Combining a Mixture of Experts with Transfer Learning in Complex Games

Mihai S. Dobre and **Alex Lascarides**
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB
Scotland
Email: m.s.dobre@sms.ed.ac.uk, alex@inf.ed.ac.uk

Abstract

We present a supervised approach for learning policies in a highly complex game from small amounts of human data consisting of state–action pairs. Our Neural Network architecture can adapt to the varying size of the set of legal actions, thus bypassing the need to hardcode the actions in the output layer or iterate over them. This makes the training more data efficient. We use synthetic data created via game simulations among AI agents to show that a mixture of experts, where each expert predicts actions in different portions of the game, improves performance. We then show that this approach applied to human data also improves performance: in particular, using transfer learning to enable one expert to learn from another enhances performance on those portions of the game for which there is relatively little training data compared to other portions. The domain chosen for evaluation is the board game *Settlers of Catan*.

Introduction

Deep Neural Networks (DNN) (Hinton 2007) have been successfully applied to many tasks such as visual processing (Lecun et al. 1998; Krizhevsky, Sutskever, and Hinton 2012) and speech (Deng, Hinton, and Kingsbury 2013). They are particularly suited to tasks where abstract formalisations of the domain are not available, but large amounts of training data are available.

Recently, DNNs have been combined with reinforcement learning algorithms (Mnih et al. 2015) and applied to supervised learning in complex games (Silver et al. 2016). In very long complex games, the policy used at the beginning of the game may differ to those used later. Furthermore, the optimal policies required to handle each phase of the game may be orthogonal, or too complex, to be represented by a single model. For these reasons, the best results in shooter games are achieved by modularising the models into a *mixture of experts*, where each expert specialises on one of the portions of the game (Tastan and Sukthankar 2011; Lample and Chaplot 2016). These approaches reduce the number of actions an agent has to consider, making the Q-function simpler to learn (Gaskett, Wettergreen, and Zelinsky 1999). The result is a much shorter training period and improved performance of the final agent.

In spite of this progress, there are certain limitations of the above models that we attempt to address in this paper. Firstly, none of them are easily extended to games where the set of all legal actions throughout the game is huge. In such situations, the model should be able adapt to the set of *current* legal actions. Secondly, the current approach to handle more complex games is to increase the depth of the model and fit more data during the training period. But in contrast to Go, large amounts of human (expert) data may simply not exist for the game you aim to model.

In this paper, we present an integrated suite of techniques for enabling deep learning to cope with small amounts of training data. We present a mixture of experts model, where each network specialises on a specific phase of the game, and we then take advantage of the larger amounts of data available for certain phases by deploying *transfer learning* to those phases for which very little training data is available. We will show the benefits of using a mixture of experts on synthetic data gathered from agent simulations in *Settlers of Catan*, followed by applying transfer learning on data gathered from human participants playing the game.

Background

Most deep learning models require large amounts of data, many iterations and a large computational budget to converge to a reasonable result. In situations where very little data is available, the large number of parameters causes these methods to overfit. In addition, many of the existing approaches in games use the raw data (Mnih et al. 2015), thereby enhancing the model's appetite for very large data sets to learn useful abstractions. An alternative is to treat the game representation as a 2D signal (e.g. in Go, Silver et al. 2016), so that a Convolutional Neural Network model can be employed. But some games cannot be easily represented this way.

Recent research has focused on games that have a fixed set of possible actions irrespective of the current state (Mnih et al. 2015). Thus the network architecture is such that the final layer size is equal to the number of actions, while the input is the current state description. But this approach is not feasible in games where the number of legal actions is highly dependent on the current state; nor does it scale well

to games where the number of legal actions is much larger than in games like Go; for instance Civilisation, Diplomacy, and Settlers of Catan, the game we study in this paper. To address this gap, we will design our network so that it gains efficiencies from the fact that only a subset of the possible actions are legal at any one state of the game.

Other implementations evaluate each state-action pair iteratively, without taking into account the other legal actions in the given state (Branavan, Silver, and Barzilay 2011) or as a batch update over the full dataset (Riedmiller 2005). The target values are computed based on the game result following Monte-Carlo simulations or by allowing the learned policy to interact with the environment. This approach, however, requires access to a decent simulation policy or some way of evaluating the samples. We, on the other hand, want to extract a policy from a dataset of state-action pairs. To achieve this we assume that the executed action is the player's preferred (optimal) one given its policy and all available options.

Mixture of experts

Ensemble learning is a popular approach for supervised learning that uses multiple models to increase performance and reduce overfitting. A popular approach is the Mixture of Experts model that is based on the divide and conquer principle (Masoudnia and Ebrahimpour 2014). Typically, a set of networks are trained alongside a gating network. The gating network chooses which expert to use for each sample (Jacobs et al. 1991). Another approach is to explicitly partition the space beforehand. Here we do the latter, by taking advantage of the existing game structure to predefine a function that chooses the expert deterministically.

Multitask and Transfer Learning

Multitask learning improves generalisation by learning multiple tasks in parallel while using the same representation (Caruana 1997). Here, we also partition the problem into tasks—in our case, each task corresponds to a particular phase in the game. However, we don't train the tasks in parallel because the training data for one game phase is different to that of another. Nevertheless, we still want to transfer what was learned about one task to another. In computer vision, for instance, features learned on one task are general and transferable to other tasks (Yosinski et al. 2014), even if the samples do not come from the same distribution (Bengio et al. 2011). More generally, transfer learning has proved useful in both unsupervised (Mesnil et al. 2012) and supervised learning (Thrun 1996), but it matters how transferable the learned features are. Due to the rules of *Settlers of Catan*, the training datasets for some tasks are significantly smaller than for others while the representation is the same, so transferring the learned features from one task to another may help improve performance. Furthermore, transfer learning should keep the experts consistent and the resulting combined policy will be closer to the policy used to play the games from which the samples were collected.

Settlers of Catan

Prior work on Settlers of Catan mostly focused on (online) reinforcement learning (Szita, Chaslot, and Spronck 2010; Dobre and Lascarides 2015) or developing heuristics based on expert knowledge (Guhe and Lascarides 2014), rather than supervised learning from attested game play. The only Deep Neural Networks implementation on the game is done by Cuayáhuitl, Keizer, and Lemon (2015), who focus only on predicting the resources that the opponents will trade during the negotiations phase. Their approach is an extension of the DRL algorithm with experience replay (Mnih et al. 2015). So they handle illegal actions by first allowing the network to reason over them and only exclude these when the best option is chosen. In contrast, our model learns *all* the phases of the game in a supervised fashion, not just the negotiations part, and it doesn't require reasoning over possible but illegal actions in the current state.

Resources

Settlers of Catan

Settlers of Catan is a multi-player win-lose board game. We focus on the core game for 4 players. The players build roads, settlements and cities on the board, which is formed of hexagonal tiles. The first player to reach 10 victory points wins the game. One obtains victory points in a variety of ways (e.g. a settlement is 1 point and a city is 2 points). Board tiles represent one of the five resources (clay, ore, sheep, wood and wheat), desert, water or ports. Each of the resource producing tiles has an associated number between 2 and 12. Players obtain resources via the location of their buildings and dice rolls that start each turn of the game. One needs different combinations of resources to build different pieces (e.g. a road costs 1 clay and 1 wood). In addition to dice rolls, players can acquire resources through trades with the bank (at a 4:1 ratio), or with a port if they have a settlement or city there (3:1 or 2:1, depending on the port) or through negotiated trades with other players. There are many special actions which increase the complexity of the game, such as playing development cards that each give different advantages; e.g. moving the robber and stealing resources from other players or gaining victory points via the longest road or largest army (see www.catan.com for details). The only modifications that we make to the game rules is that our models only consider 1:1, 1:2 and 2:1 resource trades with other players. This simplification retains a large set of possible trades (up to 540).

Game analysis

From a game theoretic perspective, Settlers of Catan is a very complex game. In addition to the incomplete information (i.e. the opponent's types), the game contains elements of imperfect information (the resources that opponents possess and the unplayed development cards) and it is stochastic (dice rolls determine the players' resources). It has a large branching factor e.g. there's a wide range of negotiation actions for trading resources. The generation of the board is done by shuffling the hexes, so the game has a huge space of possible initial states ($\approx 1.2 * 10^{15}$ compared to 1 for

the game of Go).[1] Table 1 contains approximations of the branching and depth factors, given 3 different sampling policies: Human is based on 60 human games (Afantenos et al. 2012), Heuristic on 1000 simulated games played by four rule-based AI agents and Random on 1000 simulated games with 4 players choosing actions randomly. Due to how these games have been logged, trades are considered as a single exchange action and negotiations (i.e. offer, accept and reject actions) are not counted, hence the depths reported here are smaller than the real depths.

Policy	Branch	Depth
Human	63	152
Heuristic	69	234
Random	64	11639

Table 1: Average branching and depth. Many of the human games have 2 or 3 players, hence the smaller depth.

While Settlers of Catan is highly complex, it also has a clear structure. Actions can be grouped into several types, e.g. trade actions or build road actions. Depending on the current state description, only certain action types are legal. We use a deterministic function to separate the game into 6 phases, based on what action types are legal, such that there is minimum overlap between the phases:

0 Free road building; this is either the initial state where the agent places two free roads, or the two free roads that stem from having just played a 'road building' development card;

1 Free settlement building, during the initial placement part of the game;

2 The normal state: this is a state where the agent can build, trade, buy/play development card or end their turn;

3 Before rolling the dice state: the agent can decide between playing a development card or rolling the dice;

4 Discard state due to a 7 being rolled: the agent has to discard half of the resources it is holding;

5 Moving robber state due to a 7 being rolled: the agent has to move the robber and steal one resource from an opponent.

While our experiments are on *Settlers of Catan*, many other games exhibit the same properties. Most complex games can be separated into phases in which different types of actions are legal. In fact, this separation is clearly presented in the game rules for Civilisation, Diplomacy, Battlestar Galactica, Cosmic Encounter, to name a few. Our approach could be applied to any games with such characteristics.

Software and Data

The synthetic data was collected from 4-player game simulations in the JSettlers framework using a state of the art heuristic agent as the players (Guhe and Lascarides 2014). The human data has been collected from 60 human games. We use only the trades that have been executed in the game during training. This may diverge from the player's preferred trade option in some cases—we will train on the negotiation actions in future work. We build our DNN model by extending the Deeplearning4j version 0.4-rc3.10 (Deeplearning4j 2016).

Features

The input is a 1D vector of numerical features, taken directly from the game state representation plus a set of features that abstract over the coordinates of pieces and board configuration. The abstraction is necessary for effective learning from the very small training sets of human data that we have access to. Information on the number of pieces on the board of each type, the robber, longest roads, award owners and the visible information on development cards and player's hands is taken directly from the game state. The exact location of each piece is represented via their significance with regards to the board description by including the information on resource production, the expansion possibility (i.e. if the player is blocked) and future production of the closest legal locations. Since we are abstracting state descriptions from the board coordinates, we need to also represent the actions as feature vectors. We achieve this by subtracting the state vector in which the action was executed from the outcome state vector. Only the features that can be modified by actions are included in the feature vector describing an action. The imperfect information is handled by assuming that the players have no information aside from the observable portion of the game (since the human games do not include information on the players' beliefs). For non-deterministic actions, specific features are computed using the expected outcome following the game rules (e.g. resource production is decided based on the chances of rolling two six-sided dice). The action of stealing is the only exception, since the player executing the action doesn't know the victim's hand. None of the features presented above inform the model of the benefits of certain actions. There are a total of 157 state features and 73 action features. The complete list of features, including their value ranges, will be released with the code at https://github.com/sorinMD/deepCatan.

Model description

Our goal is to make our network adaptable to the number of legal actions in the current state. We inform the network of this number—let's call it n—via the shape of the input. So, the input is a matrix, where each row is a vector of features that represent the current state and one of the legal actions. This is achieved by replicating the state vector n times, concatenating each of them with the vector describing one of the legal actions and stacking the resulting vectors. The resulting matrix has n rows and 230 columns, where n is the number of legal actions and 230 (157+73) is the size of the feature vector resulting from concatenating the state and action representations. This stacked input can be seen as a minibatch. However, the size of this minibatch varies since the number n of legal actions varies from one state to another.

[1] We are not shuffling the production numbers, which would increase the space further.

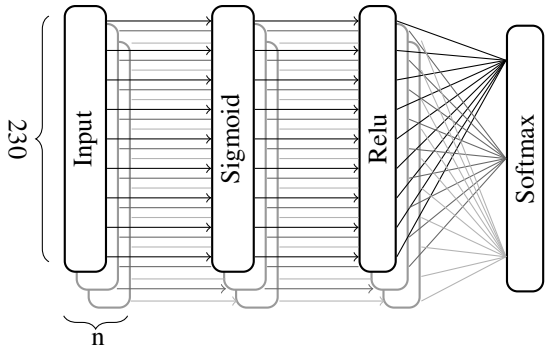

Figure 1: The DNN model

The network is a feedforward neural network with two hidden layers, as shown in Figure 1. Both hidden layers are fully-connected with 256 nodes. The first layer has a sigmoid activation function while the second one uses a rectifier activation function. The output layer is a softmax layer with a single unit. So we achieved the adaptable output layer by turning the multi-class problem into a single class with multiple inputs. The network doesn't predict which is the correct class given the input, but rather what is the likelihood of each input being the correct one (taking into account the other options).

We changed the softmax formula such that it iterates over each input, as shown in equation 1. There is a single class y with the corresponding set of weights w, but there are multiple inputs x. Since the partial derivative of this function with respect to its input $w^T x$ is equivalent to that of the standard softmax, the derivative of the error with respect to the network's weights is the same and the training can be done via the standard backpropagation algorithm.

$$P(y|x) = \frac{e^{w^T x}}{\sum_{x'} e^{w^T x'}} \tag{1}$$

Even though the softmax output layer is made of a single unit, it can handle any number of rows of the input matrix. In fact, the network architecture is equivalent to evaluating each state-action pair iteratively and turning the resulting vector into a probability distribution. This approach assumes a mutual exclusivity between the available state-action pairs, forcing the network to rank the options in the order of preference according to their features. It also normalises the output, making the training more stable. The alternative to our approach is to consider each state-action pair separately and perform a binary classification while iterating over each sample. The large branching factor of the game would cause the data to be skewed towards the negative case, resulting in poor performance and unstable training. Another advantage to our architecture is the ability to evaluate all possible actions in a given state with only a single forward pass through the network. It is also straightforward to extend to the minibatch (of states) case. Most of the forward pass would be computed in parallel, except for the softmax operation which needs to be computed iteratively on each portion of the input corresponding to a set of state-action pairs.

Since our features are numerical, we normalise the input such that it has 0 mean and unit variance. During training we feed the network one sample at a time, where each sample is a set of the legal state-action pairs stacked in a matrix format as described above. The updates are done using RMSProp:[2] they are performed using the minibatches of size equal to the number of legal actions. We noticed a large improvement when using RMSProp compared to the simple update rule. We also tried using minibatches of states, but these gave no improvements. This could be caused by the lack of data or by the fact that the number of legal actions has a huge variance throughout the game. We initialised the network's weights using the Xavier algorithm (Glorot and Bengio 2010).

The size of the network, activation functions, initialisation and update methods were chosen based on the performance on a different evaluation set made of synthetic data. Since this model will eventually be combined with a sampling method to create an agent that takes decisions in an online manner, the prediction time was another reason for keeping the size of the model relatively small compared to the state of the art in other domains.

Our best model is a mixture of experts model with 6 experts, one for each phase of the game. Since we split the game via a deterministic function, all training data is separated beforehand. We train each expert on the corresponding dataset by minimising the cross-entropy loss. Due to the abstraction described in the previous section, the target action that the network tries to learn may be encountered twice in the list of legal actions. We handle this rare case by giving equal weight to each of the corresponding indices in the target vector, so the sum of the vector is still equal to 1. During evaluation either option is considered correct.

The human data consists of only 60 games. Due to the game rules, the data available for some of the tasks is insufficient to achieve a decent prediction accuracy (e.g. 200 samples for task 4). On the other hand, there are many more samples available for other tasks (e.g. 6k training samples for task 2). To increase the performance on tasks that lack data, we use transfer learning: we initialise the network's weights by pretraining for several epochs on the data available for the other tasks. The number of epochs was finetuned to the specific task, e.g. the best performance on task 4 was achieved after 10 pretraining epochs, while task 5 required only 5 epochs. Previous work has evaluated the performance of the network after transferring only part of it, while randomly reinitialising the weights of the rest. We achieved the best performance by transferring the weights of the full network, which enforces our belief that both the representation and the policy captured during pretraining contributed to the increase in performance.

Results and analysis

We will present two set of experiments: one on the synthetic data and one on the human data. For each of these we com-

[2] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

pare the DNN model against a frequency based baseline, which chooses the most frequent action from the training set. We will show the need for a mixture of experts on the synthetic data, by comparing a single model that fits all 6 tasks against a combination of 6 models, one for each task. We have access to a large amount of synthetic training data, therefore transfer learning would not improve the performance of the models. However, we will show the benefits of transfer learning when training on the small human dataset. We report our results as *accuracy* defined as the number of times the model selects the correct action over the total number of samples. This choice reflects the goal of learning to imitate the policy that generated the data and not predicting the actions' class.

Baseline

Table 2 contains the accuracy of the frequency based model on the two datasets. For this experiment only, the human data was split in two sets: 90% for training and 10% for evaluation. Both the single model and the mixture of experts easily outperform this baseline. The inability of the baseline to select the correct option on the initial placement task is most likely due to the large number of initial board positions. It is very unlikely that the same action will be available and also chosen by the rule-based agents across multiple games. On the other hand, task 3 contains the roll action which is very often chosen over the alternative of playing a knight card. Also, it seems the human players prefer ending their turn over other actions in the normal task (i.e. task 2). Perhaps, this is due to a preference for waiting to roll the needed resources over trading with the opponents. Despite the high baseline performance, the mixture of experts model with or without transfer learning is significantly better (see tables 5 and 6).

Task	Synthetic		Human	
	Eval	Train	Eval	Train
0	29.09%	28.82%	39.77%	33.52%
1	0%	0%	0%	0%
2	6.58%	6.42%	50.51%	50.98%
3	75.09%	75.22%	77.17%	74.25%
4	11.18%	10.36%	13.16%	4.55%
5	8.33%	8.06%	4.71%	4.8%

Table 2: Baseline performance on both datasets.

Synthetic data

The results in Table 4 show that the mixture of experts model scales with the increase of the size of training data. The "Eval" column is the model's performance on 10k held-out test data; the "Train" column is the model's performance on the training data (to test for overfitting). The results of the single model are shown in Table 3. It is trained on all 300k (i.e. $6 \times 50k$) training samples, by iteratively fitting one sample from each task similar to the multi-task approach. We noticed this resulted in a much higher performance than just randomising over the whole dataset. Even so, the mixture of experts model trained on the full dataset significantly outperforms the single model on all 6 tasks (compare Table 3 and the last 2 columns of Table 4): we used the z-test and a threshold $p < 0.05$ as this significance test, with the null hypothesis that the performance of the mixture of experts and singe models are equivalent.

Task	Evaluation	Train
0	67.01%	66.88%
1	65.98%	65.75%
2	48.35%	47.30%
3	80.07%	80.38%
4	38.87%	40.36%
5	23.16%	24.51%

Table 3: Single model performance on synthetic data.

Human data

In the interest of space we will not include the results of the single model. This model presented slightly better performance compared to the standard mixture of experts, indicating the benefits of sharing the knowledge between the phases when small amounts of data is available. However, we observed that the training was very unstable and the accuracy on each task fluctuated rather than having a steady increase with the epochs. Also, when the best performance on one of the tasks was observed, the performance on others was relatively weak, making it impossible to decide when to stop the training. The most likely cause is the lack of data in combination with the large difference in the size of the training sets. This is a characteristic of complex games: or mixture of experts model addresses this.

We evaluated our models with 10-fold cross-validation over the whole set of human samples. In addition to displaying the evaluation and training accuracy, we included the 90% confidence intervals in the tables. Due to the lack of data and the multitude of policies responsible for generating the human data (Afantenos et al. 2012) a weaker performance compared to the results on the synthetic data is expected, as observed in tasks 0, 1 and 4 in Table 5. The results for the remaining tasks are very similar. However, the performance on the move robber task is better. We believe this may be due to the features chosen, which may enable the learning algorithm to learn to explain a human behaviour much better than that of a complex heuristic agent that performs obscure computations over and above those including the visible features.

Task	Evaluation	Eval CI.	Train
0	47.04%	$\pm$3.85	50.82%
1	22.56%	$\pm$3.31	27.63%
2	59.62%	$\pm$0.95	61.71%
3	81.3%	$\pm$3.29	82.89%
4	22.1%	$\pm$4.95	25.02%
5	37.76%	$\pm$3.12	47.59%

Table 5: Mixture of experts performance on human data.

Task	Size of training set							
	5k		10k		25k		50k	
	Eval	Train	Eval	Train	Eval	Train	Eval	Train
0	60.23%	59.32%	61.67%	62.45%	66.45%	71.50%	68.51%	70.90%
1	72.23%	76.74%	71.86%	73.9%	73.21%	73.96%	73.83%	73.45%
2	55.81%	55.97%	56.86%	57.90%	58.35%	58.39%	58.65%	59.33%
3	80.72%	85.1%	81.28%	85.31%	81.16%	86.11%	81.49%	84.76%
4	36.26%	54.34%	39.50%	46.48%	42.24%	55.42%	42.24%	55.42%
5	24.78%	33.24%	27.57%	39.34%	29.9%	42.08%	29.67%	36.92%

Table 4: Accuracy on synthetic data of the mixture of experts model while varying the amount of data available for training[3].

As expected, transfer learning works best for fitting the tasks where less data is available, i.e. the initial placement(1) and discard tasks(4) (see Table 6). Even though the effect of the actions is not exactly the same across different tasks (e.g. placing the initial settlements yields an *immediate* production, in contrast to building a settlement in the normal part of the game), the weights learned over the other tasks help the neural net explain why certain features are important for the target task also. We didn't see any improvements for the normal (2) and free road (0) tasks. The training data for task 2 is sufficient, while task 0 bares very little resemblance to any other task. The feature abstraction is also causing issues in task 0, as most of the cases where actions are confused with each other are included in this dataset. In future work, we aim to improve the abstraction performed.

Task	Evaluation	Eval. CI.	Train
0	46.59%	±4.86	51.94%
1	27.44%	±4.23	32.11%
2	59.57%	±1.02	62.79%
3	83.69%	±1.85	87.25%
4	32.11%	±3.75	49.71%
5	38.71%	±2.39	40.23%

Table 6: Transfer learning between experts on human data.

The goal of this research is to have a decent policy estimation that can predict human play, so we can bias the search during online learning. Therefore we are also interested in how the network ranks all the legal actions and how far the correct action is from the network's preferred option. Figure 2 shows the accuracy of the mixture of expert model as we allow the correct option to be in the first n options in the output (the value of n is given by the x axis). The accuracy for task 0 has an unusual curve due to most of the samples only containing 3 options, while the other tasks have many more than 10 legal actions (given the branching factor of the game tree). Overall, the correct option is within the model's top 10 options 80% of the time. Therefore the model could be used to perform soft pruning of a search tree during online learning.

[3]One epoch of training over 50k data on task 2 where the branching factor is 89, takes approximately 2 and a half minutes on one Nvidia GTX TITAN X GPU.

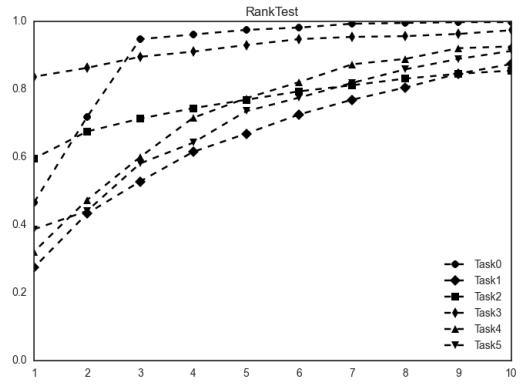

Figure 2: Percentage of correct classifications up to the first 10 choices of the best model trained on human data.

Conclusion

We have presented a method of enhancing the supervised learning procedure in a very complex game, where the set of possible actions irrespective of the current state is huge. Our neural network architecture adapts to the current set of legal actions and evaluates them simultaneously. By excluding the set of illegal actions in the current state and evaluating only the legal actions using the same set of weights, the amount of data required is reduced. The generality of the architecture allows similarities between training datasets to be exploited via transfer learning, further increasing the model's performance on portions of the game where small amounts of data are available. Further benefits stem from combining transfer learning with a mixture of experts. In future work, we will compare our adaptable architecture to standard architectures while employing different methods of handling skewed datasets or large multi-class classification problems. We will also combine our best model with a sampling method, with the aim of creating a strong Settlers of Catan player. The end goal is to show that even small amounts of human data can be used to create a strong player in complex games.

Acknowledgments

We thank the reviewers for their helpful suggestions. This work is supported by ERC grant 269427 (STAC) and Engineering and Physical Sciences Research Council (EPSRC).

References

Afantenos, S.; Asher, N.; Benamara, F.; Cadilhac, A.; Degremont, C.; Denis, P.; Guhe, M.; Keizer, S.; Lascarides, A.; Oliver Lemon, P. M.; Paul, S.; Rieser, V.; and Vieu, L. 2012. Developing a corpus of strategic conversation in the settlers of catan. In *Proceedings of the 1st Workshop on Games and NLP*.

Bengio, Y.; Bastien, F.; Bergeron, A.; Boulanger-lewandowski, N.; Breuel, T. M.; Chherawala, Y.; Cisse, M.; Ct, M.; Erhan, D.; Eustache, J.; Glorot, X.; Muller, X.; Lebeuf, S. P.; Pascanu, R.; Rifai, S.; Savard, F.; and Sicard, G. 2011. Deep learners benefit more from out-of-distribution examples. In Gordon, G. J., and Dunson, D. B., eds., *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, 164–172. Journal of Machine Learning Research - Workshop and Conference Proceedings.

Branavan, S. R. K.; Silver, D.; and Barzilay, R. 2011. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 268–277. Stroudsburg, PA, USA: Association for Computational Linguistics.

Caruana, R. 1997. Multitask learning. In *Machine Learning*, 41–75.

Cuayáhuitl, H.; Keizer, S.; and Lemon, O. 2015. Strategic dialogue management via deep reinforcement learning. *CoRR* abs/1511.08099.

Deeplearning4j, D. T. 2016. Deeplearning4j: Open-source distributed deep learning for the jvm, apache software foundation license 2.0. http://deeplearning4j.org.

Deng, L.; Hinton, G.; and Kingsbury, B. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *in Proc. Int. Conf. Acoust., Speech, Signal Process*.

Dobre, M., and Lascarides, A. 2015. Online learning and mining human play in complex games. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*.

Gaskett, C.; Wettergreen, D.; and Zelinsky, A. 1999. Q-learning in continuous state and action spaces. In *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence*, AI '99, 417–428. London, UK, UK: Springer-Verlag.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Guhe, M., and Lascarides, A. 2014. Game strategies in *the settlers of catan*. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*.

Hinton, G. E. 2007. Learning multiple layers of representation. *Trends in Cognitive Sciences* 11:428–434.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural Comput.* 3(1):79–87.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 1106–1114.

Lample, G., and Chaplot, D. S. 2016. Playing FPS games with deep reinforcement learning. *CoRR* abs/1609.05521.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 2278–2324.

Masoudnia, S., and Ebrahimpour, R. 2014. Mixture of experts: a literature survey. *Artificial Intelligence Review* 42(2):275–293.

Mesnil, G.; Dauphin, Y.; Glorot, X.; Rifai, S.; Bengio, Y.; Goodfellow, I. J.; Lavoie, E.; Muller, X.; Desjardins, G.; Warde-Farley, D.; Vincent, P.; Courville, A. C.; and Bergstra, J. 2012. Unsupervised and transfer learning challenge: a deep learning approach. In Guyon, I.; Dror, G.; Lemaire, V.; Taylor, G. W.; and Silver, D. L., eds., *ICML Unsupervised and Transfer Learning*, volume 27 of *JMLR Proceedings*, 97–110. JMLR.org.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Riedmiller, M. 2005. Neural fitted q iteration first experiences with a data efficient neural reinforcement learning method. In *In 16th European Conference on Machine Learning*, 317–328. Springer.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529:484–503.

Szita, I.; Chaslot, G.; and Spronck, P. 2010. Monte-carlo tree search in settlers of catan. In van den Herik, H., and Spronck, P., eds., *Advances in Computer Games*. Springer. 21–32.

Tastan, B., and Sukthankar, G. R. 2011. Learning policies for first person shooter games using inverse reinforcement learning. In Bulitko, V., and Riedl, M. O., eds., *AIIDE*. The AAAI Press.

Thrun, S. 1996. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, 640–646. The MIT Press.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 3320–3328.