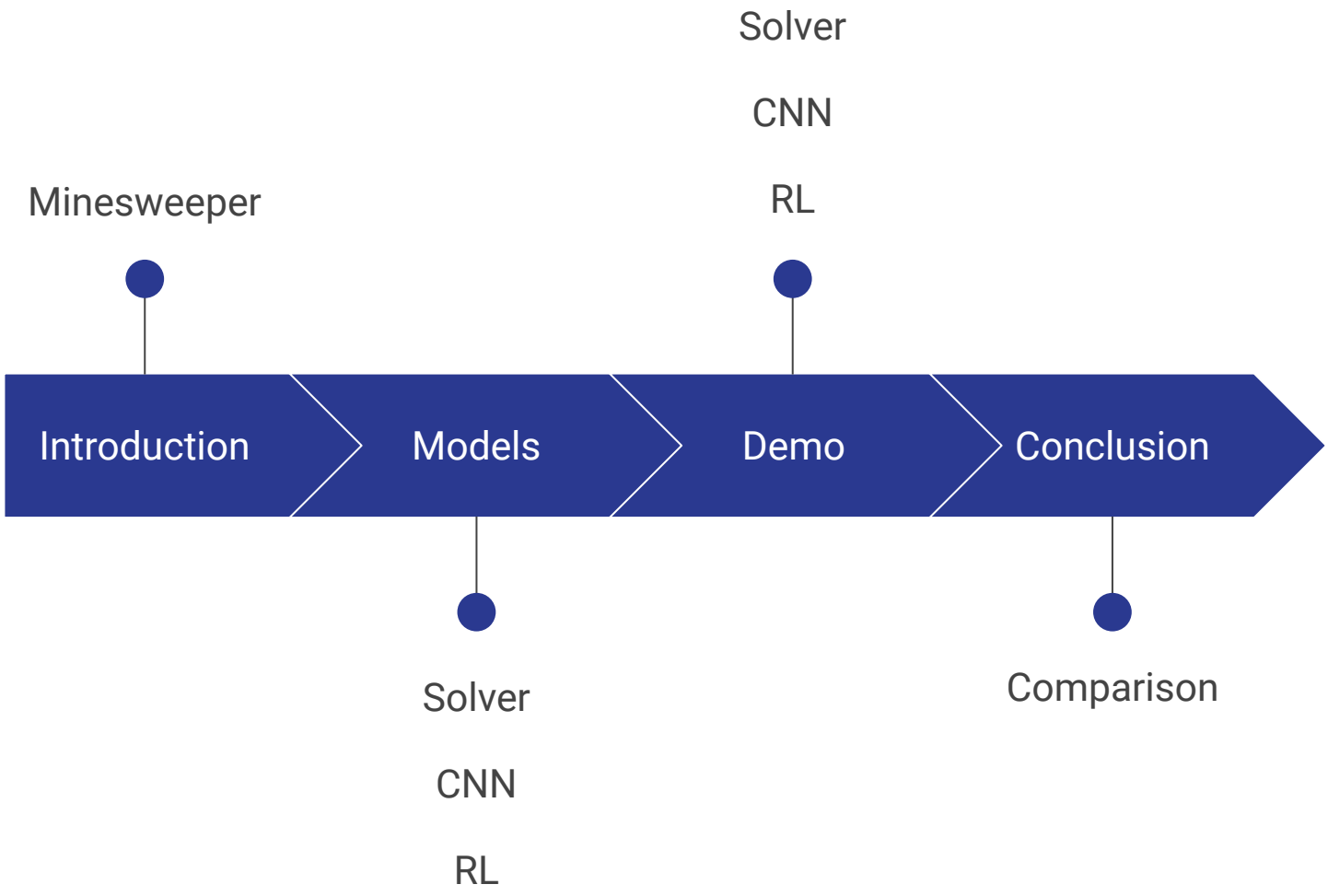
Minesweeper

Techniques of artificial intelligence

Anthony Zhou, Hamza EL Miri & Julien Baudru


Solver
CNN
RL
Minesweeper
Introduction
Models
Demo
Conclusion
Solver
CNN
RL
Comparison

Introduction

Introduction

Single-player game.

Goal : The player must discover as many tiles in the game as possible while trying to never click on a mine.

Clues : The number on a discovered tile indicating how many mines are in its neighborhood

Introduction

Minesweeper is a **NP-Complete*** puzzle so there is **no deterministic** algorithm that can solve it efficiently **in polynomial time**.

*Moti Ben-Ari, "Minesweeper is NP-Complete", Weizmann Institute of Science 2018

Models

Solver

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Solver

Algorithm

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

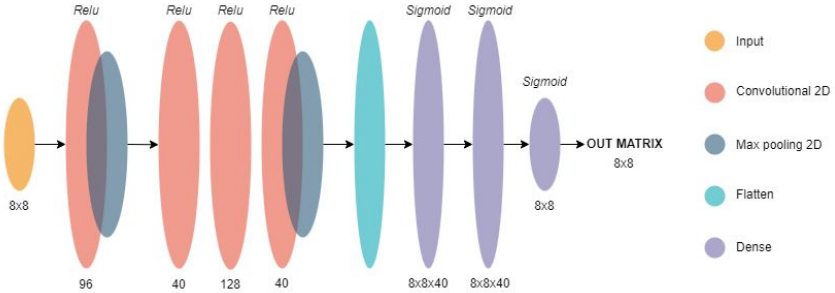
Solver

Testing

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Convolutional neural network

Architecture



We chose **CNN** because they are usually used in image processing.

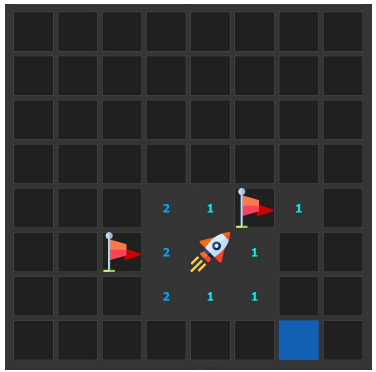
Effective for detecting **features inside matrices**.

Close to the famous **AlexNet** architecture.

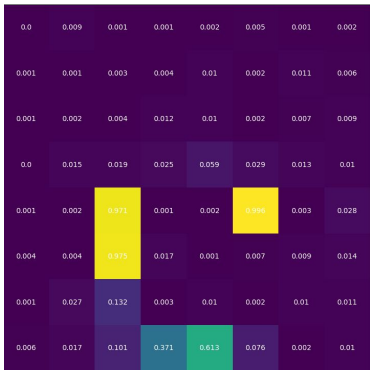
Convolutional neural network

Data

Input



Output



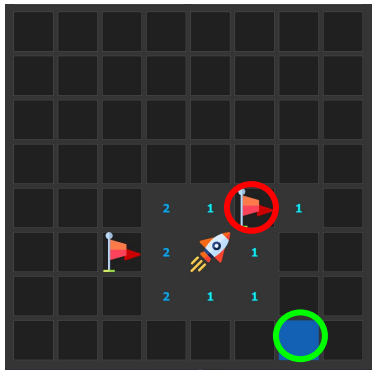
Input : Board game of size 8X8 with the value of the revealed tiles (0 to 8) and -1 for the unknown tiles.

Output : Matrix giving probability of finding a mine at each position (close to 1 if mine, close to 0 otherwise)

Convolutional neural network

Algorithm

Input



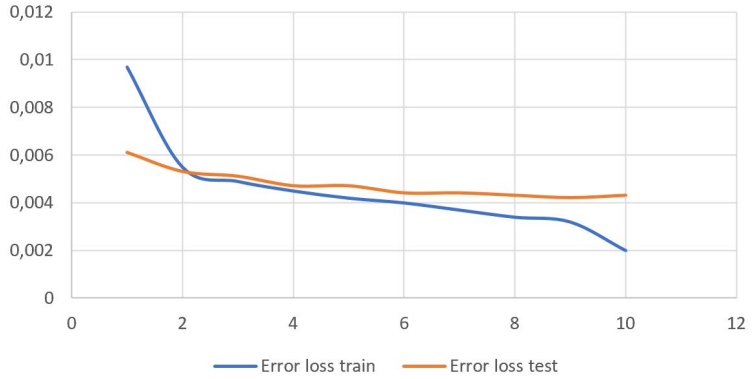
Output

Algorithm :

- **Click** on the tile with the **lowest probability** in the perimeter of the revealed tiles
- Place a **flag** on the tile with the **highest probability** in the perimeter of the revealed tiles

Convolutional neural network

Training



We trained our model on **5,000,000 boards** for **10 epochs**

On the test set the model has reached an **error loss of 0.0424%.**

Convolutional neural network

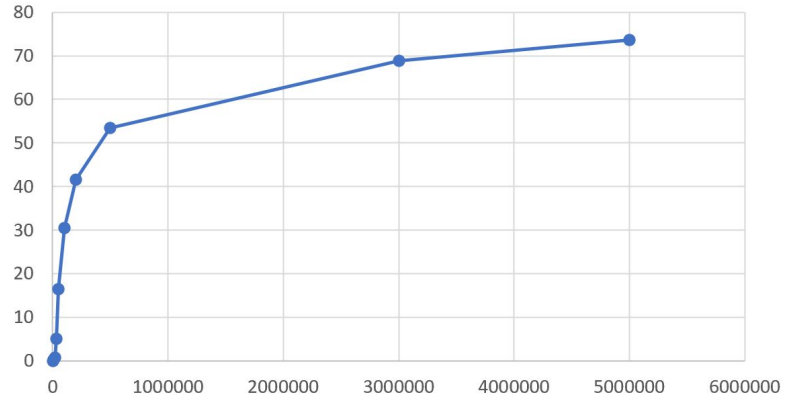
Training

We trained our model on **5,000,000 boards** for **10 epochs**

On the test set the model has reached an **accuracy of 62.35%**.

Convolutional neural network

Testing



Validation set of 1,000 games that the model has never been confronted with.

With the training of 5,000,000 boards, the model has reached a **win rate of 73.6%**.

Reinforcement learning

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Reinforcement learning

Training

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Reinforcement learning

Testing

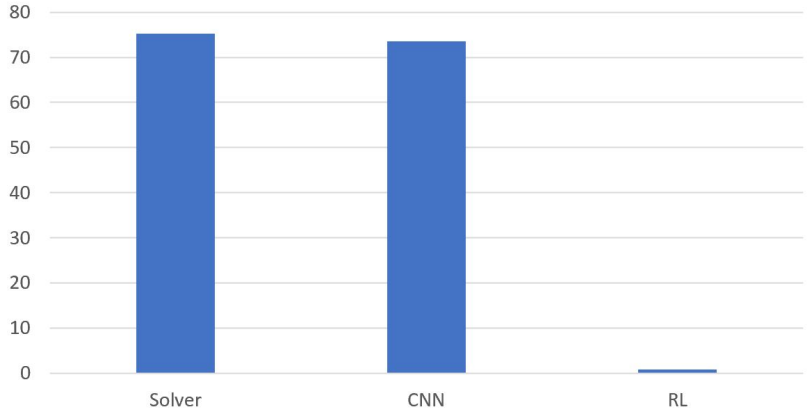
Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Demo

Conclusion

Conclusion

Win rate

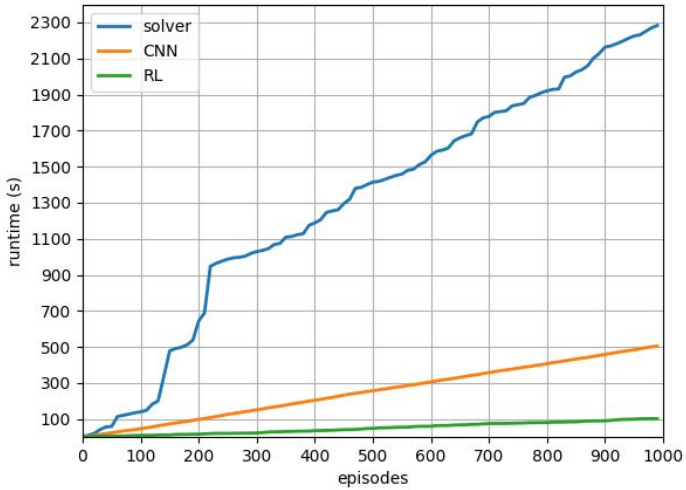


Each of the models played **1000 validation games** :

1. Solver **75.2% of win**
2. CNN **73.6% of win**
3. RL **0.8% of win**

Conclusion

Speed



Each of the models played **1000 validation games** :

1. RL took **1 min 40 sec**
2. CNN took **8 min 20 sec**
3. Solver took **38 min 20 sec**