Hooked on Phoenix: Deep Q-Learning on the Classic Atari Game
Evan Darke, Jake Smola, Michael Mernagh

Phonix
Phoenix is an Atari game in which the player controls a vehicle that moves left and right across the bottom of the screen, trying to avoid an oncoming land mine while also trying to find mines to destroy.

Challenges:
- This is a large space
- An agent only knows where a mine is when it is crossed.
- The agent only knows if an explosion occurs.
- The agent only learns when a mine is crossed.

Proposed:
- Use a convolutional network to reduce state representation and speed up learning.

Feature Extraction
- Method 1: (Autoencoder)
  - Train a deep convolutional autoencoder to compress the state representation into a factor of the state size.
  - Use output of autoencoder as input to feed forward network.
- Method 2: DeepQ-Learning
  - Downsample image by 2 and convert to grayscale.
  - Use CNN to extract features and learn Q function.

Convolutional Neural Network
We leverage a deep convolutional network (CNN) to optimize Q-values targets as our agent plays Phoenix. The CNN passes the pixel input through four convolutional layers before feeding the subsequent output into two fully connected layers that predict the Q-values.

Dueling Architecture
To improve performance, we follow the footsteps of Wang et al. in utilizing a dueling network architecture. This architecture decomposes the state input into a shared common branch and a private advantage branch. The advantage network is specifically trained for each state to improve policy stability and optimize the bellman equation:
\[ Q(s_t, a_t) = V(s_t) + A(s_t, a_t) \]

Double Q-Learning
To further improve performance, we also use a double Q-learning model which accepts either of the aforementioned models as its fit. This model prevents the model from learning on a single action, thus losing out on a specific performance, rather than the traditional deep Q-learning in the game of Phoenix. Taking a similar approach, we made our fit changes as possible to the original deep Q-learning steps to derive a double Q-learning model. We leverage the stability of both networks and optimize the bellman equation:
\[ \min_{\theta} Q(s_t, a_t) = R_{s_t,a_t} + \gamma \max_{a_t'} Q(s_{t+1}, a_t') \]

Model Performance
- The traditional deep learning model suffers from significant overfitting throughout training and only recently improves the performance of other models throughout training.
- The CNN model struggles to converge in the first 100k steps, with no improvements seen in the last 100k steps.
- The double CNN model, double dueling CNN, and dueling CNN models performed very similarly for the first 100k steps, with improvements seen in the last 100k steps.

Conclusion
- The double Q-learning model appears to continue to improve well into the end of training while both double CNN models remain constant.
- The finding is slightly surprising as the expectation is that the benefit of having a learning and dueling network can be computationally demanding and slower learning could affect this initial outcome.

Analysis
- The dueling model shows a fraction of the memory used by the other models to learn, but still shows a performance gain over the fully connected network.

Future Work
- Transfer learning from other games (e.g., speed running) may speed learning.
- Additional exploration using replay buffers may help retain the model.
- Sufficient hyperparameter tuning can lead to significant gains, as seen from Wang et al.

References