[ 0.04588518  0.17484781  0.06438573  0.65709883  0.05778244]]
[[ 0.  0.  1.  0.  0.]]
[[ 0.0207527   0.11490136  0.85277379  0.00941609  0.00215593]]
[[ 0.  0.  1.  0.  0.]]
[[ 0.03844126  0.00561398  0.01316184  0.00963981  0.93314314]]
[[ 0.  0.  0.  0.  1.]]
[[ 0.15108639  0.60666341  0.00934512  0.12334833  0.10955675]]
[[ 0.  1.  0.  0.  0.]]
[[ 0.01506826  0.06210735  0.02283692  0.89739078  0.00259666]]
[[ 0.  0.  0.  1.  0.]]
[[ 0.57524592  0.3659476   0.00903549  0.01546843  0.03430254]]
[[ 1.  0.  0.  0.  0.]]
[[ 0.12097215  0.37293005  0.02259682  0.40233132  0.08116973]]
[[ 0.  1.  0.  0.  0.]]
2, test accuracy, 0.8
Validation Labels
[[ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0.]]
Testing labels
[[ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]]
Training Size
113
3, 0, training accuracy, 0.181818
3, 0, validation accuracy, 0.2
3, 250, training accuracy, 0.160839
3, 500, training accuracy, 0.265734
3, 500, validation accuracy, 0.35
3, 750, training accuracy, 0.244755
3, 1000, training accuracy, 0.286713
3, 1000, validation accuracy, 0.3
3, 1250, training accuracy, 0.335664
3, 1500, training accuracy, 0.272727
3, 1500, validation accuracy, 0.35
3, 1750, training accuracy, 0.454545
3, 2000, training accuracy, 0.475524
3, 2000, validation accuracy, 0.35
3, 2250, training accuracy, 0.398601
3, 2500, training accuracy, 0.41958
3, 2500, validation accuracy, 0.35
3, 2750, training accuracy, 0.468531
```

```
3, 3000, training accuracy, 0.405594
3, 3000, validation accuracy, 0.35
3, 3250, training accuracy, 0.468531
3, 3500, training accuracy, 0.482517
3, 3500, validation accuracy, 0.35
3, 3750, training accuracy, 0.48951
3, 4000, training accuracy, 0.552448
3, 4000, validation accuracy, 0.25
3, 4250, training accuracy, 0.629371
3, 4500, training accuracy, 0.587413
3, 4500, validation accuracy, 0.35
3, 4750, training accuracy, 0.601399
3, 5000, training accuracy, 0.573427
3, 5000, validation accuracy, 0.4
3, 5250, training accuracy, 0.566434
3, 5500, training accuracy, 0.594406
3, 5500, validation accuracy, 0.5
3, 5750, training accuracy, 0.643357
3, 6000, training accuracy, 0.692308
3, 6000, validation accuracy, 0.4
3, 6250, training accuracy, 0.643357
3, 6500, training accuracy, 0.699301
3, 6500, validation accuracy, 0.5
3, 6750, training accuracy, 0.685315
3, 7000, training accuracy, 0.664336
3, 7000, validation accuracy, 0.55
3, 7250, training accuracy, 0.678322
3, 7500, training accuracy, 0.657343
3, 7500, validation accuracy, 0.45
3, 7750, training accuracy, 0.699301
3, 8000, training accuracy, 0.748252
3, 8000, validation accuracy, 0.65
3, 8250, training accuracy, 0.741259
3, 8500, training accuracy, 0.727273
3, 8500, validation accuracy, 0.6
3, 8750, training accuracy, 0.748252
3, 9000, training accuracy, 0.755245
3, 9000, validation accuracy, 0.7
3, 9250, training accuracy, 0.769231
3, 9500, training accuracy, 0.762238
3, 9500, validation accuracy, 0.65
3, 9750, training accuracy, 0.783217
3, 10000, training accuracy, 0.776224
```

```
3, 10000, validation accuracy, 0.65
3, 10250, training accuracy, 0.776224
3, 10500, training accuracy, 0.769231
3, 10500, validation accuracy, 0.7
3, 10750, training accuracy, 0.783217
3, 11000, training accuracy, 0.79021
3, 11000, validation accuracy, 0.75
3, 11250, training accuracy, 0.783217
3, 11500, training accuracy, 0.79021
3, 11500, validation accuracy, 0.7
3, 11750, training accuracy, 0.783217
3, 12000, training accuracy, 0.79021
3, 12000, validation accuracy, 0.75
3, 12250, training accuracy, 0.79021
3, 12500, training accuracy, 0.79021
3, 12500, validation accuracy, 0.8
3, 12750, training accuracy, 0.79021
3, 13000, training accuracy, 0.79021
3, 13000, validation accuracy, 0.75
3, 13250, training accuracy, 0.79021
3, 13500, training accuracy, 0.79021
3, 13500, validation accuracy, 0.75
3, 13750, training accuracy, 0.79021
3, 14000, training accuracy, 0.79021
3, 14000, validation accuracy, 0.8
3, 14250, training accuracy, 0.79021
3, 14500, training accuracy, 0.79021
3, 14500, validation accuracy, 0.75
3, 14750, training accuracy, 0.79021
3, 15000, training accuracy, 0.79021
3, 15000, validation accuracy, 0.9
3, 15250, training accuracy, 0.79021
3, 15500, training accuracy, 0.79021
3, 15500, validation accuracy, 0.85
3, 15750, training accuracy, 0.79021
3, 16000, training accuracy, 0.79021
3, 16000, validation accuracy, 0.8
3, 16250, training accuracy, 0.79021
3, 16500, training accuracy, 0.79021
3, 16500, validation accuracy, 0.75
3, 16750, training accuracy, 0.79021
3, 17000, training accuracy, 0.79021
3, 17000, validation accuracy, 0.85
```

```
3, 17250, training accuracy, 0.79021
3, 17500, training accuracy, 0.79021
3, 17500, validation accuracy, 0.8
3, 17750, training accuracy, 0.79021
3, 18000, training accuracy, 0.79021
3, 18000, validation accuracy, 0.85
3, 18250, training accuracy, 0.79021
3, 18500, training accuracy, 0.79021
3, 18500, validation accuracy, 0.75
3, 18750, training accuracy, 0.79021
3, 19000, training accuracy, 0.79021
3, 19000, validation accuracy, 0.8
3, 19250, training accuracy, 0.79021
3, 19500, training accuracy, 0.79021
3, 19500, validation accuracy, 0.8
3, 19750, training accuracy, 0.79021
3, 20000, training accuracy, 0.79021
3, 20000, validation accuracy, 0.8
3, 20250, training accuracy, 0.79021
3, 20500, training accuracy, 0.79021
3, 20500, validation accuracy, 0.8
3, 20750, training accuracy, 0.79021
3, 21000, training accuracy, 0.79021
3, 21000, validation accuracy, 0.85
3, 21250, training accuracy, 0.79021
3, 21500, training accuracy, 0.79021
3, 21500, validation accuracy, 0.85
3, 21750, training accuracy, 0.79021
3, 22000, training accuracy, 0.79021
3, 22000, validation accuracy, 0.85
3, 22250, training accuracy, 0.79021
3, 22500, training accuracy, 0.79021
3, 22500, validation accuracy, 0.8
3, 22750, training accuracy, 0.79021
3, 23000, training accuracy, 0.79021
3, 23000, validation accuracy, 0.85
3, 23250, training accuracy, 0.79021
3, 23500, training accuracy, 0.79021
3, 23500, validation accuracy, 0.9
3, 23750, training accuracy, 0.79021
3, 24000, training accuracy, 0.79021
3, 24000, validation accuracy, 0.9
3, 24250, training accuracy, 0.79021
```

```
3, 24500, training accuracy, 0.79021
3, 24500, validation accuracy, 0.9
3, 24750, training accuracy, 0.79021
3, 25000, training accuracy, 0.79021
3, 25000, validation accuracy, 0.85
[[  4.48406217e-06   1.55068465e-05   9.99876380e-01   9.41011967e-05
    9.56940312e-06]]
[[ 0.  0.  1.  0.  0.]]
[[  8.86301161e-04   2.24373979e-03   8.54235350e-06   5.38957102e-05
    9.96807575e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  9.99767721e-01   1.12667360e-04   1.16796582e-05   6.56077682e-05
    4.24515201e-05]]
[[ 1.  0.  0.  0.  0.]]
[[  9.92530763e-01   4.45693778e-03   3.09547941e-05   1.88353060e-05
    2.96248798e-03]]
[[ 1.  0.  0.  0.  0.]]
[[ 0.03388222  0.05448703  0.84025759  0.00763947  0.06373365]]
[[ 0.  0.  1.  0.  0.]]
[[  9.42064682e-04   6.84219241e-01   3.12927186e-01   5.95648540e-04
    1.31579908e-03]]
[[ 0.  1.  0.  0.  0.]]
[[  2.10207945e-04   1.90416642e-04   9.97235954e-01   2.33044964e-03
    3.30578514e-05]]
[[ 0.  0.  1.  0.  0.]]
[[  2.10475133e-04   1.24107394e-03   2.17753277e-05   1.78487098e-04
    9.98348236e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  8.98995996e-01   9.88128185e-02   4.72588203e-04   4.85119002e-04
    1.23343861e-03]]
[[ 1.  0.  0.  0.  0.]]
[[ 0.01634152  0.47417301  0.01879992  0.04342715  0.44725844]]
[[ 0.  1.  0.  0.  0.]]
[[ 0.00829795  0.92675775  0.05523093  0.0027194   0.00699402]]
[[ 0.  1.  0.  0.  0.]]
[[  3.46843211e-04   1.68388578e-04   3.18866991e-03   9.95456934e-01
    8.39159300e-04]]
[[ 0.  0.  0.  1.  0.]]
[[  1.35731942e-03   1.64329982e-03   1.01830000e-02   9.86200988e-01
    6.15292636e-04]]
[[ 0.  0.  0.  1.  0.]]
[[  4.73667824e-05   4.67624253e-04   1.58544444e-02   9.82449770e-01
    1.18081528e-03]]
```

```
[[ 0.  0.  0.  1.  0.]]
[[  1.63118981e-04   6.46437402e-04   8.21670890e-01   1.45946562e-01
    3.15730050e-02]]
[[ 0.  0.  1.  0.  0.]]
[[  4.42567252e-04   2.10722326e-04   9.96578753e-01   2.73976289e-03
    2.81770754e-05]]
[[ 0.  0.  1.  0.  0.]]
[[  9.98342752e-01   5.12789004e-04   2.08634185e-04   1.45536069e-05
    9.21272789e-04]]
[[ 1.  0.  0.  0.  0.]]
[[  9.99548376e-01   1.24008700e-04   4.82341493e-05   6.82451573e-05
    2.11042105e-04]]
[[ 1.  0.  0.  0.  0.]]
[[  9.99677896e-01   1.06675056e-04   5.34486344e-06   1.77585287e-04
    3.24891917e-05]]
[[ 1.  0.  0.  0.  0.]]
[[  2.16677399e-05   7.21490687e-06   6.11772586e-04   9.99322176e-01
    3.71870228e-05]]
[[ 0.  0.  0.  1.  0.]]
3, test accuracy, 1
Validation Labels
[[ 0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.]]
Testing labels
[[ 0.  0.  0.  0.  1.]
```

```
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0.]]
Training Size
113
0, 0, training accuracy, 0.160839
0, 0, validation accuracy, 0.25
0, 250, training accuracy, 0.181818
0, 500, training accuracy, 0.181818
0, 500, validation accuracy, 0.05
0, 750, training accuracy, 0.195804
0, 1000, training accuracy, 0.258741
0, 1000, validation accuracy, 0.15
0, 1250, training accuracy, 0.440559
0, 1500, training accuracy, 0.293706
0, 1500, validation accuracy, 0.25
0, 1750, training accuracy, 0.293706
0, 2000, training accuracy, 0.482517
0, 2000, validation accuracy, 0.3
0, 2250, training accuracy, 0.377622
0, 2500, training accuracy, 0.58042
0, 2500, validation accuracy, 0.35
0, 2750, training accuracy, 0.447552
0, 3000, training accuracy, 0.468531
0, 3000, validation accuracy, 0.35
0, 3250, training accuracy, 0.48951
0, 3500, training accuracy, 0.58042
```

```
0, 3500, validation accuracy, 0.35
0, 3750, training accuracy, 0.538462
0, 4000, training accuracy, 0.51049
0, 4000, validation accuracy, 0.3
0, 4250, training accuracy, 0.552448
0, 4500, training accuracy, 0.643357
0, 4500, validation accuracy, 0.35
0, 4750, training accuracy, 0.615385
0, 5000, training accuracy, 0.685315
0, 5000, validation accuracy, 0.35
0, 5250, training accuracy, 0.622378
0, 5500, training accuracy, 0.65035
0, 5500, validation accuracy, 0.35
0, 5750, training accuracy, 0.622378
0, 6000, training accuracy, 0.657343
0, 6000, validation accuracy, 0.4
0, 6250, training accuracy, 0.608392
0, 6500, training accuracy, 0.706294
0, 6500, validation accuracy, 0.55
0, 6750, training accuracy, 0.706294
0, 7000, training accuracy, 0.734266
0, 7000, validation accuracy, 0.5
0, 7250, training accuracy, 0.713287
0, 7500, training accuracy, 0.741259
0, 7500, validation accuracy, 0.55
0, 7750, training accuracy, 0.72028
0, 8000, training accuracy, 0.748252
0, 8000, validation accuracy, 0.55
0, 8250, training accuracy, 0.769231
0, 8500, training accuracy, 0.755245
0, 8500, validation accuracy, 0.65
0, 8750, training accuracy, 0.748252
0, 9000, training accuracy, 0.762238
0, 9000, validation accuracy, 0.6
0, 9250, training accuracy, 0.769231
0, 9500, training accuracy, 0.776224
0, 9500, validation accuracy, 0.65
0, 9750, training accuracy, 0.776224
0, 10000, training accuracy, 0.776224
0, 10000, validation accuracy, 0.65
0, 10250, training accuracy, 0.769231
0, 10500, training accuracy, 0.776224
0, 10500, validation accuracy, 0.65
```

```
0, 10750, training accuracy, 0.783217
0, 11000, training accuracy, 0.783217
0, 11000, validation accuracy, 0.7
0, 11250, training accuracy, 0.776224
0, 11500, training accuracy, 0.776224
0, 11500, validation accuracy, 0.6
0, 11750, training accuracy, 0.783217
0, 12000, training accuracy, 0.783217
0, 12000, validation accuracy, 0.7
0, 12250, training accuracy, 0.783217
0, 12500, training accuracy, 0.783217
0, 12500, validation accuracy, 0.55
0, 12750, training accuracy, 0.783217
0, 13000, training accuracy, 0.783217
0, 13000, validation accuracy, 0.7
0, 13250, training accuracy, 0.79021
0, 13500, training accuracy, 0.783217
0, 13500, validation accuracy, 0.7
0, 13750, training accuracy, 0.783217
0, 14000, training accuracy, 0.79021
0, 14000, validation accuracy, 0.65
0, 14250, training accuracy, 0.783217
0, 14500, training accuracy, 0.79021
0, 14500, validation accuracy, 0.75
0, 14750, training accuracy, 0.79021
0, 15000, training accuracy, 0.783217
0, 15000, validation accuracy, 0.7
0, 15250, training accuracy, 0.79021
0, 15500, training accuracy, 0.79021
0, 15500, validation accuracy, 0.7
0, 15750, training accuracy, 0.79021
0, 16000, training accuracy, 0.79021
0, 16000, validation accuracy, 0.7
0, 16250, training accuracy, 0.79021
0, 16500, training accuracy, 0.79021
0, 16500, validation accuracy, 0.7
0, 16750, training accuracy, 0.79021
0, 17000, training accuracy, 0.79021
0, 17000, validation accuracy, 0.75
0, 17250, training accuracy, 0.79021
0, 17500, training accuracy, 0.79021
0, 17500, validation accuracy, 0.75
0, 17750, training accuracy, 0.79021
```

```
0, 18000, training accuracy, 0.79021
0, 18000, validation accuracy, 0.75
0, 18250, training accuracy, 0.79021
0, 18500, training accuracy, 0.79021
0, 18500, validation accuracy, 0.75
0, 18750, training accuracy, 0.79021
0, 19000, training accuracy, 0.79021
0, 19000, validation accuracy, 0.75
0, 19250, training accuracy, 0.79021
0, 19500, training accuracy, 0.79021
0, 19500, validation accuracy, 0.7
0, 19750, training accuracy, 0.79021
0, 20000, training accuracy, 0.79021
0, 20000, validation accuracy, 0.8
0, 20250, training accuracy, 0.79021
0, 20500, training accuracy, 0.79021
0, 20500, validation accuracy, 0.6
0, 20750, training accuracy, 0.79021
0, 21000, training accuracy, 0.79021
0, 21000, validation accuracy, 0.75
0, 21250, training accuracy, 0.79021
0, 21500, training accuracy, 0.79021
0, 21500, validation accuracy, 0.75
0, 21750, training accuracy, 0.79021
0, 22000, training accuracy, 0.79021
0, 22000, validation accuracy, 0.7
0, 22250, training accuracy, 0.79021
0, 22500, training accuracy, 0.79021
0, 22500, validation accuracy, 0.75
0, 22750, training accuracy, 0.79021
0, 23000, training accuracy, 0.79021
0, 23000, validation accuracy, 0.75
0, 23250, training accuracy, 0.79021
0, 23500, training accuracy, 0.79021
0, 23500, validation accuracy, 0.75
0, 23750, training accuracy, 0.79021
0, 24000, training accuracy, 0.79021
0, 24000, validation accuracy, 0.75
0, 24250, training accuracy, 0.79021
0, 24500, training accuracy, 0.79021
0, 24500, validation accuracy, 0.75
0, 24750, training accuracy, 0.79021
0, 25000, training accuracy, 0.79021
```

```
0, 25000, validation accuracy, 0.75
0, 25250, training accuracy, 0.79021
0, 25500, training accuracy, 0.79021
0, 25500, validation accuracy, 0.75
0, 25750, training accuracy, 0.79021
0, 26000, training accuracy, 0.79021
0, 26000, validation accuracy, 0.75
0, 26250, training accuracy, 0.79021
0, 26500, training accuracy, 0.79021
0, 26500, validation accuracy, 0.7
0, 26750, training accuracy, 0.79021
0, 27000, training accuracy, 0.79021
0, 27000, validation accuracy, 0.75
0, 27250, training accuracy, 0.79021
0, 27500, training accuracy, 0.79021
0, 27500, validation accuracy, 0.75
0, 27750, training accuracy, 0.79021
0, 28000, training accuracy, 0.79021
0, 28000, validation accuracy, 0.75
0, 28250, training accuracy, 0.79021
0, 28500, training accuracy, 0.79021
0, 28500, validation accuracy, 0.7
0, 28750, training accuracy, 0.79021
0, 29000, training accuracy, 0.79021
0, 29000, validation accuracy, 0.75
0, 29250, training accuracy, 0.79021
0, 29500, training accuracy, 0.79021
0, 29500, validation accuracy, 0.75
0, 29750, training accuracy, 0.79021
0, 30000, training accuracy, 0.79021
0, 30000, validation accuracy, 0.8
0, 30250, training accuracy, 0.79021
0, 30500, training accuracy, 0.79021
0, 30500, validation accuracy, 0.7
0, 30750, training accuracy, 0.79021
0, 31000, training accuracy, 0.79021
0, 31000, validation accuracy, 0.8
0, 31250, training accuracy, 0.79021
0, 31500, training accuracy, 0.79021
0, 31500, validation accuracy, 0.75
0, 31750, training accuracy, 0.79021
0, 32000, training accuracy, 0.79021
0, 32000, validation accuracy, 0.65
```

```
0, 32250, training accuracy, 0.79021
0, 32500, training accuracy, 0.79021
0, 32500, validation accuracy, 0.75
0, 32750, training accuracy, 0.79021
0, 33000, training accuracy, 0.79021
0, 33000, validation accuracy, 0.75
0, 33250, training accuracy, 0.79021
0, 33500, training accuracy, 0.79021
0, 33500, validation accuracy, 0.7
0, 33750, training accuracy, 0.79021
0, 34000, training accuracy, 0.79021
0, 34000, validation accuracy, 0.8
0, 34250, training accuracy, 0.79021
0, 34500, training accuracy, 0.79021
0, 34500, validation accuracy, 0.75
0, 34750, training accuracy, 0.79021
0, 35000, training accuracy, 0.79021
0, 35000, validation accuracy, 0.65
0, 35250, training accuracy, 0.79021
0, 35500, training accuracy, 0.79021
0, 35500, validation accuracy, 0.75
0, 35750, training accuracy, 0.79021
0, 36000, training accuracy, 0.79021
0, 36000, validation accuracy, 0.8
0, 36250, training accuracy, 0.79021
0, 36500, training accuracy, 0.79021
0, 36500, validation accuracy, 0.85
0, 36750, training accuracy, 0.79021
0, 37000, training accuracy, 0.79021
0, 37000, validation accuracy, 0.7
0, 37250, training accuracy, 0.79021
0, 37500, training accuracy, 0.79021
0, 37500, validation accuracy, 0.75
0, 37750, training accuracy, 0.79021
0, 38000, training accuracy, 0.79021
0, 38000, validation accuracy, 0.7
0, 38250, training accuracy, 0.79021
0, 38500, training accuracy, 0.79021
0, 38500, validation accuracy, 0.75
0, 38750, training accuracy, 0.79021
0, 39000, training accuracy, 0.79021
0, 39000, validation accuracy, 0.8
0, 39250, training accuracy, 0.79021
```

```
0, 39500, training accuracy, 0.79021
0, 39500, validation accuracy, 0.75
0, 39750, training accuracy, 0.79021
0, 40000, training accuracy, 0.79021
0, 40000, validation accuracy, 0.8
0, 40250, training accuracy, 0.79021
0, 40500, training accuracy, 0.79021
0, 40500, validation accuracy, 0.8
0, 40750, training accuracy, 0.79021
0, 41000, training accuracy, 0.79021
0, 41000, validation accuracy, 0.75
0, 41250, training accuracy, 0.79021
0, 41500, training accuracy, 0.79021
0, 41500, validation accuracy, 0.75
0, 41750, training accuracy, 0.79021
0, 42000, training accuracy, 0.79021
0, 42000, validation accuracy, 0.8
0, 42250, training accuracy, 0.79021
0, 42500, training accuracy, 0.79021
0, 42500, validation accuracy, 0.8
0, 42750, training accuracy, 0.79021
0, 43000, training accuracy, 0.79021
0, 43000, validation accuracy, 0.8
0, 43250, training accuracy, 0.79021
0, 43500, training accuracy, 0.79021
0, 43500, validation accuracy, 0.75
0, 43750, training accuracy, 0.79021
0, 44000, training accuracy, 0.79021
0, 44000, validation accuracy, 0.8
0, 44250, training accuracy, 0.79021
0, 44500, training accuracy, 0.79021
0, 44500, validation accuracy, 0.85
0, 44750, training accuracy, 0.79021
0, 45000, training accuracy, 0.79021
0, 45000, validation accuracy, 0.8
0, 45250, training accuracy, 0.79021
0, 45500, training accuracy, 0.79021
0, 45500, validation accuracy, 0.75
0, 45750, training accuracy, 0.79021
0, 46000, training accuracy, 0.79021
0, 46000, validation accuracy, 0.8
0, 46250, training accuracy, 0.79021
0, 46500, training accuracy, 0.79021
```

```
0, 46500, validation accuracy, 0.8
0, 46750, training accuracy, 0.79021
0, 47000, training accuracy, 0.79021
0, 47000, validation accuracy, 0.8
0, 47250, training accuracy, 0.79021
0, 47500, training accuracy, 0.79021
0, 47500, validation accuracy, 0.85
0, 47750, training accuracy, 0.79021
0, 48000, training accuracy, 0.79021
0, 48000, validation accuracy, 0.8
0, 48250, training accuracy, 0.79021
0, 48500, training accuracy, 0.79021
0, 48500, validation accuracy, 0.8
0, 48750, training accuracy, 0.79021
0, 49000, training accuracy, 0.79021
0, 49000, validation accuracy, 0.8
0, 49250, training accuracy, 0.79021
0, 49500, training accuracy, 0.79021
0, 49500, validation accuracy, 0.8
0, 49750, training accuracy, 0.79021
[[  1.52388355e-03   1.23742851e-04   6.41098779e-07   4.11653991e-06
    9.98347640e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  9.08350408e-01   7.89054334e-02   9.11571179e-03   3.49530751e-06
    3.62487533e-03]]
[[ 0.  1.  0.  0.  0.]]
[[  1.92548171e-07   2.44585777e-07   9.99998808e-01   5.17715989e-07
    2.53325084e-07]]
[[ 0.  0.  1.  0.  0.]]
[[  5.67831506e-04   9.99081850e-01   5.05977596e-06   7.56986992e-05
    2.69561482e-04]]
[[ 0.  1.  0.  0.  0.]]
[[ 0.10204661  0.27877843  0.00144751  0.11121619  0.50651121]]
[[ 0.  0.  0.  0.  1.]]
[[  7.49453056e-06   9.97477114e-01   1.96949439e-03   6.49099748e-05
    4.81028575e-04]]
[[ 0.  1.  0.  0.  0.]]
[[  3.39394392e-05   4.96770442e-03   9.94913459e-01   2.21718219e-05
    6.27673726e-05]]
[[ 0.  0.  1.  0.  0.]]
[[  9.99521017e-01   9.04981498e-05   1.87547278e-06   4.58655137e-07
    3.86078900e-04]]
[[ 1.  0.  0.  0.  0.]]
[[  5.96695065e-01   2.44016154e-03   6.90116503e-05   3.07224582e-05
    4.00765061e-01]]
[[ 0.  1.  0.  0.  0.]]
[[  1.29284011e-03   1.14706962e-03   7.65573084e-01   2.31855214e-01
    1.31814726e-04]]
[[ 0.  0.  0.  1.  0.]]
[[  4.31599583e-05   2.55244913e-06   5.60198669e-06   8.85918096e-04
    9.99062836e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  3.47797965e-07   9.10275139e-06   9.99958873e-01   2.85470778e-05
    3.16857472e-06]]
[[ 0.  0.  1.  0.  0.]]
[[  1.18951488e-04   4.93320310e-03   1.84573850e-03   3.56137636e-04
    9.92745936e-01]]
[[ 0.  0.  0.  0.  1.]]
[[  5.47588206e-06   1.77746460e-05   9.99927402e-01   2.51295160e-06
    4.68543840e-05]]
[[ 0.  0.  1.  0.  0.]]
[[  8.70592415e-01   1.22295789e-01   1.50476873e-03   2.83894860e-06
    5.60422614e-03]]
[[ 1.  0.  0.  0.  0.]]
[[  5.61861307e-05   9.99830008e-01   9.91707639e-05   3.18315165e-06
    1.14779232e-05]]
[[ 0.  1.  0.  0.  0.]]
[[  3.62178020e-04   6.89077948e-04   6.15087897e-03   9.92745101e-01
    5.27379634e-05]]
[[ 0.  0.  0.  1.  0.]]
[[  2.65477523e-02   7.20563471e-01   8.08281708e-04   7.05482089e-04
    2.51374960e-01]]
[[ 0.  1.  0.  0.  0.]]
[[  3.73913426e-05   4.62270895e-04   1.60044543e-02   9.83208597e-01
    2.87334376e-04]]
[[ 0.  0.  0.  1.  0.]]
[[  9.98944461e-01   4.77174064e-04   1.04977800e-04   6.05189553e-06
    4.67247039e-04]]
[[ 1.  0.  0.  0.  0.]]
0, test accuracy, 0.85
```

Appendix B

Parsed Test Results

```
Average Test Accuracy: 0.8875
=================
=================
=================
=================
FOLD 0.0
step, validation accuracy, training accuracy
0.0, 0.25, 0.203539619469
250.0, null, 0.230088265487
500.0, 0.05, 0.230088265487
750.0, null, 0.247787362832
1000.0, 0.15, 0.327433300885
1250.0, null, 0.557521566372
1500.0, 0.25, 0.371681044248
1750.0, null, 0.371681044248
2000.0, 0.3, 0.610618858407
2250.0, null, 0.477875628319
2500.0, 0.35, 0.73451380531
2750.0, null, 0.566371115044
3000.0, 0.35, 0.592919761062
3250.0, null, 0.61946840708
3500.0, 0.35, 0.73451380531
3750.0, null, 0.681416513274
4000.0, 0.3, 0.646018318584
4250.0, null, 0.699115610619
4500.0, 0.35, 0.814159743363
4750.0, null, 0.778761548673
5000.0, 0.35, 0.867257035398
5250.0, null, 0.787611097345
5500.0, 0.35, 0.823009292035
5750.0, null, 0.787611097345
6000.0, 0.4, 0.831858840708
6250.0, null, 0.769912
6500.0, 0.55, 0.893805681416
6750.0, null, 0.893805681416
7000.0, 0.5, 0.929203876106
7250.0, null, 0.902655230088
7500.0, 0.55, 0.938053424779
7750.0, null, 0.911504778761
8000.0, 0.55, 0.946902973451
8250.0, null, 0.973451619469
8500.0, 0.65, 0.955752522124
8750.0, null, 0.946902973451
9000.0, 0.6, 0.964602070796
9250.0, null, 0.973451619469
9500.0, 0.65, 0.982301168142
```

```
9750.0, null, 0.982301168142
10000.0, 0.65, 0.982301168142
10250.0, null, 0.973451619469
10500.0, 0.65, 0.982301168142
10750.0, null, 0.991150716814
11000.0, 0.7, 0.991150716814
11250.0, null, 0.982301168142
11500.0, 0.6, 0.982301168142
11750.0, null, 0.991150716814
12000.0, 0.7, 0.991150716814
12250.0, null, 0.991150716814
12500.0, 0.55, 0.991150716814
12750.0, null, 0.991150716814
13000.0, 0.7, 0.991150716814
13250.0, null, 1
13500.0, 0.7, 0.991150716814
13750.0, null, 0.991150716814
14000.0, 0.65, 1
14250.0, null, 0.991150716814
14500.0, 0.75, 1
14750.0, null, 1
15000.0, 0.7, 0.991150716814
15250.0, null, 1
15500.0, 0.7, 1
15750.0, null, 1
16000.0, 0.7, 1
16250.0, null, 1
16500.0, 0.7, 1
16750.0, null, 1
17000.0, 0.75, 1
17250.0, null, 1
17500.0, 0.75, 1
17750.0, null, 1
18000.0, 0.75, 1
18250.0, null, 1
18500.0, 0.75, 1
18750.0, null, 1
19000.0, 0.75, 1
19250.0, null, 1
19500.0, 0.7, 1
19750.0, null, 1
20000.0, 0.8, 1
20250.0, null, 1
20500.0, 0.6, 1
20750.0, null, 1
21000.0, 0.75, 1
21250.0, null, 1
21500.0, 0.75, 1
21750.0, null, 1
22000.0, 0.7, 1
22250.0, null, 1
22500.0, 0.75, 1
22750.0, null, 1
23000.0, 0.75, 1
```

```
23250.0, null, 1
23500.0, 0.75, 1
23750.0, null, 1
24000.0, 0.75, 1
24250.0, null, 1
24500.0, 0.75, 1
24750.0, null, 1
25000.0, 0.75, 1
25250.0, null, 1
25500.0, 0.75, 1
25750.0, null, 1
26000.0, 0.75, 1
26250.0, null, 1
26500.0, 0.7, 1
26750.0, null, 1
27000.0, 0.75, 1
27250.0, null, 1
27500.0, 0.75, 1
27750.0, null, 1
28000.0, 0.75, 1
28250.0, null, 1
28500.0, 0.7, 1
28750.0, null, 1
29000.0, 0.75, 1
29250.0, null, 1
29500.0, 0.75, 1
29750.0, null, 1
30000.0, 0.8, 1
30250.0, null, 1
30500.0, 0.7, 1
30750.0, null, 1
31000.0, 0.8, 1
31250.0, null, 1
31500.0, 0.75, 1
31750.0, null, 1
32000.0, 0.65, 1
32250.0, null, 1
32500.0, 0.75, 1
32750.0, null, 1
33000.0, 0.75, 1
33250.0, null, 1
33500.0, 0.7, 1
33750.0, null, 1
34000.0, 0.8, 1
34250.0, null, 1
34500.0, 0.75, 1
34750.0, null, 1
35000.0, 0.65, 1
35250.0, null, 1
35500.0, 0.75, 1
35750.0, null, 1
36000.0, 0.8, 1
36250.0, null, 1
36500.0, 0.85, 1
```

```
36750.0, null, 1
37000.0, 0.7, 1
37250.0, null, 1
37500.0, 0.75, 1
37750.0, null, 1
38000.0, 0.7, 1
38250.0, null, 1
38500.0, 0.75, 1
38750.0, null, 1
39000.0, 0.8, 1
39250.0, null, 1
39500.0, 0.75, 1
39750.0, null, 1
40000.0, 0.8, 1
40250.0, null, 1
40500.0, 0.8, 1
40750.0, null, 1
41000.0, 0.75, 1
41250.0, null, 1
41500.0, 0.75, 1
41750.0, null, 1
42000.0, 0.8, 1
42250.0, null, 1
42500.0, 0.8, 1
42750.0, null, 1
43000.0, 0.8, 1
43250.0, null, 1
43500.0, 0.75, 1
43750.0, null, 1
44000.0, 0.8, 1
44250.0, null, 1
44500.0, 0.85, 1
44750.0, null, 1
45000.0, 0.8, 1
45250.0, null, 1
45500.0, 0.75, 1
45750.0, null, 1
46000.0, 0.8, 1
46250.0, null, 1
46500.0, 0.8, 1
46750.0, null, 1
47000.0, 0.8, 1
47250.0, null, 1
47500.0, 0.85, 1
47750.0, null, 1
48000.0, 0.8, 1
48250.0, null, 1
48500.0, 0.8, 1
48750.0, null, 1
49000.0, 0.8, 1
49250.0, null, 1
49500.0, 0.8, 1
49750.0, null, 1
=================
FOLD 2.0
step, validation accuracy, training accuracy
0.0, 0.1, 0.221238716814
250.0, null, 0.327433300885
500.0, 0.3, 0.353981946903
750.0, null, 0.34513239823
1000.0, 0.2, 0.327433300885
1250.0, null, 0.469026079646
1500.0, 0.2, 0.353981946903
1750.0, null, 0.362831495575
2000.0, 0.4, 0.495574725664
2250.0, null, 0.415928787611
2500.0, 0.3, 0.424778336283
2750.0, null, 0.513273823009
3000.0, 0.45, 0.592919761062
3250.0, null, 0.575220663717
3500.0, 0.4, 0.584070212389
3750.0, null, 0.610618858407
4000.0, 0.65, 0.628317955752
4250.0, null, 0.61946840708
4500.0, 0.7, 0.769912
4750.0, null, 0.752212902655
5000.0, 0.5, 0.646018318584
5250.0, null, 0.73451380531
5500.0, 0.45, 0.637168769912
5750.0, null, 0.769912
6000.0, 0.55, 0.769912
6250.0, null, 0.716814707965
6500.0, 0.75, 0.840708389381
6750.0, null, 0.831858840708
7000.0, 0.65, 0.876106584071
7250.0, null, 0.902655230088
7500.0, 0.55, 0.787611097345
7750.0, null, 0.938053424779
8000.0, 0.75, 0.920354327434
8250.0, null, 0.955752522124
8500.0, 0.75, 0.938053424779
8750.0, null, 0.938053424779
9000.0, 0.8, 0.955752522124
9250.0, null, 0.929203876106
9500.0, 0.8, 0.929203876106
9750.0, null, 0.973451619469
10000.0, 0.75, 0.964602070796
10250.0, null, 0.973451619469
10500.0, 0.8, 0.982301168142
10750.0, null, 1
11000.0, 0.9, 0.982301168142
11250.0, null, 1
11500.0, 0.8, 0.991150716814
11750.0, null, 0.991150716814
12000.0, 0.85, 0.982301168142
12250.0, null, 0.982301168142
12500.0, 0.85, 0.973451619469
12750.0, null, 0.982301168142
```

```
13000.0, 0.95, 1
13250.0, null, 0.991150716814
13500.0, 0.95, 0.991150716814
13750.0, null, 0.991150716814
14000.0, 0.9, 1
=================
FOLD 1.0
step, validation accuracy, training accuracy
0.0, 0.25, 0.194690070796
250.0, null, 0.221238716814
500.0, 0.15, 0.292035106195
750.0, null, 0.38053059292
1000.0, 0.1, 0.451326982301
1250.0, null, 0.442477433628
1500.0, 0.05, 0.30973420354
1750.0, null, 0.513273823009
2000.0, 0.05, 0.34513239823
2250.0, null, 0.398229690265
2500.0, 0.2, 0.504424274336
2750.0, null, 0.504424274336
3000.0, 0.4, 0.557521566372
3250.0, null, 0.566371115044
3500.0, 0.3, 0.628317955752
3750.0, null, 0.681416513274
4000.0, 0.4, 0.663717415929
4250.0, null, 0.73451380531
4500.0, 0.35, 0.761062451327
4750.0, null, 0.761062451327
5000.0, 0.35, 0.761062451327
5250.0, null, 0.769912
5500.0, 0.4, 0.849557938053
5750.0, null, 0.840708389381
6000.0, 0.55, 0.876106584071
6250.0, null, 0.858407486726
6500.0, 0.45, 0.893805681416
6750.0, null, 0.876106584071
7000.0, 0.5, 0.929203876106
7250.0, null, 0.902655230088
7500.0, 0.45, 0.876106584071
7750.0, null, 0.955752522124
8000.0, 0.45, 0.973451619469
8250.0, null, 0.973451619469
8500.0, 0.65, 0.982301168142
8750.0, null, 0.938053424779
9000.0, 0.6, 0.973451619469
9250.0, null, 0.991150716814
9500.0, 0.65, 0.991150716814
9750.0, null, 0.955752522124
10000.0, 0.55, 0.991150716814
10250.0, null, 0.982301168142
10500.0, 0.55, 0.991150716814
10750.0, null, 1
11000.0, 0.75, 1
11250.0, null, 0.991150716814
```

```
11500.0, 0.55, 0.991150716814
11750.0, null, 1
12000.0, 0.8, 1
12250.0, null, 0.991150716814
12500.0, 0.65, 0.991150716814
12750.0, null, 1
13000.0, 0.8, 1
13250.0, null, 0.991150716814
13500.0, 0.85, 0.991150716814
13750.0, null, 1
14000.0, 0.85, 1
14250.0, null, 0.991150716814
14500.0, 0.7, 1
14750.0, null, 1
15000.0, 0.8, 1
15250.0, null, 1
15500.0, 0.65, 1
15750.0, null, 0.991150716814
16000.0, 0.75, 1
16250.0, null, 1
16500.0, 0.85, 1
16750.0, null, 1
17000.0, 0.8, 1
17250.0, null, 1
17500.0, 0.8, 1
17750.0, null, 1
18000.0, 0.85, 1
18250.0, null, 1
18500.0, 0.8, 1
18750.0, null, 1
19000.0, 0.8, 1
19250.0, null, 1
19500.0, 0.8, 1
19750.0, null, 1
20000.0, 0.85, 1
20250.0, null, 1
20500.0, 0.75, 1
20750.0, null, 1
21000.0, 0.7, 1
21250.0, null, 1
21500.0, 0.65, 1
21750.0, null, 1
22000.0, 0.8, 1
22250.0, null, 1
22500.0, 0.85, 1
22750.0, null, 1
23000.0, 0.85, 1
23250.0, null, 1
23500.0, 0.85, 1
23750.0, null, 1
24000.0, 0.8, 1
24250.0, null, 1
24500.0, 0.85, 1
24750.0, null, 1
```

```
25000.0, 0.85, 1
25250.0, null, 1
25500.0, 0.85, 1
25750.0, null, 1
26000.0, 0.8, 1
26250.0, null, 1
26500.0, 0.85, 1
26750.0, null, 1
27000.0, 0.85, 1
27250.0, null, 1
27500.0, 0.85, 1
27750.0, null, 1
28000.0, 0.9, 1
28250.0, null, 1
28500.0, 0.85, 1
=================
FOLD 3.0
step, validation accuracy, training accuracy
0.0, 0.2, 0.230088265487
250.0, null, 0.203539619469
500.0, 0.35, 0.336282849558
750.0, null, 0.30973420354
1000.0, 0.3, 0.362831495575
1250.0, null, 0.424778336283
1500.0, 0.35, 0.34513239823
1750.0, null, 0.575220663717
2000.0, 0.35, 0.601769309735
2250.0, null, 0.504424274336
2500.0, 0.35, 0.530972920354
2750.0, null, 0.592919761062
3000.0, 0.35, 0.513273823009
3250.0, null, 0.592919761062
3500.0, 0.35, 0.610618858407
3750.0, null, 0.61946840708
4000.0, 0.25, 0.699115610619
4250.0, null, 0.796460646018
4500.0, 0.35, 0.743363353982
4750.0, null, 0.761062451327
5000.0, 0.4, 0.725664256637
5250.0, null, 0.716814707965
5500.0, 0.5, 0.752212902655
5750.0, null, 0.814159743363
6000.0, 0.4, 0.876106584071
6250.0, null, 0.814159743363
6500.0, 0.5, 0.884956132743
6750.0, null, 0.867257035398
7000.0, 0.55, 0.840708389381
7250.0, null, 0.858407486726
7500.0, 0.45, 0.831858840708
7750.0, null, 0.884956132743
8000.0, 0.65, 0.946902973451
8250.0, null, 0.938053424779
8500.0, 0.6, 0.920354327434
8750.0, null, 0.946902973451
```

```
9000.0, 0.7, 0.955752522124
9250.0, null, 0.973451619469
9500.0, 0.65, 0.964602070796
9750.0, null, 0.991150716814
10000.0, 0.65, 0.982301168142
10250.0, null, 0.982301168142
10500.0, 0.7, 0.973451619469
10750.0, null, 0.991150716814
11000.0, 0.75, 1
11250.0, null, 0.991150716814
11500.0, 0.7, 1
11750.0, null, 0.991150716814
12000.0, 0.75, 1
12250.0, null, 1
12500.0, 0.8, 1
12750.0, null, 1
13000.0, 0.75, 1
13250.0, null, 1
13500.0, 0.75, 1
13750.0, null, 1
14000.0, 0.8, 1
14250.0, null, 1
14500.0, 0.75, 1
14750.0, null, 1
15000.0, 0.9, 1
15250.0, null, 1
15500.0, 0.85, 1
15750.0, null, 1
16000.0, 0.8, 1
16250.0, null, 1
16500.0, 0.75, 1
16750.0, null, 1
17000.0, 0.85, 1
17250.0, null, 1
17500.0, 0.8, 1
17750.0, null, 1
18000.0, 0.85, 1
18250.0, null, 1
18500.0, 0.75, 1
18750.0, null, 1
19000.0, 0.8, 1
19250.0, null, 1
19500.0, 0.8, 1
19750.0, null, 1
20000.0, 0.8, 1
20250.0, null, 1
20500.0, 0.8, 1
20750.0, null, 1
21000.0, 0.85, 1
21250.0, null, 1
21500.0, 0.85, 1
21750.0, null, 1
22000.0, 0.85, 1
22250.0, null, 1
```

```
22500.0, 0.8, 1
22750.0, null, 1
23000.0, 0.85, 1
23250.0, null, 1
23500.0, 0.9, 1
23750.0, null, 1
24000.0, 0.9, 1
24250.0, null, 1
24500.0, 0.9, 1
24750.0, null, 1
25000.0, 0.85, 1

HA Correct : (17/17) 100.00%, First or second : (17/17) 100.00%
[]
SA Correct : (14/17) 82.35%, First or second : (15/17) 88.24%
['DI', 'HA', 'HA']
AN Correct : (15/17) 88.24%, First or second : (15/17) 88.24%
['DI', 'DI']
DI Correct : (12/13) 92.31%, First or second : (12/13) 92.31%
['AN']
FE Correct : (14/16) 87.50%, First or second : (14/16) 87.50%
['SA', 'DI']
```