Mathematical Foundations of Machine Learning for Detecting Counterfeit Currency

Jiayi Li

International School of Beijing, Beijing, China

Abstract. Due to advancements in printing technology and digital scanning tools, eliminating counterfeit bills has become increasingly challenging. As a powerful solution, machine learning has provided an efficient and automatic manner to facilitate fake bill detection. Here, we focus on the mathematical foundations of machine learning, specifically supervised learning, to develop a neural network capable of detecting fake bills. Using a dataset of dollar bills, key statistic features of images of the bills, e.g., variance and skewness, are analyzed. We unveil how these features are processed through a neural network that leverages layers, nodes, activation functions, and binary cross-entropy errors for accuracy in detection. As a practical example, we create and train a neural network model that performs well in counterfeit detection with a superior high accuracy. By reinforcing core mathematical concepts essential to building such systems, we demonstrate the flexibility and power of machine learning techniques, which may promise the same level of excellence in other applications, such as credit card fraud prevention, identity theft detection, and anomaly detection in financial transactions.

Keywords: Machine learning, supervised learning, neural network, counterfeit bill detection.

1. Introduction

Machine learning has become a building block of modern artificial intelligence, facilitating many applications that impact our daily lives, such as spam email filters and recommendation systems on streaming platforms and search engines [1, 2]. Among its many applications, counterfeit currency detection is particularly relevant as the technology evolves to help combat financial fraud, which has long challenged governments and financial institutions [3]. The growing sophistication of counterfeiting techniques calls for advanced technological solutions. With its scalability and accuracy, machine learning offers an efficient approach to detecting counterfeit bills, supplementing traditional methods reliant on human inspection [4].

Identification of counterfeit banknote detection has primarily focused on utilizing multiple imaging sensors, e.g., ultraviolet (UV) [5] and infrared (IR) [6], to detect distinct features of banknotes, which are applied in automated systems like vending machines. Such conventional detection-based methods have been effective in recognizing banknotes under various challenging conditions [7]. More recently, classification-based methods leverage visible-light images and classifiers like support vector machines (SVM) and convolutional neural networks (CNNs) to identify counterfeit currency [8].

This study aims to explore the mathematical foundations of machine learning and to showcase their application in detecting counterfeit currency. Specifically, we utilize supervised learning, a subfield of machine learning that relies on labeled data to train predictive models. The dataset used for this research, obtained from the UC Irvine Machine Learning Repository [9], contains statistical features of banknotes extracted using image processing techniques such as wavelet transformation [10]. The dataset consists of four key features: variance, skewness, virtuosity, and entropy, all of which are used to classify bills as either authentic or counterfeit. The mathematical methods employed in this study focus on linear algebra, logistic regression, neural networks, and error minimization techniques like binary cross-entropy. Two machine learning models are examined: logistic regression and neural networks. Logistic regression is a simple yet powerful tool for binary classification tasks, while neural networks offer a more flexible and robust framework, especially for complex datasets. We investigate how these models process the bill features to generate predictions about their authenticity. Logistic regression provides a straightforward approach, where the features are linearly

combined and passed through a sigmoid function to produce a probability of the bill being real or fake. On the other hand, neural networks use a multi-layered architecture, allowing for more nuanced and sophisticated decision-making processes.

The results demonstrate the effectiveness of both logistic regression and neural networks in detecting counterfeit bills, with neural networks showing superior accuracy due to their ability to model more complex patterns in the data. Our results illustrate how the mathematical principles of machine learning can be effectively applied to a real-world problem, offering a pathway for further research and application in other domains of fraud detection and financial security. Concepts such as the sigmoid and ReLU activation functions, binary cross-entropy as a measure of error, and optimization of weights and biases form the core of the mathematical framework for building, training, and applying machine learning models to solve real-world problems.

The future implications of this work are significant for both the financial sector and the global economy. Adopting machine learning in counterfeit detection can substantially reduce the cost for governments and financial institutions by reducing the reliance on manual inspection processes and mitigating the circulation of counterfeit currency. Additionally, improving the detection of counterfeit money can help stabilize economies by reinforcing trust in currency systems and reducing the risks of inflation and financial instability. Machine learning's potential in automating fraud detection extends beyond counterfeit money, with potential in other areas, such as credit card fraud prevention, identity theft protection, and anomaly detection in financial transactions [11].

2. Methodology

2.1. Key Concepts

2.1.1 Dataset

The data used in this study consists of statistical features extracted from images of banknotes obtained from a banknote authentication dataset provided by the UC Irvine Machine Learning Repository [9]. Each bill was photographed, and after applying image processing techniques, such as the wavelet transform explained in [10], four numerical values were extracted. For brevity, we will not delve into the details of these techniques. In essence, the dataset for each bill includes these four numerical values along with a label indicating whether the bill is genuine or counterfeit. Table 1 presents a sample of this tabular data, with each row representing a data point, also referred to as an example in machine learning terminology. In this context, each example corresponds to a bill. Each bill is characterized by two types of information: the first type includes the bill's *features* (variance, skewness, virtuosity, and entropy), and the second type is the *label*, which indicates whether the bill is authentic or counterfeit (as shown in the last column).

Table 1. Example data of 4 out of 1372 bills

Variance	Skewness	Virtuosity	Entropy	Authentic
3.62160	8.66610	-2.8073	-0.44699	Yes
4.54590	8.16740	-2.4586	-1.46210	Yes
-3.56370	-8.38270	12.3930	-1.28230	No
-2.54190	-0.65804	2.6842	1.19520	No

2.1.2 Supervised learning

The problem addressed in this article falls under supervised learning, a core subfield of machine learning that has been extensively used for tasks such as classification and regression [1, 2]. In supervised learning techniques, the model we create is trained on a dataset known as the training set, where both the input features and the corresponding labels of all examples are known. These labels serve as the ground truth, allowing the model to "learn" the patterns possesed by the data by minimizing prediction errors. The learning process typically involves optimizing the model's parameters based on a loss function, which measures the discrepancy between the predicted and actual

labels. Once trained, the model can predict new, unseen examples by mapping their input features to a predicted label. These features include the variance, skewness, virtuosity, and entropy of banknotes, as illustrated in Table 1. After the model is trained on this data, it can predict whether a new bill is counterfeit or genuine based on its features, even if it has not been previously encountered. Supervised learning models such as this are essential for counterfeit detection systems because they can generalize from labeled data, making accurate predictions on new instances [7]. This approach can outperform traditional rule-based methods by learning more complex patterns in the data [6].

2.1.3. Binary classification

Each bill belongs to precisely one of two categories: it is either authentic or counterfeit, making this a binary classification problem. In binary classification, the label takes one of two values, typically "yes" or "no," to indicate whether the bill is real. Supervised learning models are often employed for these types of problems, where the model is trained to predict the binary outcome based on input features [12]. In such cases, one category is identified with the value 1 and the other with the value 0. Consequently, we refer to the categories as "category 1" and "category 0," respectively. For this problem, we assign the label 1 if the bill is real and 0 if it is fake. After replacing "Yes" with 1 and "No" with 0, the labels in Table 1 become numerical values. We now obtain a fully numerical tabular dataset.

A binary classification model is a function that predicts the label of an example by processing its features. Specifically, the model takes the input features (in our case, variance, skewness, virtuosity, and entropy) and produces an output between 0 and 1, representing the probability of the bill being authentic. The input features are typically denoted by x_1, x_2, x_3, and x_4, corresponding to variance, skewness, virtuosity, and entropy, respectively.

Models are usually denoted as $\hat{y}$, which is a function of the input features. In our scenario, $\hat{y}$ is expressed as $\hat{y} = \hat{y}(x_1, x_2, x_3, x_4)$. Regardless of the input feature values, the output $\hat{y}$ will always be between 0 and 1. To make predictions using the model, we follow a simple rule based on the output $\hat{y}$ [1]. If $\hat{y}$ is less than 0.5, the model predicts that the bill belongs to category 0, indicating that the bill is fake. Conversely, if $\hat{y}$ is greater than 0.5, the model predicts that the bill belongs to category 1, meaning that the bill is real. This threshold-based decision rule allows the model to classify bills as genuine or counterfeit based on the provided features.

2.1.4. Binary cross-entropy error

We introduce the concept of error to better understand how predictions are evaluated. Let y represent the actual label of an example (either 0 or 1), and $\hat{y}$ be the output of the model when the features of the example are inputted. For instance, if $y = 0$ and $\hat{y} < 0.5$, the prediction is considered correct. However, a prediction of $\hat{y} = 0.2$ seems better than a prediction of $\hat{y} = 0.4$, as the former is closer to 0. This difference in quality between predictions is quantified by the binary cross-entropy error (BCE), which is defined as follows [13]:

$$\mathrm{BCE}(y, \hat{y}) = \begin{cases} -\log(1 - \hat{y}) & \text{if } y = 0 \\ -\log(\hat{y}) & \text{if } y = 1 \end{cases} \tag{1}$$

Figure 1 shows a plot $\mathrm{BCE}(y, \hat{y})$ as a function of $\hat{y}$, with the left graph corresponding to $y = 1$ and the right graph corresponding to $y = 0$. Assume first that $y = 1$. As illustrated in the left graph of Fig. 1, the closer $\hat{y}$ is to $y = 1$, the smaller the binary cross-entropy error $\mathrm{BCE}(y = 1, \hat{y})$. As $\hat{y}$ approaches 1, the error approaches 0. Assume now that $y = 0$. In the right graph, we see that the closer $\hat{y}$ is to $y = 0$, the smaller the binary cross-entropy error $\mathrm{BCE}(y = 0, \hat{y})$. Similarly, as $\hat{y}$ approaches 0, the error also approaches 0.

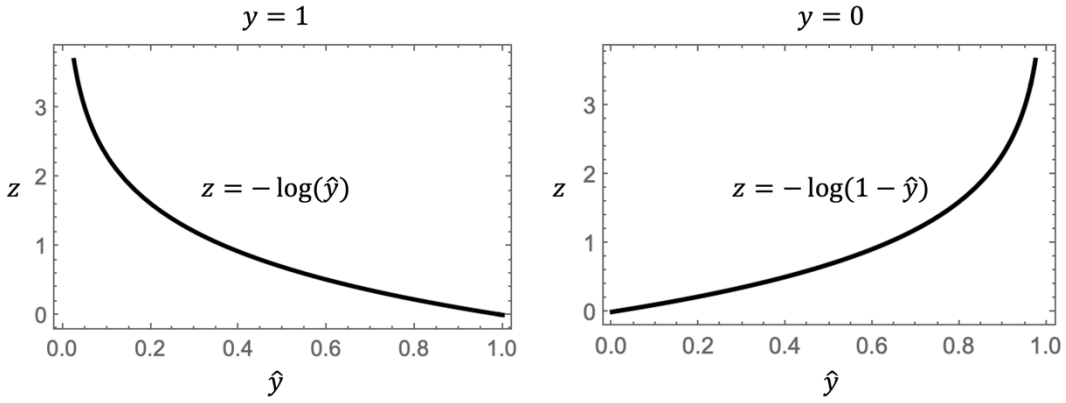

Figure 1. Plot of the curve $z = \mathrm{BCE}(y, \hat{y})$. The left panel shows the case $y = 1$ and the right panel shows the case $y = 0$

The binary cross-entropy error possesses several important properties that make it suitable for evaluating predictions. First, the error is always non-negative, meaning $\mathrm{BCE}(y, \hat{y}) \geq 0$, which ensures that the error is never less than zero, regardless of the prediction. Additionally, the closer the predicted value $\hat{y}$ is to the actual label y, the smaller the error will be. This means the function correctly penalizes predictions that deviate further from the exact value. Finally, as the predicted value $\hat{y}$ approaches the actual label y, the error approaches zero, indicating that the prediction becomes more accurate as it aligns with the actual label. These properties make binary cross-entropy a robust measure for evaluating the accuracy of predictions in binary classification tasks. Revisiting the example of $\hat{y} = 0.2$ v.s. $\hat{y} = 0.4$ when $y = 0$, we have

$$\mathrm{BCE}(y = 0, \hat{y} = 0.2) = -\log 0.8 \approx 0.22$$
$$< \mathrm{BCE}(y = 0, \hat{y} = 0.4) = -\log 0.6 \approx 0.51$$

Thus, we can conclude that $\hat{y} = 0.2$ is a better prediction than $\hat{y} = 0.4$, even though both predictions fall within the correct category of 0. The binary cross-entropy for a model on a dataset, rather than a single example, is the average of the binary cross-entropy errors for all examples in the dataset.

2.1.5 Validation set

The accuracy of a model on a given set of examples is calculated by dividing the number of correctly predicted labels by the total number of examples in the set. In other words, accuracy represents the percentage of correct predictions the model makes. It is important to note that we can only compute the accuracy if the actual labels of the examples are known. For instance, consider the example in Table 2. The table shows the labels y, the model's predictions $\hat{y}$, and whether each prediction was correct for four examples. The model classified three correctly and one incorrectly from these four examples. Therefore, the model's accuracy on this set is 3/4, or approximately 75%.

Table 2. Label and model predictions of three examples

y (label)	$\hat{y}$ (model prediction)	Predicted category	Correct
1	0.45	0	No
0	0.2	0	Yes
1	0.97	1	Yes
0	0.4	0	Yes

After introducing the concept of accuracy, we now have two critical metrics for evaluating a model's performance on a set of examples: accuracy and binary cross-entropy error. Higher accuracy and lower binary cross-entropy error indicate a better model. The ultimate goal is to develop a model that can accurately predict examples whose labels are unknown. These examples will inevitably come from outside the training set since the labels of the training set examples are already known.

From the perspective of machine learning techniques, naturally, an optimal model should minimize the binary cross-entropy error on the training set. As we will explore in Section 3, in bot the case of logistic regression and neural networks, we optimize model parameters to reduce error. It is common for models to perform better on their training set than on unseen examples. This means the model's accuracy will typically be higher, and the binary cross-entropy error lower for the training set than for other sets of examples.

For this reason, accuracy and binary cross-entropy error on the training set alone are not reliable indicators of how well the model will generalize to new, unseen examples. A more reliable strategy involves splitting the dataset into two sets: the training set and the validation set. The training set typically consists of about 80% of the examples randomly selected from the full dataset, while the validation set contains the remaining examples [1, 2]. The model is trained exclusively on the training set, and its performance is evaluated by calculating both accuracy and binary cross-entropy error on the validation set.

2.1.6. The sigmoid function

After evaluating model performance using accuracy and binary cross-entropy, the next step is understanding how these models are constructed. In many binary classification models, including logistic regression and neural networks, a critical component is the activation function [1, 2]. One of the most widely used activation functions is the sigmoid function, which plays a key role in transforming the output of a model into a probability value between 0 and 1. This transformation is essential for binary classification tasks, as it allows the model to predict the likelihood of an example belonging to a certain category. We now introduce the sigmoid function, which is typically denoted by the Greek letter σ. The sigmoid function is defined by the following formula:

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

Figure 2 shows a plot of the graph of this function.

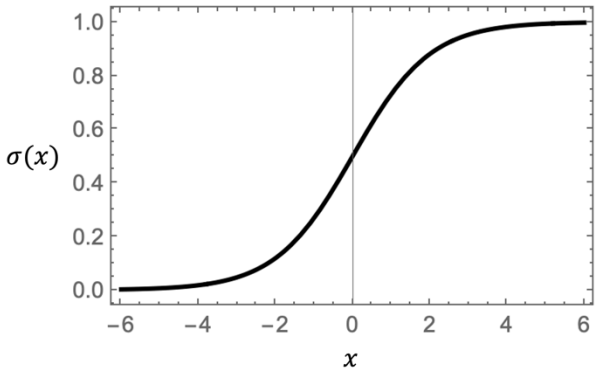

Figure 2. Plot of the sigmoid function $\sigma(x)$

The sigmoid function has several important properties that are essential in machine learning applications [14], and these can be observed from its graph. The function always produces outputs in the range $0 < \sigma(x) < 1$ for all real-valued x. Specifically, when $x = 0$, the function equals 0.5. Additionally, the sigmoid function is monotonic, meaning it consistently increases as x increases. As x approaches positive infinity, the function approaches 1, and as x decreases toward negative infinity, the function approaches 0. The primary role of the sigmoid function in models like logistic regression is to transform the output from the weighted sum into a value between 0 and 1, which could be interpreted as a probability for binary classification.

2.2. Machine Learning Model

2.2.1. Logistic regression

Logistic regression is a commonly used machine learning technique for developing models in binary classification tasks. While it is not the only method available for such problems, it stands out

for its simplicity and effectiveness. In this section, we provide an overview of logistic regression and its application to binary classification.

In logistic regression, the functional form of the model is

$$\hat{y}(x_1, x_2, x_3, x_4) = \sigma(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b) \quad (3)$$

Where as always x_1, x_2, x_3, x_4 are the features. While we assume four features, as in our examples, Equation (3) obviously extends to any number of features. The numbers w_1, w_2, w_3, w_4 are referred to as weights, and b is called the bias. Both weights and bias are known as parameters in logistic regression. Different values of these parameters result in different models, and it is essential to select the optimal parameters to minimize prediction error. This is where the training set and the concept of binary cross-entropy error come into play. In logistic regression, the parameters are chosen to minimize the binary cross-entropy error on the training set, ensuring that the model can make accurate predictions on unseen data [15]. Although the mathematical process for determining these parameters is beyond the scope of this paper, standard machine learning software can efficiently calculate the optimal weights and biases for us.

2.2.2. Neural networks

While logistic regression performed well in this study, it may not always yield highly accurate models for other datasets. In cases where logistic regression falls short, other machine learning techniques can be employed to potentially achieve better results. One such technique is neural networks, which have proven to be both flexible and powerful in handling more complex datasets.

Similar to logistic regression, neural networks model the output $\hat{y} = \hat{y}(x_1, x_2, x_3, x_4)$ as a function of the input features. However, unlike logistic regression, the relationship between the features and the output in neural networks is not straightforward to express with a single formula. Instead, neural networks use a layered architecture to describe this relationship, which is why they are called "networks."

The architecture of a neural network consists of several key components. A network is composed of a sequence of layers, where each layer contains nodes. These nodes are interconnected, with each node in one layer being connected to every node in the next layer through edges. The first layer in the network is referred to as the input layer, and it is responsible for receiving the initial data. The last layer is known as the output layer, which provides the final prediction or classification. Layers that are situated between the input and output layers are called hidden layers, and they are responsible for processing the data and identifying patterns that contribute to the network's overall performance. Figure 3 illustrates the architecture of a neural network [1, 2]. The input layer is often referred to as layer 0, followed by successive layers numbered sequentially. In other words, Layer 1 is the first layer immediately to the right of the input layer, and layer 2 follows layer 1, continuing this pattern. In the example shown in Fig. 3, the network contains a single hidden layer [16], meaning Layer 0 is the input layer, Layer 1 is the hidden layer, and Layer 2 is the output layer.

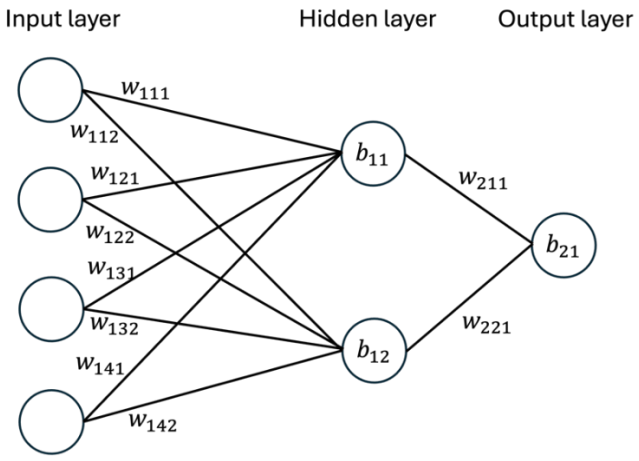

Figure 3. The architecture of a neural layer with one hidden layer

In Fig. 3, the nodes are represented by circles. The top node of each layer is referred to as node 1, with subsequent nodes in that layer being numbered sequentially below it (node 2, node 3, and so on). In the example shown, the input layer consists of four nodes, the hidden layer contains 2 nodes, and the output layer has a single node. The number of nodes in the input layer corresponds to the number of features in the dataset, so for this study, the input layer must have four nodes. For binary classification tasks like this one, the output layer should have only one node.

The number of hidden layers and the number of nodes in each hidden layer are flexible choices. In fact, it is possible to have no hidden layer at all. These values, known as hyperparameters, can be adjusted according to the problem at hand [16]. We will later discuss strategies for selecting these hyperparameters wisely. The edges connecting the nodes (represented as segments) are important components of the neural network structure.

Now, let's explain the concept of weights and biases, which are also referred to as parameters in a neural network. Each node (except those in the input layer) has an associated bias, denoted by b_{ij} for node j in layer i. Additionally, each edge has an associated weight, denoted by w_{ijk}, which represents the weight of the edge connecting node j in layer $i-1$ with node k in layer i.

To compute the network's output, we assign a value (data) to each node, which is calculated differently from the node's bias. This data is computed as follows: the data of node jjj in layer iii is represented as a_{ij}, and the data of node iii in the input layer corresponds to the i^{th} feature of the example, denoted by $a_{0j} = x_j$. The data of a node in any layer depends on the data from the previous layer, the weights of the connecting edges, and the bias of the node. If layer i is a hidden layer, this relationship is represented by the formula:

$$a_{ij} = \text{ReLU}(b_{ij} + \sum_k w_{ikj} a_{(i-1)k}) \tag{4}$$

Where the summation is over the nodes in layer $i-1$. ReLU( ) denotes the Rectified Linear Unit (ReLU) activation function, which plays a key role in these models. The ReLU function is defined by the following formula:

$$ReLU(x) = \begin{cases} x & if \quad x > 0 \\ 0 & if \quad x \leq 0 \end{cases} \tag{5}$$

If the layer i is the output layer hidden layer, Eq. (4) changes to

$$_{i1} = \sigma(b_{i1} + \sum_k w_{ik1} a_{(i-1)k}) \tag{6}$$

In this case, $j = 1$ since there is only one node in the output layer. The prediction, $\hat{y}$, is the data of the node in the output layer. As with logistic regression, the software finds the parameters that minimize the binary cross-entropy error on the training set.

3. Results

3.1. Results of Logistic Regression

We used TensorFlow, a widely adopted Python library, to perform our calculations and create machine learning models [17]. The dataset consisted of 1372 examples, 80% of which were randomly selected to form the training set, while the remaining 20% constituted the validation set.

The logistic regression model parameters calculated by the software were as follows: $w_1 = -2.3$, $w_2 = -1.9$, $w_3 = -1.4$, $w_4 = -0.032$, $b = 2.8$. Table 4 shows the feature labels and predictions of 10 examples of the validation set.

Table 3. Features labels and predictions of 10 examples by linear regression

Variance	Skewness	Virtuosity	Entropy	Authentic	Predictions	Predicted category
0.64295	7.10180	0.34930	-0.41337	0	0.000506	0
2.30660	3.53640	0.57551	-0.41938	0	0.000542	0
-3.57410	3.94400	-0.07912	-2.12030	1	0.998386	1
3.95290	-2.35480	2.37920	0.48274	0	0.000952	0
0.74307	11.17700	-1.38240	-4.07280	0	0.000047	0
2.69460	6.79760	-0.40301	0.44912	0	0.000020	0
3.47690	-0.15314	2.53000	2.44950	0	0.000157	0
4.33650	-3.58400	3.68840	0.74912	0	0.000246	0
-2.28600	-5.44840	5.80390	0.88231	1	0.997142	1
-1.80760	-8.81130	8.70860	-0.21682	1	0.989105	1

As Table 3 illustrates, the predictions are excellent. All the ten examples are predicted correctly. In fact, the accuracy of the entire validation set of this model is 0.992, or 99.2%.

3.2. Results of the Neural Network

While the results achieved using logistic regression were impressive by any standard, in this section, we present the results obtained from a neural network with a simpler architecture: one hidden layer containing two nodes, as illustrated in Fig. 3. The architecture mirrors that of logistic regression in terms of functionality, but the additional layer adds flexibility.

The values of the parameters determined by the software were as follows:

$$w_{111} = -0.86, \quad w_{112} = 1.7,$$
$$w_{121} = -0.40, \quad w_{122} = 0.86,$$
$$w_{131} = -1.5, \quad w_{132} = 0.93,$$
$$w_{141} = 0.29, \quad w_{142} = -0.31,$$
$$b_{11} = 0.29, \quad b_{12} = 0.95,$$

And

$$w_{211} = 1.8,\ w_{221} = -2.3,\ b_{21} = 3.8.$$

Table 4 shows the feature labels and predictions of 10 examples of the validation set. These are the same examples as in the section showing the results using the logistic regression section

Table 4. Features labels and predictions of 10 examples by neural networks

Variance	Skewness	Virtuosity	Entropy	Authentic	Predictions	Predicted category
0.64295	7.10180	0.34930	-0.41337	0	9.628536e-10	0
2.30660	3.53640	0.57551	-0.41938	0	2.629130e-09	0
-3.57410	3.94400	-0.07912	-2.12030	1	9.999194e-01	1
3.95290	-2.35480	2.37920	0.48274	0	1.889810e-08	0
0.74307	11.17700	-1.38240	-4.07280	0	5.774807e-13	0
2.69460	6.79760	-0.40301	0.44912	0	2.596019e-12	0
3.47690	-0.15314	2.53000	2.44950	0	3.560831e-09	0
4.33650	-3.58400	3.68840	0.74912	0	2.366774e-09	0
-2.28600	-5.44840	5.80390	0.88231	1	9.856984e-01	1
-1.80760	-8.81130	8.70860	-0.21682	1	9.856984e-01	1

In Table 4, "e-01" means 10^{-1}. Similarly, "e-09" means 10^{-9}, and so on. As Table 4 illustrates, the predictions are excellent, even better than when using logistic regression as the predected values are much closer the actual value. In fact, the accuracy on the entire validation set of this model is 100%. All examples were correctly classified.

4. Summary

In conclusion, this study has demonstrated the effectiveness of machine learning, particularly supervised learning models like logistic regression and neural networks, in detecting counterfeit currency. Our models achieved high accuracy in distinguishing between real and fake bills by utilizing key features such as variance, skewness, virtuosity, and entropy extracted from banknote images. The study highlighted the superiority of neural networks in capturing complex patterns and achieving higher accuracy than simpler models like logistic regression. Additionally, using binary cross-entropy error as a performance metric emphasized the precision required for making reliable predictions.

Our work underscores the importance of incorporating machine learning into fraud detection systems. The scalability and efficiency of these models offer a significant improvement over traditional methods of detecting counterfeit currency, which often rely on manual inspection or expensive hardware. By automating this process, financial institutions and governments can reduce the costs and risks associated with counterfeit currency, thereby reinforcing public trust in economic systems. Beyond counterfeit detection, the methods presented here have broader applications, including credit card fraud detection, anomaly detection in financial transactions, and identity theft protection. These technologies are crucial in an increasingly digital financial landscape, where rapid, reliable fraud detection is essential for maintaining security.

In summary, this research not only provides a solution to the immediate issue of counterfeit detection but also highlights the broader potential of machine learning in securing financial systems. As these models evolve and become more sophisticated, they will play a critical role in ensuring the integrity of global financial systems, reducing the risk of fraud, and promoting economic stability.

Acknowledgements

Jiayi Li gratefully acknowledges the fruitful discussions with and advice from Dr. Guillermo Goldsztein.

References

[1] Andriy Burkov. The Hundred-Page Machine Learning Book. Andriy Burkov, 2019.

[2] Charu C. Aggarwal. Neural Networks and Deep Learning: A Textbook. Springer, 2018.

[3] Long Yang, Chuan Liu, Jinyang Li. A survey on counterfeit currency detection using machine learning techniques. IEEE Access, 2021, 9: 76471-76490.

[4] U.S. Department of Treasury, Federal Reserve Research Reports, 2012.

[5] S.-H. Chae, J. K. Kim, and S. B. Pan. A Study on the Korean Banknote Recognition Using RGB and UV Information, in Communication and Networking, Berlin, Heidelberg: Springer, 2009.

[6] Seung Baek, Eun Choi, Yeon Baek, Chan Lee. Detection of counterfeit banknotes using multispectral images. Digital Signal Processing, 2018, 78: 294–304.

[7] F. M. Hasanuzzaman, X. Yang, Y. Tian. Robust and effective component-based banknote recognition for the blind. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42 (6): 1021-1030.

[8] T. D. Pham, Y. W. Lee, C. Park, K. R. Park. Deep learning-based detection of fake multinational banknotes in a cross-dataset environment utilizing smartphone cameras for assisting visually impaired individuals. Mathematics, 2022, 10: 1616.

[9] Volker Lohweg. Banknote authentication. UCI Machine Learning Repository, 2013. https://doi.org/10.24432/C55P57.

[10] Euisun Choi, Jongseok Lee, Joonhyun Yoon. Feature extraction for banknote classification using wavelet transform. 18th International Conference on Pattern Recognition (ICPR'06), 2006, 2: 934-937.

[11] Dal Pozzolo Andrea, Boracchi Giacomo, Caelen Olivier, Alippi Cesare, Bontempi Gianluca. Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29 (8): 3784-3797.

[12] Christopher M. Bishop. Pattern recognition and machine learning. Springer, 2006.

[13] Usha Ruby, Vamsidhar Yendapalli. Binary cross entropy with deep learning technique for image classification. International Journal of Advanced Trends in Computer Science and Engineering, 2020, 9 (10).

[14] Jun Han, Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. International Workshop on Artificial Neural Networks, Berlin, Heidelberg: Springer, 1995.

[15] Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2009.

[16] DeepAI. Hidden layer. DeepAI, 17 May 2019. https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning.

[17] Andreas C. Müller, Sarah Guido. Introduction to machine learning with Python: a guide for data scientists. O'Reilly Media, Inc., 2016.