Exploring Optical Character regonition with Artificial Neural Networks and their potential in simplified CAPTCHA regonition.

Author: Carl Chang
Student ID: 17971871

***Abstract*—This paper explores a feed forward artificial neural network for optical character recognition (OCR), with the aim to determine distorted image characters displayed within CAPTCHAs. The problem will be explored through the design and development of an artificial neural network model to correctly classify visual patterns represented by simplified character matrices. The model will be tested against character matrices with removed pattern features, then tested against character matrices with both removed features and added noise. We will also explore the degradation of the Artificial Neural Network's ability to predict visual patterns as a consequence of pattern distortions.**

Keywords — Artificial Neural Network, Back Propagation Algorithm, Stochastic gradient descent, Hidden-layer, optical character recognition.

I. Introduction

In recent years, many websites have imbedded CAPTCHA (completely automated public Turing test to tell computer and humans apart) technology (Ahn, Blum, Hopper, & Langford, 2003). The test aims to distinguish between humans and computer. It involves a human to correctly determine a sequence of distorted alphanumeric characters displayed on the screen, and then requiring the user to input the perceived characters in sequence. This test has been adopted to prevent bot scripts from executing iterative malicious behaviour on websites. The test is intended to be easily recognized by humans, but difficult for machines to defeat.

A possible solution for machines to defeat CAPTCHA, is by the implementation of an optical character recognition system (OCR). These systems recognize and classify images by extracting the features of their visual representation. The collection of extracted features are termed feature vectors and used to train a recognition classification model. OCR is widely used in many fields including postal services, where machines will recognize images of hand-written text of addresses and sort them accordingly.

A suitable underlying model for an OCR system is an artificial neural network (ANN), more specifically the multilayer perceptron (MLP). Analogous to the human brain, the ANN model architecture utilizes densely connected units called neurons to learn by adjusting the weights (dendrites) at each node, with an optimization technique, such as back-propagation (also known as stochastic gradient descent). The ANN enables computers to learn the visual features without the need to hard code every possible pattern into the system, which takes an unfeasible amount of time.

The aim of this report is to investigate how effective an ANN is at solving a simplified CAPTCHA problem. The simplification falls upon the ability to recognise *individual characters* based on their subsequently distorted visual patterns, rather than a distorted *full sequence* of characters.

	A	B	C	D	E	F	G
1	0	0	1	1	1	0	0
2	0	1	1	1	1	1	0
3	1	1	0	0	0	1	1
4	1	1	1	1	1	1	1
5	1	1	0	0	0	1	1
6	1	1	0	0	0	1	1
7	1	1	0	0	0	1	1

	A	B	C	D	E	F	G
1			x	x	x		
2		x	x	x	x	x	
3	x	x				x	x
4	x	x	x	x	x	x	x
5	x	x				x	x
6	x	x				x	x
7	x	x				x	x

Figure 1 – Generating a visual representation of a letter within a 7x7 character matrix for the letter 'A'. Character matrix (left). Visualizing the character matrix (Right).

II. Generating the Dataset

The CAPTCHA systems usually displaying a sequence of alphanumeric characters. However, to demonstrate a fundamental level of understanding for ANN in OCR, the scope of the problem will be simplified. This paper will explore only individual upper-case characters of the English language represented by a 7x7 matrix (*Figure 1*).

A. Feature extraction

To create the initial dataset, a series of upper-case English language letters were planned and drawn out on a 7x7 matrix according to the unique visual features that define each letter (*Figure 1 – right*). Where a defining feature would be present in the matrix, it is denoted with 1, and when there is an absence of a defining feature, with 0 (*Figure 1 – left*).

B. Preparing the training set

For each letter, the matrices are converted to feature vectors. Given that 7x7 matrices are used, the feature-vectors will have 49 columns. Each coordinate on the letter matrix represents a column for the training set. For example, on *Figure 1*, the first coordinate will be a column labelled A1, with value 0 in the training set. The second coordinate will be labelled A2, with value 0. This is repeated until the last coordinate which will be a column labelled G7, with value 1.

Each individual character is represented in a single row. The final column is used as the class-label (*Figure 2*). This is important for supervised learning. The collection of all the vectorized characters (A to Z) form the final training set. Feature vectors were exported as a csv file from google sheets.

	A1	B1	C1	D1	E1	F1	G1	A2	B2	C2	...	F6	G6	A7	B7	C7	D7	E7	F7	G7	label
0	0	0	1	1	1	0	0	0	1	1	...	1	1	1	1	0	0	0	1	1	A
1	1	1	1	1	1	1	0	1	0	0	...	0	1	1	1	1	1	1	1	1	B
2	0	0	0	1	1	1	1	0	1	1	...	0	0	0	0	1	1	1	1	1	C

Figure 2 – Feature vector representations of character matrices A, B, C.

C. *Preparing the test set (feature removal)*

The aim of this paper is to investigate the performance of the ANN on visual pattern distortions. Creating the test set with distorted features involves randomly removing the feature signals, where the value 1 is replaced with a 0 one at a time.

When a signal is removed, it is removed collectively, such that every character (row) will have a feature removed at the same time. It was determined that the maximum features to be removed collectively is 10, as the character 'Y' is made up of only 10 features.

The original training set was read into Pandas (Python data manipulation library) as a csv file, where a custom script was used to randomly and collectively remove features. This script ran recursively until 10 test sets were prepared. Each set differed on the number of features removed.

As the feature removal is randomized, there may be instances where the removal of a feature in a particular position may either significantly or insignificantly affect the model prediction for that label. Therefore, a sample of 10 for varying removal positions is taken for each set that differed on the number of features removed. Note that each sample is a set of altered vectors for each letter A to Z.

The above resulted in 100 test sets that represents visually distorted character features for varying feature removal numbers and with different removal positions.

D. *Further tests (feature removal or noise introduction)*

Similar to preparing the test set, instead of only removing features; noise is also introduced by randomly changing non-features with a value of 0 to 1 on the original training set.

The custom script randomly chooses a position in each row (per character feature vector) and changes 0 to 1 or if 1 to 0. Each variation of the dataset on positions of alteration and the number of alterations made were saved.

This data preparation resulted in 150 datasets, where up to 15 alterations were made, with 10 samples of differing alteration positions.

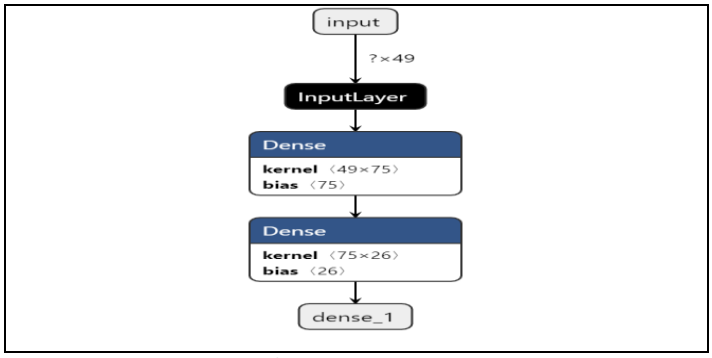

Figure 3 – High level view of the TensorFlow Neural Network Architecture plot visualized with Netron.

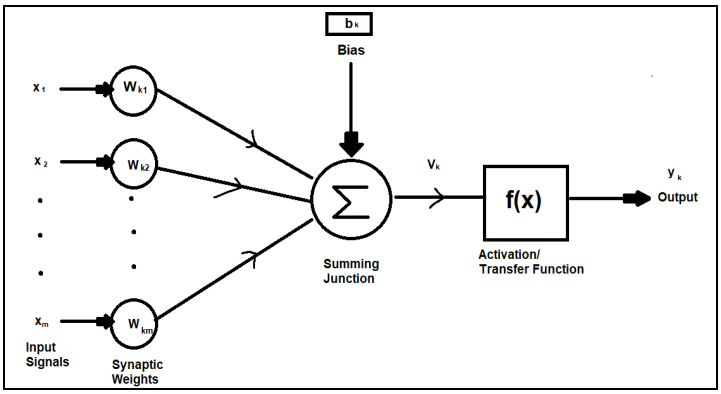

Figure 4 – ANN neuron simple architecture overview.

III. Artificial Nerual Network Architecture

The ANN is built and compiled using the Python programming language with the TensorFlow 2.0 package.

A. *ANN overview*

A simple representation of an ANN's neuron is illustrated in *Figure 4*. It shows how input signals interact with the weights and biases to form an output. Weights are input reflect the importance of an input and are assigned to each neuron synapse, then later optimized during training. The bias is a constant value that provides an input to the activation function when the weighted sum of input is zero.

$$v = \sum_{i=1}^{m} w_i \cdot x_i + b \tag{1}$$

Each layer receives *v*, the weighted sum of input from the previous layer. The weighted sum of inputs is given by equation (1). Where *w* is the weight, *x* is the input and *b* is the bias. Given *v*, the activation function *f(v)* acts as a gate which determines if the corresponding neuron will activate. There are many activation functions, each with their unique thresholds for firing.

B. *Layers*

Figure 3 displays a high-level architecture of the Neural Network, where a dense input layer takes an input shape of size '?×49'. The question mark represents the number of rows (flexible size) and the '49' represents the number of columns (fixed size).

The size of the (75) hidden layer size is calculated by adding the number of rows (26) and feature columns (49) in the training set. There is a dense hidden layer with 75 bias. which receives 49×79 inputs. Which amounts to a total of 3675 adjustable weights between the hidden layer and the input layer. The transfer function used at this stage is Rectified Linear unit (ReLU).

The last layer (output) has 26 neuron nodes. It is controlled by a Softmax activation function. This layer produces 26 outputs, each representing a prediction probability for each label (A-Z).

C. Optimizer

Adaptive moment estimation (*Adam*) was chosen as the optimizer for this ANN model. *Adam* is a type of stochastic gradient descent (SGD) optimization algorithm for machine learning (Kingma & Lei Ba, 2015). Unlike traditional SGD, it computes the moving averages of the gradient, in contrast with just the gradient itself, in combination with momentum. It has proven to be computationally inexpensive and converges quickly during back-propagation for large-scale problems with high-dimensionality datasets. The paper (Kingma & Lei Ba, 2015) also suggests a good set of default parameters (*Table 1*) to use with *Adam*.

Parameter	Description	Value
α	Learning rate.	0.001
β_1, β_2	Exp decay rate for the first and second moment.	0.9, 0.999
ε	Constant for numerical stability.	1e-7

Table 1 – Default parameters used for the Adam optimizer for back propagation.

D. Activation functions

Logistic sigmoid and ReLU were considered as activation functions for the hidden layer. They are both non-linear activation functions allowing the ANN to learn complex structures in the dataset. Below explains the rationale.

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}. \tag{2}$$

Logistic sigmoid activation function given by equation (2). This function has an output value between 0 and 1, which normalizes neuron outputs and gives clear predictions when the input *x* is strongly negative or strongly positive.

However, there is a disadvantage where the function suffers from the vanishing gradient problem. This is where strongly positive or strongly negative inputs are set to 1 or 0 respectively, when too many strongly positive or negatives appear as input values, and we observe saturation at 1 and 0.

$$f(x) = x^+ = \max(0, x) \tag{3}$$

ReLU is an activation function given by equation (3). The output value is equal to the input if it is not negative, otherwise 0. This function may seem non-linear; however, a derivative exists which enables back propagation (learning). It is computationally inexpensive and can converge faster than logistic sigmoid as it does not suffer from the vanishing gradient problem (Glorot, Bordes, & Bengio, 2011). Due to ReLU's ability to converge quickly, run efficiently and not saturate, and thus the better candidate to control the hidden layer for this ANN.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K \tag{4}$$

Softmax is another activation function used for this ANN at the output layer, given by equation (4). It is primarily used for multi-class classification problem, by normalizing the outputs (from the ReLU controlled layer) for each class to give a probability of the input values being in a specific class. The class with the highest probability is selected as the predicted class label.

E. Epochs

The maximum number of epochs refer to the maximum number of cycles of backpropagation weight adjustments the model undergoes during training. TensorFlow allows the implementation of an early stop where if no further improvement (in a specified metric *e.g.* accuracy) is observed for *n* number of epochs, the training process will halt. For the training process, the maximum number of epochs is set at 1000 with an early stop of 200 epochs.

IV. Experiments and Results

A. Training

During training, the model converged after 320 epochs. The evaluation metrics for accuracy, loss and cross-entropy were tracked with each epoch during training (*Figure 5*). The mean absolute error (MAE) is 12.47, and the mean squared error (MSE) is 211.58. After convergence, the cross-entropy fell to 0, and the accuracy reaches 100 percent. This means the trained model can predict all 26 labels correctly on the training data set and is reflected by the confusion matrix (*Figure 6*).

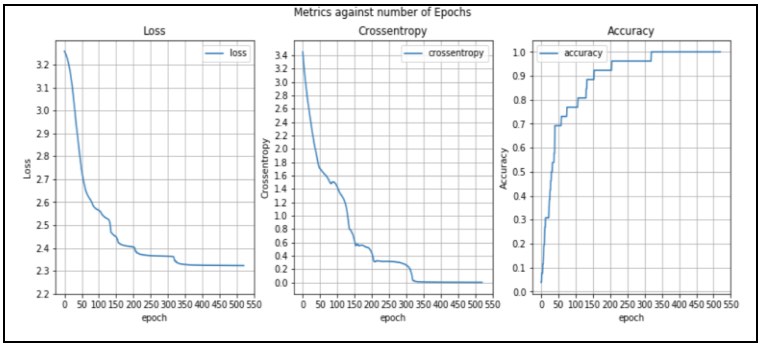

Figure 5 – Evaluation metrics for *Loss, Cross-entropy* and *Accuracy* against the number of Epochs during training

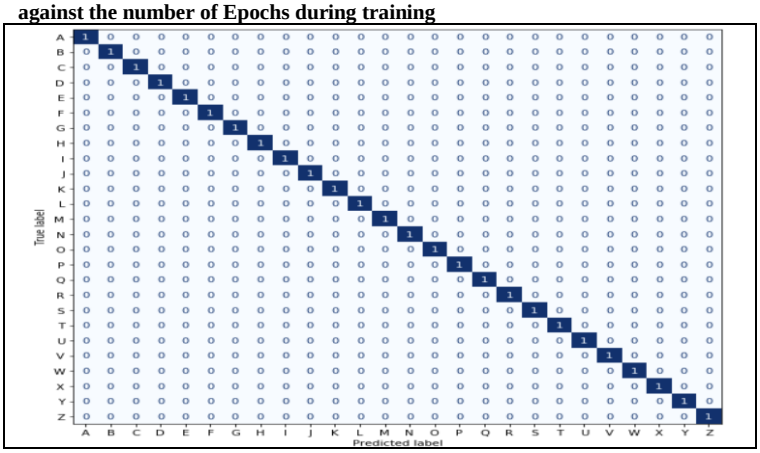

Figure 6 – Confusion matrix displaying perfect ANN prediction on training set. True label is on the y-axis and predicted label on the x-axis.

removed_features	mean	median	std	min	max
1	0.998077	1.000000	0.008425	0.961538	1.0
2	0.995769	1.000000	0.012095	0.961538	1.0
3	0.993462	1.000000	0.014520	0.961538	1.0
4	0.991154	1.000000	0.018011	0.923077	1.0
5	0.984615	1.000000	0.023829	0.923077	1.0
6	0.975385	1.000000	0.031174	0.884615	1.0
7	0.963462	0.961538	0.036003	0.846154	1.0
8	0.949615	0.961538	0.037356	0.846154	1.0
9	0.931923	0.923077	0.046980	0.807692	1.0
10	0.910000	0.923077	0.052212	0.769231	1.0

Figure 7 - Accuracy metric summary across 10 degrees of features being removed.

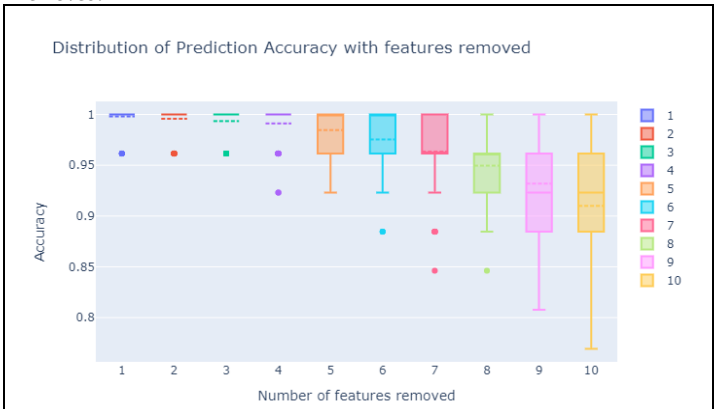

Figure 8 - Distribution of prediction accuracy with number of removed features.

B. *Testing on removed features*

As previously stated, depending on feature removal position, there will be varying effects on the ability of the ANN to predict the labels. Hence a sample size of 10 is taken for various numbers (1 to 10) of randomly removed features. The ANN model makes predictions for the class label for all 100 datasets. The results are grouped by their respective number of features removed.

Accuracy is the most suitable metric to determine the predictive ability of the ANN in a multi-class classification problem with equal number of examples in each class. The accuracy for each degraded dataset is computed and summarised in *Figure 7*. By observing the large differences between the minimum accuracy and the maximum accuracy, it is evident that removal of features in certain positions can be more sensitive than in others. With reference to *Figure 8, the* overall trend shows that with increased removal of features, the accuracy of prediction degrades. An accuracy prediction median of 1.0 and mean of 0.99 is maintained for the first 3 test sets. It is also observed that on average, degradation of accuracy does not occur until 5 features have been removed.

The confusion matrix (*Figure 9*) is computed for the ANN's ability to classify all 2600 feature altered examples within the 100 test sets from this experiment, with an accuracy of 0.969. It is observed that the label '*Q*' is most frequently misclassified as '*O*' 111 times out of 1000 (11%).

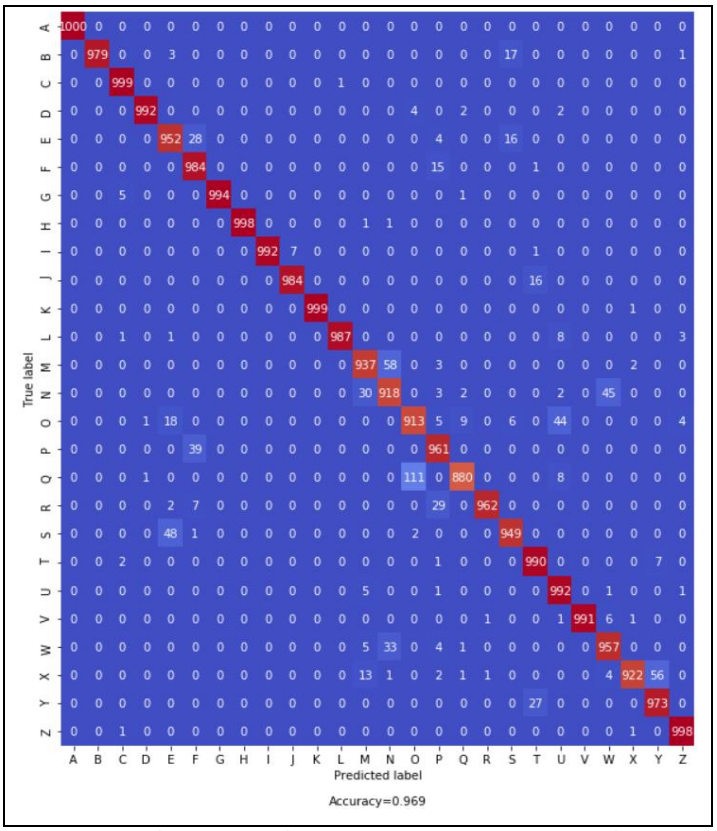

Figure 9 – Confusion matrix for all 2600 examples with removed features, across 10 degrees of features removed.

C. *Testing on disrupted features (random feature removal or random addition of noise)*

Like the previous experiment, 10 samples for each degree of disruption (1 to 15) is analysed based on its prediction accuracy. However, the maximum disruption of features has been increased from 10 to 15 (no longer restricted by 'Y' with only 10 features).

Figure 10 summarises the prediction accuracies for each degree of disrupted features. It is evident that the overall (*mean* and *median*) accuracy degrades with increased disruption. The uncertainty, measured by the standard deviation, increases with increasing disruption.

disrupted_features	mean	median	std	min	max
1	0.997692	1.000000	0.009180	0.961538	1.000000
2	0.996154	1.000000	0.011597	0.961538	1.000000
3	0.993077	1.000000	0.016743	0.923077	1.000000
4	0.989615	1.000000	0.022443	0.884615	1.000000
5	0.981538	1.000000	0.027595	0.846154	1.000000
6	0.973077	0.961538	0.031641	0.846154	1.000000
7	0.965385	0.961538	0.036875	0.846154	1.000000
8	0.948077	0.961538	0.048704	0.807692	1.000000
9	0.933077	0.923077	0.046264	0.807692	1.000000
10	0.906923	0.923077	0.059929	0.692308	1.000000
11	0.878846	0.884615	0.066139	0.730769	1.000000
12	0.848846	0.846154	0.068335	0.692308	1.000000
13	0.806923	0.807692	0.073948	0.615385	1.000000
14	0.776538	0.769231	0.072254	0.576923	1.000000
15	0.745000	0.730769	0.077822	0.538462	0.923077

Figure 10 – Accuracy metric summary across varying degrees of feature disruption (removing feature or adding noise).

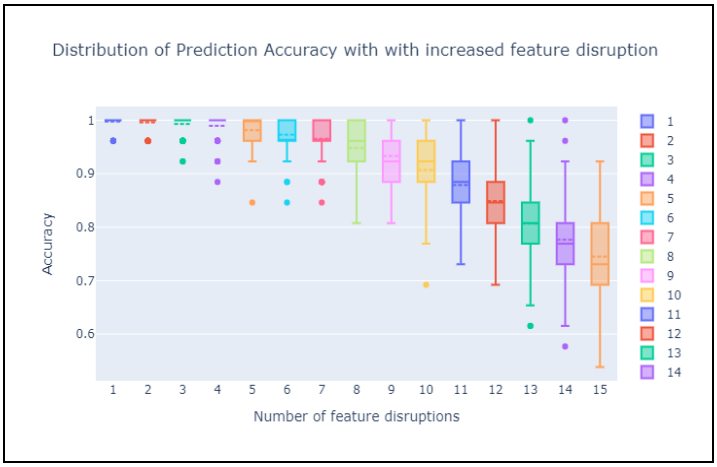

Figure 11 - Distribution of prediction accuracy with number of disrupted features (removing feature or adding noise).

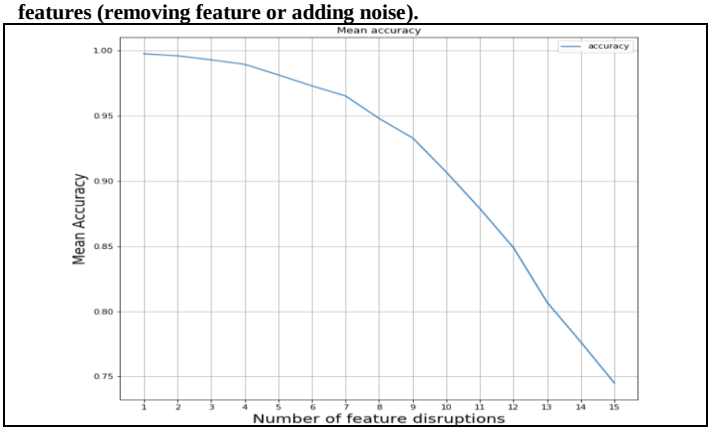

Figure 12 – ANN's Mean prediction accuracy with increasing feature disruption.

Figure 11 visually represents the effects of disrupted features on the prediction accuracy of the ANN. Significant signs of degradation is observed at 5 feature disruptions. The accuracy distribution falls with increased disruptions. At 15 disruptions, the accuracy falls to a lowest point of 0.538, and a highest point of 0.923.

It was also found that when an empty vector (when all 49 feature columns are 0) is passed into the ANN for prediction, it recognizes this as the letter 'Y'. This is interesting because 'Y' is the class represented with the least number of visual features.

V. Discussion

A. Training

Training the model was quite successful, as it converged quickly, and can predict all 26 capital letters with 100% accuracy. Observing the metric graphs, it is evident that convergence has occurred during training.

B. Test – feature removal

It is evident that feature removal position at random causes variations in prediction accuracy on groups with the same number of features removed. The greatest range is observed in the 10th group (between 1.0 and 0.91). Hence this testing method is appropriate.

The trained ANN model has shown robustness, where on average it can predict labels with 0.99 accuracy with minor (up to group 4) features removed and 0.91 with major (from group 5 to 10) features removed. At its best, it can accurately predict all characters with 1.0 accuracy across all 10 removal groups (insignificant removal positions).

The test has shown that the ANN's accuracy for prediction gracefully degrades when the number of features removed is increase. Accuracy begins to degrade at a greater rate with respect to the number of removed features after 4 features have been removed.

C. Test – feature removal with noise introduction

This test also shows graceful degradation of accuracy with increasing number of feature disruptions. The first 4 groups are observed to have no significant decrease in accuracy (the dots represent outliers). On *Figure 12*, it is observed that the mean accuracy gradient with respect to the number of feature disruptions becoming steeper at disruption group 4, then 7, 9, and 12. It is observed that at 15 disruptions, the maximum accuracy within the group cannot achieve 1.0 accuracy anymore.

VI. Conclusion

The experiments have demonstrated ANN as a suitable candidate for performing OCR on CAPTCHA. It is observed that ANN can learn to accurately classify and interpret all 26 upper case English characters based their respective (unchanged) visual patterns. Experimental evidence has shown the ANN to exhibit robustness to the removal of minor features as well as noise introduction. Prediction accuracy is maintained on average above 0.9 with minor (less than 8) alterations. The prediction accuracy degrades gracefully in face of increased alterations to the datasets.

This paper has only explored training an ANN to recognise 26 English upper-case letters. In order to defeat a CAPTCHA, further ANN experiments should be explored to include alpha-numeric characters, various fonts, larger visual matrices, geometric translations, full character sequences, and convoluted neural network architectures.

VII. References

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *International Conference on Artificial Intelligence and Statistics (AISTATS). 15*, pp. 315-323. Fort Lauderdale, FL, USA: JMLR: W&CP.

Kingma, D. P., & Lei Ba, J. (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. *ICLR* (pp. 1-15). San Diego: International Conference on Learning Representations. Retrieved from https://arxiv.org/pdf/1412.6980.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM, 60*(6), 84-90. doi:https://doi-org.ezproxy.aut.ac.nz/10.1145/3065386

Yi, Z., Zhang, L., Yu, J., & Tan, K. K. (2009). Permitted and Forbidden Sets in Discrete-Time Linear Threshold Recurrent Neural Networks. *IEEE Transactions on Neural Networks, 20*(6), 952-963.