Solving *Sudoku* with Neural Networks

Charles Akin-David and Richard Mantey
{aakindav, rmantey}@stanford.edu

Predicting

- Sudoku is one of the most popular logic-based games of all time, exploiting individuals' abilities for relational reasoning and pattern development.
- It relies heavily on temporal and positional dependencies between the numbers on the board
- This distinctive nature presented us with an opportunity to explore relational and visual interaction networks.
- To accomplish this we decided to attempt to solve these puzzles using Convolutional and Recurrent Neural networks where the inputs are unsolved sudoku quizzes and the outputs are fully solved sudoku boards!

Data

- **Data Source:** sudoku generator, created by Ariel Cordero: https://www.ocf.berkeley.edu/~arel/sudoku/main.html
- Fully generated boards are used as our labels
- Our x input is puzzles with 0's representing a blank in the board.
- Each (quiz, solution) pair becomes a training example.
- We generated 1,000,000 examples, kept 999,000 examples for our training set and set away 100 examples for our **train-dev** set.

Data Example

- Quiz: 005630070001082500806050904980004607020000030600590400450300210000100008070026300
- Solution: 245639871791482563836751924983214657524867139617593482459378216362145798178926345

Model & Features

- We trained two different models for this problem
  - CNN
  - Bi-LSTM (RNN)

CNN

- Model Architecture:
  - 15 convolution layers
  - First 10 of 512 filters each having a filter size of 3x3
  - Final layer - 1x1 filter size with 10 filters.
- We used **inference** to predict by adding an extra dimension to our input where, in the FC layer, the softmax probabilities for each label 1-9 are stored for each cell.
- The max of the probabilities was taken and the category with the highest probability (argmax) was chosen for that cell.

Bi-LSTM (RNN)

- Model Architecture:
  - 3 bidirectional LSTMS each with 1,2 or 3 layers
  - 200 hidden and memory units
  - 1 feed-forward layer - linear output, 200 neurons
  - Softmax layer - 9 neurons
- Problem was modelled to represent human approach to solving sudoku
- Board was represented as set of 3, 81 x 9 input vectors
- Each input was a concatenation across Rows or Columns or 3*3 mini Squares
- Input data then converted into one hot form to model the individual labels from 1-9
- Each input vector was fed into its particular LSTM with their outputs pooled together
- Softmax probabilities over the pooled output was taken and the category with the highest probability (argmax) was chosen for that cell.

References

1. "Human-level control through deep reinforcement learning." https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf.
2.

Results

Architecture	Training Accuracy	Test Accuracy
1 layer LSTM	0.791	0.467
2 Layer LSTM	0.814	0.467
3 Layer LSTM	0.814	0.467
10 Conv CNN	0.97	0.78
15 Conv CNN	0.96	0.81

Discussion

-

Future Work

- Hyperparameter Optimization
  With more time and resources, we would optimize our parameters further following Bergstra and Bengio's *Random Search for Hyperparameters Optimization*
- Derived Features
  We also could have worked further with Seibert et al. to see how we could develop some derived features for our model
- Stronger Neural Networks
  We also would have been able to do more research on different variations of neural networks