Results of the SET project

Leonardo Amabile, Jacopo Omodei

1 Introduction to the game

The game *SET* consists of identifying, within a set of 12 cards, a trio of cards that constitute a winning *SET*. Each card is characterized by four distinct attributes:

- **Color**: red, green, purple
- **Shape**: rhombus, S, ellipse
- **Fill**: empty, shaded, full
- **Number of symbols**: 1, 2, 3

A winning *SET* is a trio of cards that, for each attribute, has either all the same or all different values. Some examples of valid *SETs* are illustrated in Figure 1.

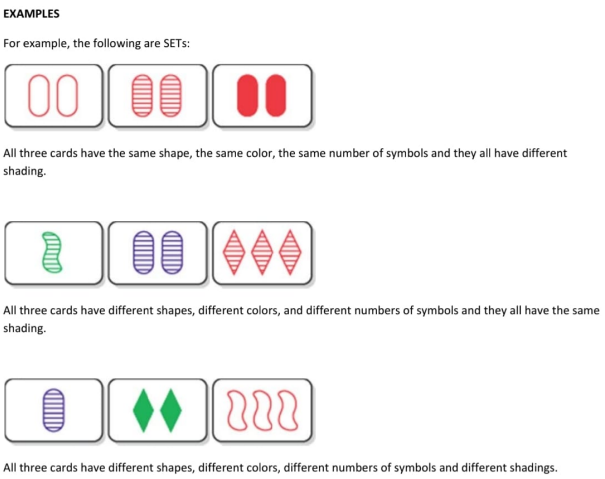

Figure 1: Samples of valid *SET* taken from the official game instructions

.

2 Problem Definition

The goal of the project was to estimate the minimum number of cards needed to guarantee the existence of at least one *SET*, as a function of the number of attributes. In the literature such a value is known up to $n_{att} = 5$, and our aim was to confirm existing results and propose an estimate for higher values of n_{att}.

Initially, we attempted to solve the problem using a neural network capable of estimating the number of *SETs* present in a game hand. However, the results obtained were not satisfactory. We therefore simplified the problem: to determine, given three cards, whether they constitute a *SET* or not. This approach has the advantage of providing an unambiguous answer with a single operation instead of checking the sum over all attributes. In addition, if the results are satisfactory, it could be implemented in the search algorithm to reduce computational time.

3 Data Generation

For the generation of training data, a program was developed in `C++`, chosen for its efficiency in performing loops and searches.

n	# of cards
1	2
2	4
3	9
4	20
5	45
6	between 112 and 114
≥ 7	unknown

Figure 2: Table summary of the maximum number of cards that do not contain *SET*, as the number of attributes varies

We modeled a game hand as a class represented by an array $n_{cards} \times n_{attributes}$, whose elements take values -1, 0 or 1 (corresponding to the three possible options for each attribute). To ensure the uniqueness of the cards, a hash function combined with a `unordered_map` was employed.

The search for *SET* was optimized by exploiting the property of our problem that three cards form a *SET* if and only if the sum of their attribute vectors consists solely of -3, 0 or 3. The search process was accelerated by sorting and binary searching the first column, resulting in a time gain of up to a factor of 3 compared to an exhaustive approach.

4 Neural Network Models

We next trained two types of neural network models for binary classification of the *SET*:

- **DNN** (Deep Neural Network) with densely connected layers.
- **CNN** (Convolutional Neural Network) with 1D convolutions to exploit invariance with respect to card order.

The matrices were enriched with row permutations to augment the dataset in a manner consistent with the semantics of the problem. *Early stopping* and *learning rate reduction* were also adopted to avoid overfitting and optimize training.

5 Results

In the initial phase, we focused on optimizing the time required to detect all *SET* in a table. The optimized algorithm proved significantly more efficient for tables containing more than 50 cards, reducing computation time by up to a factor of 3.

```
Enter the number of cards: 500
Enter the number of attributes: 6
Enter the number of tables: 1
Boolean version (Y/n): n
Generating tables...
[==================================================] 0 %
Tables generated!

Finding SETs...
[==================================================] 0 %
Results saved to Data.txt
Execution time for the optimized process: 18.7706 seconds.
Finding SETs in brute force...
[==================================================] 0 %
Results saved to Data_brute_force.txt
Execution time for the brute-force process: 50.0521 seconds.
```

Figure 3: Screenshot of the terminal showing the time taken to identify *SET* on 500 cards, comparing the optimized model with the brute-force approach

Both neural models showed good generalization capabilities. The loss function used is the *binary cross-entropy* (BCE), where a value of 0.69 indicates random behavior. The results are shown in the following graphs.

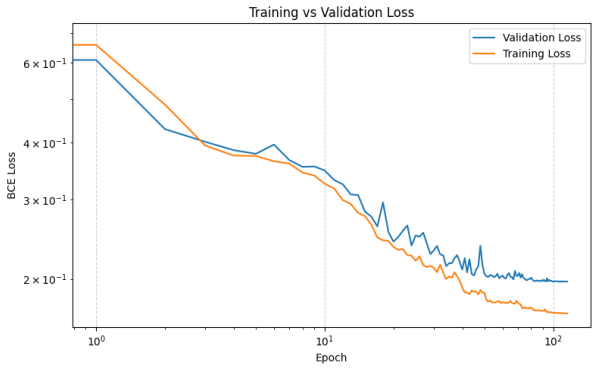

Figure 4: Loss trend for the CNN model as a function of epochs

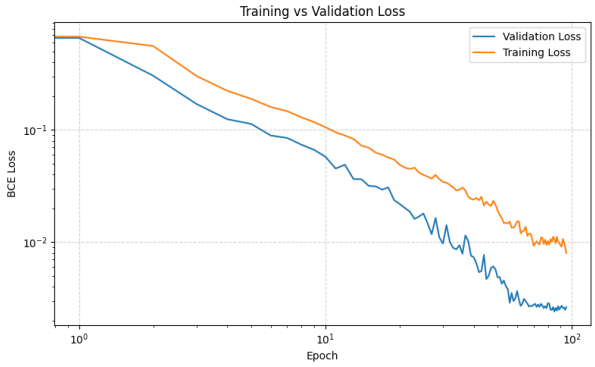

Figure 5: Loss trend for the DNN model as a function of epochs

The DNN model proved to be particularly efficient: with about 61,000 parameters, compared to the CNN's 258,000, it achieved better performance in less training time.

For validation, we applied the models to 10,000 tables generated independently of the training dataset. The DNN obtained 4 misclassifications in 0.54 seconds, while the optimized algorithm in `C++` performed the same task without errors but in about 31.8 seconds. It is important to consider that, even in the case where only three cards are analyzed, the search algorithm in `C++` is not limited to simply checking the conditions of *SET*. In fact, before summing over the attributes, a sorting of the three cards based on the value of the first column is performed. As a result, the computation time is significantly longer than the expected theoretical time of $n_{attributes} \cdot \Delta t$.

6 Conclusions

We can say that the project has achieved significant results. We have developed a fast and efficient method for testing whether a triplet of cards constitutes a *SET*, achieving significantly higher classification speed than the analytical approach. The integration of the neural network (DNN) with the optimized search algorithm allows us to find the *SET* in a hand of cards efficiently, with minimal loss of accuracy (negligible mismatches) compared to the exact solution.

In conclusion, we can say that the identification of *SET* within a hand of cards can be greatly optimized by synergistically combining the two main components of our project: on the one hand, the implementation in `C++` allows minimizing the number of operations required to analyze a hand; on the other hand, the neural network allows significantly speeding up the verification of the validity of a trio of cards, determining in a very short time whether or not it constitutes a *SET*.