A Neural Network Approach for Birds of a Feather Solvability Prediction

Benjamin Sang, Sejong Yoon
The College of New Jersey
Ewing Township, New Jersey 08618

Abstract

Birds of a Feather is a single player, perfect information card game. The game can have multiple board sizes with larger boards introducing larger search spaces that grow exponentially. In this paper, we investigate the solvability of the game, aiming at building a machine learning method to automatically classify whether a given board state has a solution path or not. We propose a method based on image-based features of the board state and deep neural network. Experimental results show that the proposed method can make reasonable predictions of the solvability of a game at an arbitrary stage of the game.

Introduction

Birds of a Feather (BoF) is a single-player, perfect information card game that was introduced for a student-faculty research challenge at EAAI 2019 (Neller 2018). The goal of the game is to find a deterministic solution sequence given an initial board state following a list of rules. The initial state consist of cards dealt from a shuffled, basic 52 card deck, which consists of four suits (Club, Diamond, Heart, and Spade) and thirteen ranks per suit (A to K). A board state is partitioned into square grids, which can vary in sizes from 1×1 to 7×7. The player must move one card at a time, on top of the other card, following the rules, until there are no more movable cards left on the board. There are two rules that must be followed in the game: a card can be placed on top of the other (a) if cards share a suit, or (b) if the two cards have an equal or adjacent ranks. The winning condition is satisfied when only one stack of cards remains on the board.

The research challenge is open-ended and many topics of research were suggested by the organizer. In this paper, we chose to investigate the solvability prediction problem, that aims to predict whether a game state has a solution path or not. There are two reasons for this choice. First, it is fairly straightforward to formulate this problem as a classification problem which is intuitive to understand for a beginning student researcher. Second, it was possible to collect large scale data for the problem, both by obtaining from the organizers and by generating our own dataset.

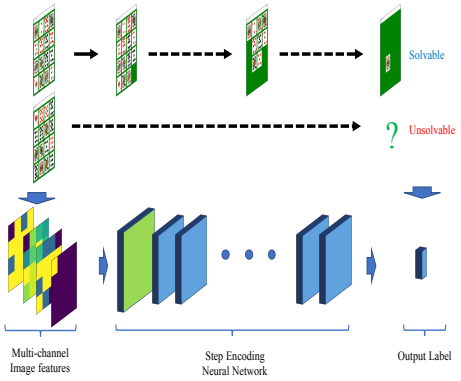


Figure 1: Overview of the proposed classification framework

Particularly, we aim to answer the following three research questions in this work: (a) Is it possible to train a machine learning model to find a pattern that distinguishes both trivial and non-trivial unsolvable initial board states from solvable ones? (b) Is it possible to apply the model trained by using initial board states to predict the solvability of the interim (i.e. game-in-progress) board states? (c) What are effective representations of the board state (or features extracted from the states) that may improve the prediction performance?

To this end, we designed a neural network-based classification framework that accepts board states as inputs to the framework, and solvability as the corresponding binary output of the framework. The main contributions of this paper are two-fold: (a) We propose a BoF solvability classification framework that utilizes image-based features to represent the board state, (b) We show how the proposed framework can utilize the suggested image-based features, either individually or in combination, to effectively predict BoF solvability. Additionally, we provide a dataset with 1 million unique initial board states with solvability information and (if solvable) solution sequences we used in our study. Fig. 1 depicts the overview of the proposed framework.

The rest of the paper is organized as following: We first describe our classification framework, introduce the dataset we collected, followed by image-based features we propose. Then, we present our experimental results with analysis. We

conclude our paper with a list of future directions for follow-up research topics.

Method

In this section, we introduce the neural network-based BoF solvability classification framework we used. First, we formulate the BoF solvability problem to tackle, and then explain the deep neural network architecture we used, followed by image-based features we designed.

Problem Formulation

Our goal is to find a nonlinear mapping $f : \mathbf{X} \rightarrow \mathbf{y}$ from an input board state $\mathbf{X} \in \mathbb{R}^{M \times M \times C}$ represented in a feature tensor, to a 2-dimensional output column vector $\mathbf{y} \in \mathbb{R}^2$. Here, M denotes the size of the board's grid and C is the number of feature channels. To find f, we employ a data-driven approach, and prepare a dataset of known pairs of input board states and solvability information, $\mathcal{D} = \{(\mathbf{X}_n, \mathbf{y}_n) | n = 1..N\}$, where n denotes the index of pairs, and N is the total number of pairs in the dataset. Note that in our dataset, labels $\mathbf{y}_n$ are either $[1; 0]$ or $[0; 1]$ in MATLAB/Octave notation (a two dimensional column vector), denoting solvable and unsolvable board states, respectively. Then, our objective is to find f that minimizes

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \| f(\mathbf{X}_n) - \mathbf{y}_n \|^2, \tag{1}$$

also known as the Mean Squared Error (MSE) loss function.

Neural Network-based Classification Framework

If f is in the form of a function of parameters $\mathbf{w}$, finding f is essentially equivalent to finding $\mathbf{w}$. Finding an analytic solution to the problem above may be possible if the derivative of f with respect to $\mathbf{w}$ is known. We can utilize the gradient descent algorithm to reach the optimal solution for $\mathbf{w}$. To enhance the expression power of f, one can design f as a composite function of multiple simpler functions and then apply the chain rule to derive the derivative of the overall function.

Artificial neural networks (ANN), with the stochastic gradient descent (Robbins and Monro 1951) and the back-propagation algorithms, are one of the standard tools to solve this type of problems. In ANN, each layer can be considered as a function to be combined, and the stack of layers essentially represent the overall nonlinear function we want to find. With the advances in hardware technologies, one can design and train very *deep* and *wide* ANN. It is also a well-known fact that ANN are particularly useful in learning unknown representations of data (Rumelhart, Hinton, and Williams 1988). Since we are unsure which feature representation of the board state is effective for the solvability prediction problem, it would be reasonable to consider adopting ANN for our research. Moreover, our preliminary experiments using other machine learning methods, e.g., Support Vector Machine (Cortes and Vapnik 1995) was not as successful as we expected. Further investigation may be needed to justify the shallow discriminating models for the solvability prediction for the BoF data we collected.

Figure 2: Proposed Step Encoding Network. Each FC layer shows the numbers of input and output node.

Thus, we propose to employ deep ANN for our solvability classification framework. For the network architecture, we employ two types of layers, a convolutional neural network (CNN) (LeCun et al. 1998) and 15 standard fully connected (FC) layers. The whole network consist of two parts: representation encoding and step encoding. The former learns spatial structural patterns from the training board states, represented as image-based features we will describe below. The latter learns the step-by-step game procedure from the initial (or current) board states to the goal state, which hopefully will regress to the solvability label vector $\mathbf{y}$. Fig. 2 describes the overall ANN architecture we used. We provide empirical justification on the choices of the network structure hyper-parameters in the experiment results section.

Features

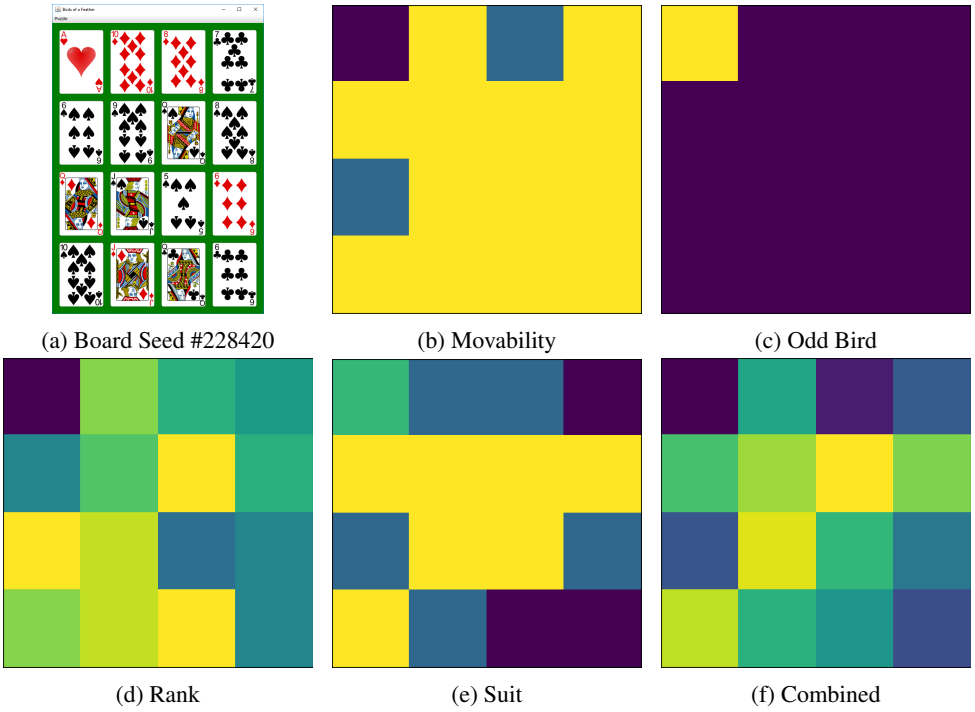
For our application, desirable features should describe well the implicit characteristics of the game states, and preferably have strong discriminating power for the solvability classification task. Naturally, one must start with investigating the rules of the game to design such features. Players (or computer systems) must be able to recognize the current board state as a square grid and understand the rules of the game which are:

- Cards can only move left, right, up, and down.
- Cards can only move on top of another if:
  - The cards share a suit.
  - The cards share the same or are an adjacent rank.

From the set of restrictions, we defined three primary features for our model.

- `Rank`: The rank of a card on the board (0, 1, 2, ... , 13)
- `Suit`: The suit of a card on the board (0, 1, 2, 3, 4)
- `Movability`: Whether a card is unmovable, movable horizontally (left-right), vertically (up-down), or both (0, 1, 2, 3, 4, respectively)

We reserved 0 to indicate cases when the card does not exist in that grid cell. We considered one more feature, `Odd Bird` (OB), which was identified by the Challenge organizer (Neller 2018). This is the case when a card cannot



(a) Board Seed #228420 (b) Movability (c) Odd Bird

(d) Rank (e) Suit (f) Combined

Figure 3: Example image-based features. Colors indicate different normalized intensities of the feature value at each grid cell. The combined figure (f) was generated by taking the average of all features.

move at all from its current position, regardless of other cards' movements at an arbitrary board state. The OB feature is particularly useful in distinguishing solvable and unsolvable cases from the initial board state when there is no empty position in the grid. For OB feature, we used 1 to indicate the card in that cell is OB, and 0 otherwise. After extracting the features using the encoding above, we normalized each feature to fall in $[0, 1]$ range.

We also transformed these features into images of square grid cells, to preserve the spatial structure information. Utilizing image processing methods to pre-process originally non-image-based information and generate image-based features that encode structural information is a common technique in machine learning applications. One can easily find examples in literature, e.g. (Zhang et al. 2015). The generated image-based features are fed into CNN, so that our ANN can automatically extract implicit structural pattern to distinguish the unsolvable cases from solvable ones. Fig. 3 shows an example of the features we used in this work.

Experiments

In this section, we describe the datasets we collected and the experimental setup we used in this study.

Datasets

We used two datasets in this work. One datset we used is the one publicly distributed by the organizers, and we refer to this dataset as `BOAF`. Each initial board state can be uniquely determined by the corresponding seed value, which

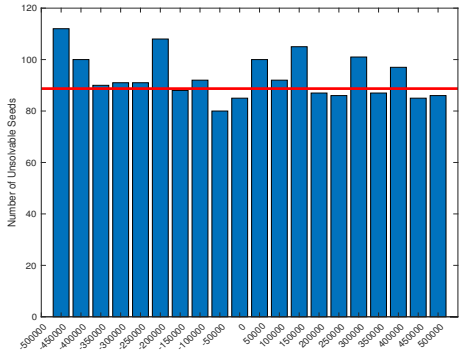


Figure 4: Histogram of unsolvable seed (initial board state) distribution. Red line indicates the mean.

is an integer input to the randomization algorithm (linear congruential generator) included in the code provided by the Challenge organizers. This dataset contains only the seeds between 0 and 10,000 and it has all interim board states throughout each game and corresponding solvability information. From this dataset, we used both the initial and interim states as independent samples. The other dataset is the one we collected in-house utilizing the provided code. We refer to this dataset as `InitSet` since we collected only initial board states as independent samples in the dataset.

To start with `InitSet` data collection, we began with modifying the program "Experiment 1" that returns the solv-

Table 1: Statistics on Datasets. OB denotes the Odd Birds case when a card cannot move at all from its current position, regardless of other cards' movements at an arbitrary board state. In this study, we only used the balanced set of samples from `InitSet`. The collected dataset contains extra solvable cases for the future research. For BOAF dataset, each interim board state counts as an independent sample.

Dataset	Train	Val.	Test
InitSet (unsolvable-w/ OB)	734	442	295
InitSet (unsolvable-w/o OB)	197	118	77
InitSet (solvable)	931	560	372
InitSet (total)	**1,862**	**1,120**	**744**
InitSet (solvable-extra)	498,138	298,880	199,256
BOAF (unsolvable-w/ OB)	63,742	38,103	38,597
BOAF (unsolvable-w/o OB)	300,931	180,701	181,859
BOAF (solvable)	318,056	144,218	127,224
BOAF (total)	**682,728**	**363,022**	**347,680**
All (InitSet + BOAF)	**684,590**	**364,142**	**348,200**

ability of a range of seeds. After confirming that negative seeds yield distinct and different board states from their positive, binary, and hexadecimal counterparts, we chose to take the seeds in the range -500,000 and 499,000, resulting in a data pool of 1 million seeds. After generating the data, we reconfirmed that every initial board state is unique in the pool. We also discovered that out of the 1 million seeds, only about 1,800 seeds are unsolvable. As shown in Fig. 4, the unsolvable seeds are almost uniformly distributed.

Experiment Setup

For the experiments, we first extracted the same set of features from both datsets. Since the unsolvable cases are uniformly distributed, we randomly shuffled the samples and split the samples into three sets: 50% for training, 30% for validation, and 20% for testing. We always used validation set to find the best learning hyper-parameters (e.g. batch size, number of total iterations, initial learning rates, decaying factor). In `InitSet`, the initial seeds within the separate training, validation, and testing sets are also chosen within the pool of the solvable seeds and unsolvable seeds separately, so that each set contained an equal amount solvable and unsolvable seeds. This way we can address the issue of having a severely imbalanced dataset in the `InitSet`. Table 1 summarizes the statistics of the dataset we used.

Implementation

We utilized PyTorch (Paszke et al. 2017) for the neural network architecture implementation. We used each data set's training and validation sets to find the best batch size, learning rate, and the number of epochs to train our network on. We used the batch size of 100, a learning rate of 0.001, and an epoch length of 50,000 for `InitSet` in all our experiments. For the learning rate, we used a learning rate scheduler, which would increase the learning rate by 0.1 every 13,000 epochs. When training and testing on `BOAF`, we increased the batch size to 10,000 but all the other parameters remained the same. For optimization, we used ADAM (Kingma and Ba 2014).

Table 2: Comparison of Different Network Architecture.

Condition	Dataset	Val.	Test
w/o CNN	InitSet	67.3%	68.5%
w/ CNN	InitSet	**90.1**%	**90.6**%
w/o CNN	BOAF	76.1%	76.0%
w/ CNN	BOAF	76.6%	76.7%
w/o CNN	All	76.5%	76.7%
w/ CNN	All	76.4%	76.6%
w/o shrink	InitSet	88.9%	89.8%
w/ shrink	InitSet	**90.1**%	**90.6**%
w/o shrink	BOAF	76.8%	77.0%
w/ shrink	BOAF	76.6%	76.7%
w/o shrink	All	76.4%	76.6%
w/ shrink	All	76.4%	76.6%

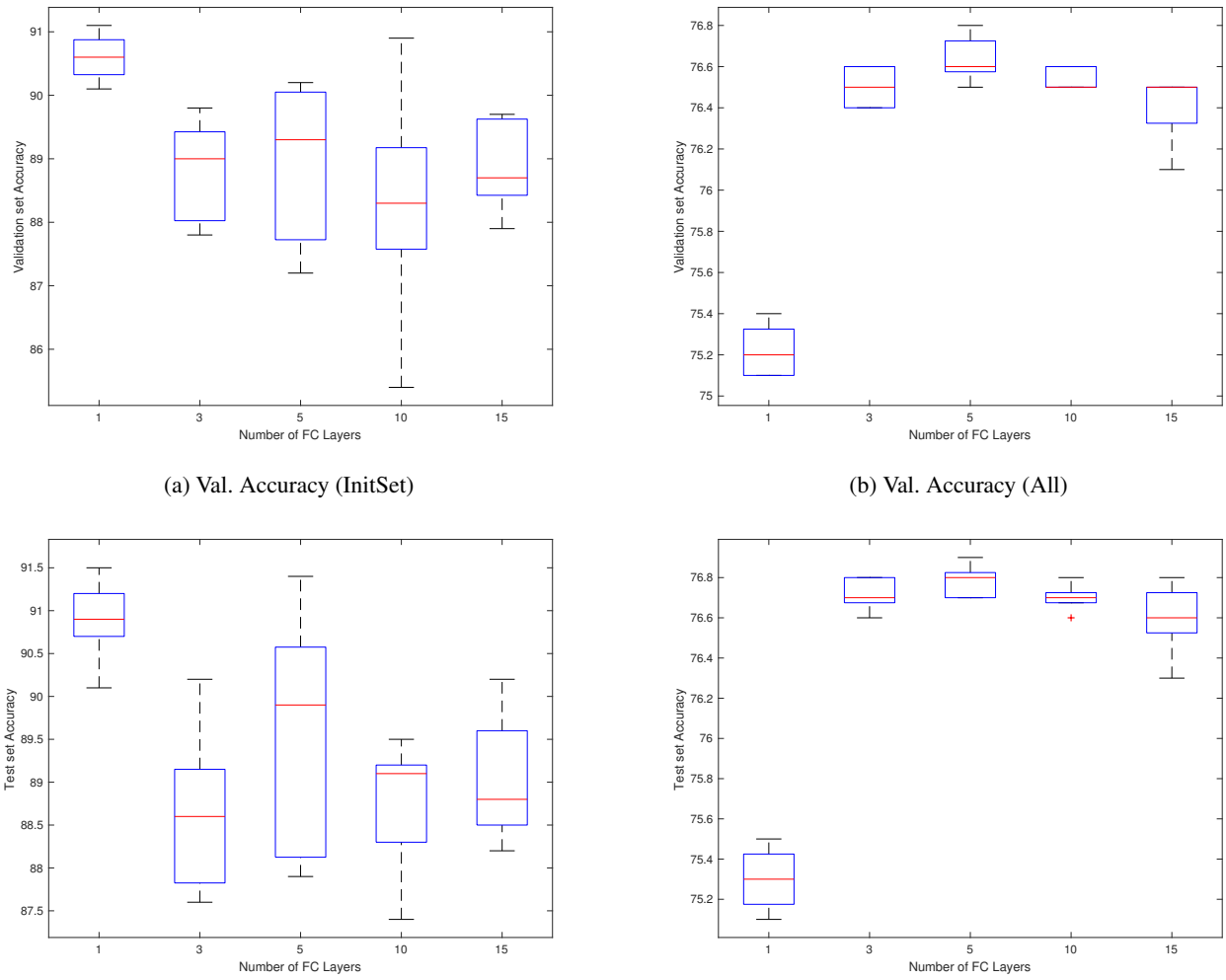
Results and Discussion

In this section, we first describe empirical properties of the proposed classification framework. Then, we answer our three research questions, i.e. (a) training a classification model to distinguish both trivial and non-trivial unsolvable initial board states from solvable ones, (b) generalization ability of the learned model to predict the solvability of the interim board states, and (c) effective feature representations to improve the prediction performance.

Hyper-Parameter Choices

We empirically investigated four framework design questions here: (a) Is CNN necessary? (b) Should the FC layers shrink (i.e. gradually reducing from 32 dimension to 2 dimensions)? and (c) How many FC layers are needed? To answer (a) and (b), we tried to train our model with and without the CNN and shrinking. In non-shrinking case, we set all internal FC layers to have same input and output size (32) except the last node that regress to the output (2 dimensions). The experimental results indicate that both CNN and shrinking layers are necessary to improve the solvability classification performance on `InitSet`, but not significantly in `BOAF` and `All`, where we combined both `InitSet` and `BOAF`. This is because our `InitSet` only consists of initial board states with no missing cells, thus CNN can find spatial dependency better. Table 2 summarizes this result.

Fig. 5 shows the validation and test accuracies on `InitSet` and `All` cases with varying number of FC layers. We started our study with the choice of 15 layers trying to encode the 15 game steps from the initial state to the goal state. The result on `InitSet` shows that 1 layer actually performed best and the other four layer choices have no significant difference. However, this can be due to the small dataset size of `InitSet` since in `All`, we see that 1 layer actually performed significantly worse than the other four configurations. Based on this analysis we concluded that there is no significant difference among the layer choices among 3, 5, 10, and 15. Thus we kept 15 layers to hopefully encode the 15 game steps.



(a) Val. Accuracy (InitSet)

(b) Val. Accuracy (All)

(c) Test Accuracy (InitSet)

(d) Test Accuracy (All)

Figure 5: Comparison with different number of FC layers. Note that `InitSet` is less than one percent of `All` in terms of number of samples. Thus the result for `BOAF` is similar to that of `All`. Similar trends can be observed in Table 2 and Table 3.

Fig. 6 shows the loss function value trend over the training iterations. It shows that our framework converges to reasonably small value as the maximum loss per sample is 1.

Direct Inference from Initial and Arbitrary State

Table 3 summarizes our experimental results using all datasets. We conducted several independent trials with different random shuffling but the variances were negligible in most cases. In both `InitSet` and `BOAF`, the proposed framework was able to obtain good classification performance with the proposed image-based features, well-above both the random decision (50%), and also simple rule-based decision based on existence of OB (89% for `InitSet` and 56% for `BOAF`).

Generalization of Trained Models

We also conducted experiments across datasets, i.e. using one dataset as training set and test on the other. This way, particularly trained using `InitSet` and test on `BOAF`, we can see whether our ANN model trained using only initial board states can generalize its prediction to arbitrary game state. Table 4 summarizes this result. When trained with `BOAF` and tested on `InitSet`, our method showed comparable performance with the case when the model was trained using `InitSet`. This is expected given large amount of training data. When trained with `InitSet` and tested on `BOAF`, the performance was not as good as the other case. This is expected since `InitSet` does not have cases with empty cells. However, the result is at least better than the cases when using only Movability or Rank/Suit information for features. It is also better than rule-based decisions based on the existence of OB. Given that the percentage of unsolv-

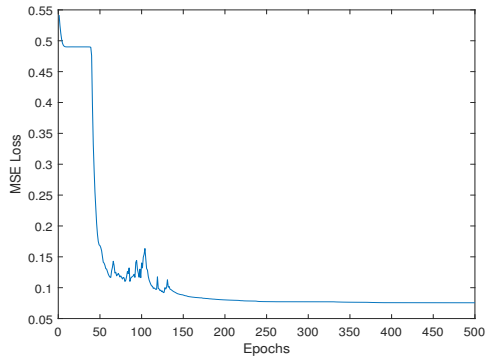


Figure 6: Mean Squared Error loss over the training iterations, using all features, on InitSet dataset.

Table 3: Comparison of Features

Dataset	Feature	Val.	Test
InitSet	Movability	66.2%	61.0%
InitSet	OB	89.7%	89.7%
InitSet	Rank, Suit	50.4%	51.3%
InitSet	Movability, Rank, Suit	68.0%	68.0%
InitSet	Movability, OB	89.6%	**91.0**%
InitSet	All	**90.0**%	90.3%
BOAF	Movability	60.3%	63.4%
BOAF	OB	73.1%	73.5%
BOAF	Rank, Suit	60.3%	63.4%
BOAF	Movability, Rank, Suit	74.8%	75.2%
BOAF	Movability, OB	76.0%	76.3%
BOAF	All	**76.6**%	**76.7**%
All	Movability	72.2%	72.9%
All	OB	60.2%	63.4%
All	Rank, Suit	71.1%	71.4%
All	Movability, Rank, Suit	74.7%	75.8%
All	Movability, OB	75.7%	75.9%
All	All	**76.5**%	**76.7**%

able seeds with odd birds in `BOAF` is roughly 17.5% out of all unsolvable seeds, we can safely say that the framework has learned a bit of non-obvious patterns without OB.

In Table 5, we provide our method's performance on all samples in `InitSet`, including the solvable cases. We trained our model using the same training data (1,862 samples), but we added all remaining solvable-extra cases in Table 1 to the test set. In this case, since we have many more positive (solvable) cases than the negative (unsolvable) ones, we report true positive rate and true negative rate as well as accuracy. As it can be seen in the result, OB has the most important role in this data. It is not surprising that, using the OB only, one can obtain 1.00 TPR and 0.79 TNR from the dataset by predicting all samples without OB as solvable. More important thing to note is that, using the proposed combination of features, one can improve TNR without yielding too much TPR. It should be reminded that we have shown the utility of the proposed method in non-initial board states as shown in Table 3.

Table 4: Comparison of datasets as trainset using the proposed method.

Trainset	Val/Testset	Val.	Test
InitSet	BOAF	64.4%	68.1%
BOAF	InitSet	89.9%	89.9%

Table 5: Results on all samples in `InitSet`. We trained our model using only 1,862 train samples and tested on all remaining samples. True Positive Rate and True Negative Rate are reported to demonstrate the performance in the imbalanced dataset.

Method	Accuracy	TPR	TNR
Movability	82.5%	0.83	0.37
OB	99.9%	1.00	0.79
Rank, Suit	58.7%	0.59	0.41
Movability, Rank, Suit	80.2%	0.80	0.24
Movability, OB	97.6%	0.98	0.80
All	89.1%	0.89	0.85

Feature Selection

Table 3 also summarizes performances of different combinations of the proposed features. As it can be seen, employing all features showed the best performance in almost all cases we considered. Only in `InitSet` test set, it was second best. Moreover, our framework consistently improved performance over OB-only cases, showing that the proposed features can learn non-obvious patterns in unsolvable states.

Conclusion and Future Work

In this work, an artificial neural network-based framework was proposed for the solvability classification problem of Birds of a Feather card game. With the proposed image-based features, the proposed system can effectively learn patterns of non-obvious unsolvable board states both from initial and in-progress game states. The framework also showed that it is possible to generalize patterns learned from one type of dataset (either only initial states or only in-progress states) can be used for the other type of dataset.

We hope the outcome of this work opens several future research directions to the community. An interesting topic would be to find a way to possibly utilize the large number of solvable initial board states in `InitSet`, learn the general patterns of solvable cases, and use them to further improve the solvability classification performance. Another direction would be to investigate our approach's applicability to different board sizes other than 4×4.

Acknowledgments

This work was supported in part by The College of New Jersey under Support Of Scholarly Activity (SOSA) 2017-2019 grant. The authors acknowledge use of the ELSA high performance computing cluster at The College of New Jersey for conducting the research reported in this paper. This cluster is funded by the National Science Foundation under grant number OAC-1828163.

References

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20(3):273–297.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Neller, T. 2018. EAAI-19 Birds of a Feather Undergraduate Research.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Robbins, H., and Monro, S. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 22(3):400–407.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1988. Neurocomputing: Foundations of research. Cambridge, MA, USA: MIT Press. chapter Learning Representations by Back-propagating Errors, 696–699.

Zhang, X.; Zhu, B.; Li, L.; Li, W.; Li, X.; Wang, W.; Lu, P.; and Zhang, W. 2015. Sift-based local spectrogram image descriptor: a novel feature for robust music identification. *EURASIP Journal on Audio, Speech, and Music Processing* 2015(1):6.