Mastering the game of Omok
predicting the next game moves using deep learning

Jisoo Min (jisoomin@mit.edu, /jisoomin)

TA Mentor: Martin Schneider (martinfs@mit.edu)
Industry Mentor: Stan Bileschi (bileschi@google.com)
Omok

- **What is Omok?**
  - played on a 15 X 15 board
  - two players alternate turns
  - place a stone of their color on an empty intersection
  - winner is the first player to form an unbroken chain of **five** stones horizontally, vertically, or diagonally

- **Restrictions on Black Stone Player**
  - *three and three*: bans a move that simultaneously forms two open rows of three stones
  - *four and four*: bans a move that simultaneously forms two rows of four stones
  - *six stones*: bans a move that forms an unbroken chain of **six** stones
Project Information

- **Goal**
  - train human-like neural networks for Omok
  - predict moves made by professional players
  - build an AI bot for the game

- **Personal goals**
  - have fun exploring deep learning with my favorite game Omok!
  - better understand the Omok moves
  - solve complex problems using deep learning
Datasets

- http://www.renju.net/downloads/downloads.php (~50k games, ~2m states)
- GomokuWorld.com (~100k games, ~5m states)

*RenjuLib software required to convert .lib files to .txt files
System Input and Output

- **Input**
  - 15 x 15 x 3 board state
  - three possible states per board position
    - black: [1,0,0]
    - white: [0,1,0]
    - empty: [0,0,1]

- **Output**
  - length 225 vector with each value representing the probability of that position being the next best move

```
INPUT = [
  [[0,0,1], [0,0,1], ..., [0,0,1]],
  [[0,0,1], [0,0,1], ..., [0,0,1]],
  ..., 
  [[0,0,1], [0,0,1], ..., [0,0,1]]
]
```

```
OUTPUT = [0.01, 0.05, 0.02, ..., 0.03, 
           0.06, 0.90, 0.05, ..., 0.01, 
           ..., 
           0.01, 0.02, 0.01, ..., 0.09]
```

\[ \text{len(OUTPUT)} = 225 \]
**Architecture**

```python
x = keras.layers.ZeroPadding2D(padding=3, name="padding_conv1")(inputs)
x = keras.layers.Conv2D(64, (7, 7), strides=(2, 2), use_bias=False, name="conv1")(x)
x = keras_resnet.layers.BatchNorm(axis=axis, epsilon=1e-5, freeze=freeze_bn, name="bn_conv1")(x)
x = keras.layers.Activation("relu", name="conv1_relu")(x)
x = keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding="same", name="pool1")(x)
```

**OPTIMIZER** = keras.optimizers.RMSprop(lr=LEARNING_RATE, rho=0.9, epsilon=None, decay=0.0)

model.compile(optimizer=OPTIMIZER, loss='categorical_crossentropy', metrics=['accuracy'])

---

rmsprop: Keep a moving average of the squared gradient for each weight

\[ \text{MeanSquare}(w, t) = 0.9 \text{MeanSquare}(w, t-1) + 0.1 \left( \frac{dE}{dw} (t) \right)^2 \]

\[ H(p, q) = - \sum_{x} p(x) \log(q(x)) \]
**Loss**

Train on 1508700 samples, validate on 282124 samples

Epoch 1/30
2018-05-12 01:18:20.545642: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA

```
1598700/1598700 [==============================] - 38418s 19ms/step - loss: 4.0861 - acc: 0.8768 - val_loss: 3.9095 - val_acc: 0.8861
```

Epoch 00001: val_loss improved from inf to 3.90948, saving model to outputs/renjunet/2018-05-12-01-18-epoch-30_batch-256_split-15/weights.01-3.91.hdf5

Epoch 2/30
```
1598700/1598700 [==============================] - 29999s 19ms/step - loss: 3.8290 - acc: 0.8153 - val_loss: 3.8110 - val_acc: 0.8142
```


Epoch 3/30
```
1598700/1598700 [==============================] - 29566s 18ms/step - loss: 3.7222 - acc: 0.8140 - val_loss: 3.7421 - val_acc: 0.8193
```


Epoch 4/30
```
1598700/1598700 [==============================] - 29372s 18ms/step - loss: 3.6520 - acc: 0.8145 - val_loss: 3.6678 - val_acc: 0.8128
```


Epoch 5/30
```
1598700/1598700 [==============================] - 29346s 18ms/step - loss: 3.6077 - acc: 0.8152 - val_loss: 3.6522 - val_acc: 0.8130
```

Epoch 00005: val_loss improved from 3.66784 to 3.65222, saving model to outputs/renjunet/2018-05-12-01-18-epoch-30_batch-256_split-15/weights.05-3.65.hdf5

Epoch 6/30
```
1278528/1598700 [==============================] - ETA: 1:40:05 - loss: 3.5092 - acc: 0.1540
```

**loss**: the value of cost function for the training data

**val_loss**: the value of cost function for the cross validation data

- **MIT Engaging Cluster (GPU support)**: 234 64GB, 2 x 8-core 2.0GHz CPUs, 90 K20m GPU, 16 Xeon phi, base OS - RHEL/Centos 6.4
- **Google Cloud**: 6 vCPUs, 32 GB memory
- **Personal machine**: 2.7 GHz Intel Core i7, 16 GB 2133 MHz LPDDR3
Move Prediction Quality

introduction

datasets

networks

performance

possible extensions
Future Steps

- **Finer Implementation**
  - give different weights to players
  - consider board rotations
  - filter datasets by game rules

- **More Intensive Computing**
  - train model on all 8 million datasets
  - train model on different parameters
  - test more architectures and evaluate performance

- **Usability**
  - set up an online demo so that anyone can try out the game
  - make sure to check out the link in a few days!
Thank you.

This project was supported by...

**Instructors:** Professor Hal Abelson, Natalie Lao  
**Project Manager:** Jessy Lin ([lnj@mit.edu](mailto:lnj@mit.edu))  
**Course Staff on Piazza:** ([6.s198-staff@mit.edu](mailto:6.s198-staff@mit.edu))

**TA Mentor:** Martin Schneider ([martinfs@mit.edu](mailto:martinfs@mit.edu))  
**Industry Mentor:** Stan Bileschi ([bileschi@google.com](mailto:bileschi@google.com))  
**MIT Engaging Cluster Staff:** Chris Hill and many more ([engaging-admin@techsquare.com](mailto:engaging-admin@techsquare.com))