Optimized Non-visual Information for Deep Neural Network in Fighting Game

Nguyen Duc Tang Tri, Vu Quang and Kokolo Ikeda
School of Information Science, JAIST, 1-1 Asahidai, 923-1292, Nomi, Ishikawa, Japan

Keywords: Deep Learning, Fighting Game, Convolutional Neural Network.

Abstract: Deep Learning has become most popular research topic because of its ability to learn from a huge amount of data. In recent research such as Atari 2600 games, they show that Deep Convolutional Neural Network (Deep CNN) can learn abstract information from pixel 2D data. After that, in VizDoom, we can also see the effect of pixel 3D data in learning to play games. But in all the cases above, the games are perfect-information games, and these images are available. For imperfect-information games, we do not have such bit-map and moreover, if we want to optimize our model by using only important features, then will Deep CNN still work? In this paper, we try to confirm that Deep CNN shows better performance than usual Neural Network (usual NN) in modeling Game Agent. By grouping important features, we increase the accuracy of modeling strong AI from 25.58% with a usual neural network to 54.24% with our best CNN structure.

1 INTRODUCTION

Nowadays with the explosion of data, Deep Learning has become one of the most popular research fields with its efficient in modeling and learning from data. When we talk about Deep Learning, it means we talk about the neural network with many layers and their structure (how neurons in one layer connect to the ones in other layers). Based on the problem, we have to choose the fittest structure to solve it. For example, in image processing, for MNIST dataset-digit number from '0' to '9' (Nielsen, 2015) and CIFAR 10-subset of tiny image dataset (Krizhevsky et al., 2012), Deep Convolutional Neural Network (Deep CNN) are chosen and shows the good performance. In game informatics, some video games which can be represented as a bitmap (image) or some board games, Deep CNN are also chosen as an effective approach. But the question why does CNN work is still a mystery, somehow when we group the neighborhood neuron together by a window and slide this window through the image (we usually call it filter) we get better information about this image.

Even when DeepMind-Google published their research (Atari 2600 games) and claimed that they succeeded in training Deep CNN as a network structure with Q-learning algorithm, they only confirmed that using a Deep CNN to represent a Q-function is better than a usual Q-Learning (Mnih et al., 2015). After that, Kapathy in his research with Pong-game (Karpathy, 2016), proved a usual neural network (one input layer, one hidden layer, and one output layer) can also be trained by a more general algorithm called Policy Gradient. There has been no work compare the performance of Deep CNN and usual neural network in case inputs are features instead of an image.

In this research, we do our experiment with the fighting game and we select FightingICE, an environment developed and maintained by Intelligent Computer Entertainment Lab, Ritsumeikan University (Lu et al., 2013). In FightingICE environment, images of the game are not available. Instead, AIs will receive features such as hitpoint, energy level, or in-game positions as input. That information is necessary for our experimental purposes. We want to verify the effectiveness of position of features when they are used as input in a convolutional neural network.

2 RELATED WORKS

In this section, we introduce related works using CNN in modeling and training strong AI.

2.1 CNN in Modeling Strong AI

Modeling a strong AI is a difficult task because it requires a huge dataset and careful optimization of many parameters. The more actions that AI can per-

Duc Tang Tri N., Quang V. and Ikeda K.
Optimized Non-visual Information for Deep Neural Network in Fighting Game.
DOI: 10.5220/0006248106760680
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 676-680
ISBN: 978-989-758-220-2

form the more difficulty we get. In particular, in the game Go, it seems impossible for researchers to model and predict the next move of the opponent. But, with Deep CNN, finally, researchers are succeeded in that hard task. Christopher Clark used 81,000 professional games dataset and an average network structure (four convolutional layers, one fully-connected layer) and he gets a good accuracy 41% in predict the next move (Clark and Storkey, 2014). After that, DeepMind even did the impossible work when creating the strong AI AlphaGo that defeat the human champion. In supervised learning step, AlphaGo used 13 layers, 30 million samples, training by 200 GPUs in 2 weeks and get an amazing accuracy 57% (Silver et al., 2016).

2.2 CNN in Training AI via Reinforcement Learning

The results in Atari 2600 games show that CNN works quite well in 2D games. Taking an 84x84x4 color image as an input, the agent can learn to distinguish many different states. It also has the ability to evaluate how good a state is by calculating a 'quality' base on the pair state-action-value. As it is successful, DeepMind even copyrights the algorithm as the name Deep Q-Network.

After that, in research with Doom game, Micha Kempka and his team confirm that CNN also works well in 3D games. Using Deep Q Network, the agent can be trained from an input image that contains depth information, and it can clear many scenarios from easy to hard smoothly (Kempka et al., 2016).

But, until now, there have no studies about using features instead of pixel image for the input of Deep CNN. The current hypothesis is that the relationship of neighboring pixels contains high-level information that can make our model better if we group them together by using a window.

3 GROUPING AND LOCATING FEATURES IN CNN

In image processing, dividing a big image into many sub-images is a common technique. For each sub-image, we can extract some important local features. This is the background knowledge for convolutional neural network. By using a window slide from left to right, from top to bottom through the image, CNN can learn local information. Then, this information is generalized in later layers to get a deeper understanding of the image. For example in Figure 1, CNN can learn to distinguish the image of digit number '0' from other numbers by its curve.

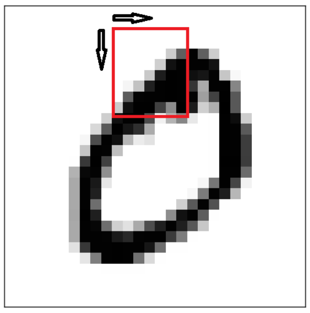

Figure 1: A window is a small sub-image. A CNN can learn the curve of digit number '0' by sliding the window through the image.

However, in the fighting game, we have standalone features which contain fixed information. For example, 'hitpoint' indicates the health of characters, 'energy' indicates the ability to use special skills, 'distance' may indicate safety level (the further the distance between our character and opponent's, the safer our character gets) etc. When we combine some of these features, we can get rules that strongly effective in decision making. For example, if we combine 'distance' and 'energy' we can get some rules like this:

$$\begin{cases} \text{throw energy,} & \text{if distance} > 300 \text{ \& energy} > 60 \\ \text{normal kick,} & \text{if distance} < 300 \text{ \& energy} < 60 \\ \text{hard kick,} & \text{if distance} < 300 \text{ \& energy} > 60 \\ \text{jump,} & \text{otherwise} \end{cases}$$

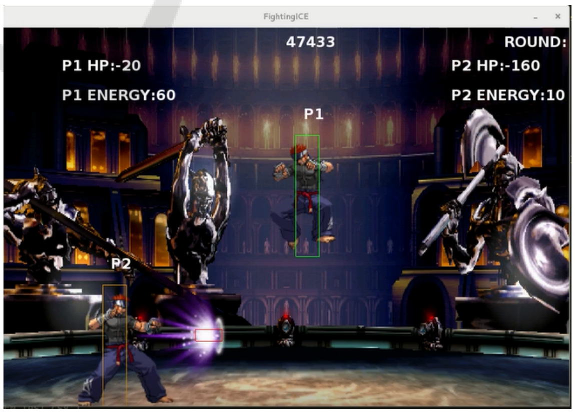

Figure 2: Player 2 (P2) is a rule-based AI, when all conditions are satisfied, it throw a energy ball to player 1 (P1).

Such combinations as the example in Figure 2 are very popular in rule-based AIs. If features have a strong relationship, we can make strong rules, otherwise, the rule becomes weaker. Imagine rules combined 'energy' and 'size of character', this strange combination would make the agent act clumsily.

We hypothesize that the connection between relative features in Fighting game is similar to the relation of neighboring pixels in an image. The nearer neighboring pixels have strong connections, while the further neighboring pixels have weaker or no connection to others. If grouping pixels help to improve the performance of modeling strong AI then grouping relative features could also work. Although usual neural network has the ability to combine these features if we use multiple layers, we can not control which feature would combine with others because all neurons in one layer are fully-connected with next layer. In those networks, strange connections like 'energy' and 'size of character' in the previous example are redundant.

By using CNN, we can reduce the numbers of ineffective connections, and put features that we need to a group. This idea can be implemented simply by:

- Represent input feature as a grid
- Find some important features and put them in a group by using a window
- When sliding the window, make sure the window always contain such important feature. This leads us to duplicate the important feature.

4 EXPERIMENTS

4.1 Dataset

We collect 560 games between top 3 players of Fighting AI Competition in 2015. Each game contains 3 rounds, each round last 60 seconds, and there are 60 frames per second. In other words, we have 560x3x60x60 = 6,048,000 pairs of state-action. We use 70% for training and 30% for validating. From FightingICE environment, we get information from our character and the opponent's character such as hitpoint, energy, the location of characters, size of characters, etc. Totally, we have 15 features from our character and 15 features from the opponent, then we compute 5 more important relative features such as distance, difference in hitpoint, difference in energy and 2 relative positions. Using this dataset, we try to model the next move of the top 3 strong AIs.

4.2 Experimental Setup and Results

Since FightingICE is written in Java, we have to write a short description in Java to get the dataset and save it in CSV format. After that, we build our neuron networks in Python with supported from two famous libraries: Numpy and Theano. We also use GPU GTX970 to run experiments.

In the first experiment, we use a usual neural network setting with 3 layers: one input layer, one hidden layer, and one output layer. The input layer has 35 neurons (because we have 35 features), hidden layer has 100 neurons and output layer has 56 neurons (because our agent can perform 56 actions). Training this network, we get a model which can archive an accuracy of 19.45% when predicting the next move. Tuning the number of neuron in hidden layer, we get the highest accuracy is 22.42%.

In the second experiment, we use a usual neural network setting with 4 layers: one input layer, two hidden layers, and one output layer. This setting based on the well-known fact that usual two hidden layer networks have higher expressiveness than one hidden layer. Training this network, we get a model which can archive an accuracy of 25.38% (with 1000 neurons per hidden layer).

In the third experiment, we use a naive Deep CNN setting: represent 35 features as a 5x7 grid input, one convolutional layer with 20 filters size 2x2 stride length 1, one fully-connected layer and one softmax layer (Figure 3). Training this network, we get a model which can archive an accuracy of 33.59%. It is reasonable because the location of features are select at random. Some features which are grouped together may have strong relationships. As a result, our model increased performance slightly compare with usual NN in Table 1

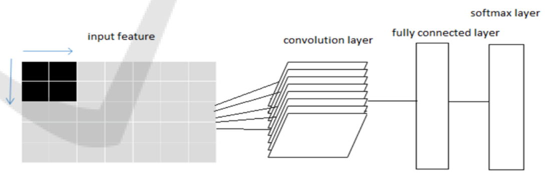

Figure 3: Naive CNN structure with 5x7 grid input, follow by one convolutional layer, one fully-connected layer and one softmax layer.

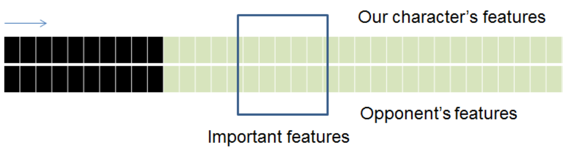

Figure 4: The structure of CNN in experiment 4. Information of our character and opponent's are separated: our information is at the top row, opponent's information is at the bottom row.

In the fourth experiment, we improve our model by separating information of our character and opponent's character (Figure 4). We duplicate all the

Table 1: Summary of our experimental results. The optimized CNN structure is clearly better than others and significantly better than a usual neural network.

	input	structure	training time (min)	accuracy
usual NN one-layer	35 features	1 fully connected-100 neurons	26.61	19.45%
	35 features	1 fully connected-200 neurons	30.91	18.80%
	35 features	1 fully connected-400 neurons	45.24	20.86%
	35 features	1 fully connected-1000 neurons	83.36	22.42%
usual NN two-layer	35 features	2 fully connected-100 neurons	36.92	20.97%
	35 features	2 fully connected-200 neurons	48.68	22.08%
	35 features	2 fully connected-400 neurons	51.32	23.38%
	35 features	2 fully connected-1000 neurons	167.84	25.38%
naive CNN	5x7 grid	5 filters size 2x2-1 fully connected	54.08	29.39%
	5x7 grid	10 filters size 2x2-1 fully connected	81.66	33.93%
	5x7 grid	20 filters size 2x2-1 fully connected	123.65	33.59%
CNN with 2x35 grid	2x35 grid	5 filters size 2x10-1 fully connected	43.62	33.70%
	2x35 grid	10 filters size 2x10-1 fully connected	54.76	38.32%
	2x35 grid	20 filters size 2x10-1 fully connected	77.12	37.98%
optimized CNN	4x15 grid	5 filters size 2x5-1 fully connected	77.10	49.20%
	4x15 grid	10 filters size 2x5-1 fully connected	116.16	54.14%
	4x15 grid	20 filters size 2x5-1 fully connected	182.22	54.24%
CNN with 3x5 filter	3x15 grid	5 filters size 3x5-1 fully connected	44.58	46.29%
	3x15 grid	10 filters size 3x5-1 fully connected	55.38	49.09%
	4x15 grid	5 filters size 3x5-1 fully connected	61.5	46.22%
	4x15 grid	10 filters size 3x5-1 fully connected	86.64	49.26%
	4x15 grid	20 filters size 3x5-1 fully connected	150.16	49.95%

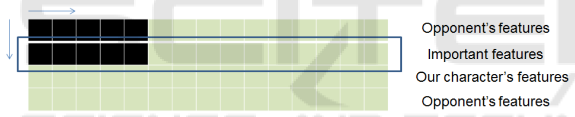

Figure 5: The input grid of the optimized CNN. Important features are duplicated 2 times and put between information of the 2 players. Opponent information is also duplicated and put at the bottom of the input grid.

features, so we have total 70 features and represent them as 2x35 grid input. The first row of the grid has 15 features from our character, 5 important features and 15 features from our character again, in this order. The second row has 15 features from opponent's character, 5 important features and 15 features from opponent's character again. We also use one convolutional layer with 20 filters size 2x10 and stride length 1, one fully-connected layer and one softmax layer. This setting lets the network compare the relationship between our information and opponent's information. It also shows that the location of features in the input grid has positive effects. When we sliding the window from left to right, the same features of both characters are accessible, allows the network to make comparisons between the states of the two characters. This structure improves the performance of our model to 37.98%.

In the fifth experiment, the input grid is expanded to preserve characteristic of the previous experiment while adding new potential combination. We duplicate 5 important features 2 times and the opponent's features 1 time. Totally, we have 60 features, which were represented as a 4x15 grid input. Information of our character and opponent's character are also separated in different row and the duplicated opponent's features are put at the bottom of the input grid, so that information of both characters can be combined similar to previous experiment (Figure 5). We use one convolutional layer with 20 filters size 2x5 and stride length 1, one fully-connected layer and one softmax layer. By using a 2x5 filter and duplicating the important features, it is possible for every filter in our CNN to have access to all 5 important features in any position in the input grid. This structure improves the performance of our model significantly to 54.24%.

In the last experiment, we try to confirm that if a bigger window can improve accuracy. First, we try a 3x15 grid input which contains the same information as the 4x15 grid input in previous experiment without the last duplicated row. Then, we paired it with a 3x5 filter which will capture information of our character, opponent and mutual information at the same time. However, the accuracy is reduced to 49.09%. Even when we duplicate the opponent's feature again, the result is only slightly improved to 49.95%. Table 1 summarizes the result of our experiments.

5 CONCLUDING REMARKS

In this paper, we have described a method to successfully incorporate CNN with optimized non-visual information. Before, CNN was mostly used with visual information such as images. Our method has shown that non-visual features can also be used effectively with CNN. By intentionally arrange features as an input grid, with the same information, CNN achieves 54.24% accuracy when predicting the next moves of AIs in the experiment. Meanwhile, the normal neural network can only reach 25.38% accuracy. With the promising result, we can expect CNN to be applied in even more type of problems where visual or similar information is not available.

Although in current experiments, we can model strong AI with high accuracy of 54.24% but it's not enough to win other strong AI. In future work, we will improve the network by a reinforcement learning method such as via Policy Gradient methods.

ACKNOWLEDGEMENTS

We thank to 'Research Grants for JAIST Students 2016-2017' for funding us in this work.

REFERENCES

Clark, C. and Storkey, A. (2014). Teaching deep convolutional neural networks to play go. *arXiv preprint arXiv:1412.3409*.

Karpathy, A. (2016). Deep reinforcement learning: Pong from pixels. Technical report.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lu, F., Yamamoto, K., Nomura, L. H., Mizuno, S., Lee, Y., and Thawonmas, R. (2013). Fighting game artificial intelligence competition platform. In *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, pages 320–323. IEEE.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

APPENDIX

In this section, we show more details of our experiments. Each character will be represented in our network by 15 features:

- Hitpoint
- Energy
- Character's size: width and height
- Character's hitbox coordinates: left, right, bottom and top
- Remaining frame of current action
- Type of current action
- Current speed of character: in x-axis and in y-axis
- Character's current facing direction
- Character's current state: can be controlled or not
- Information about projectiles in current state: quantity and relative position to the character (in front or behind)

The 5 important relative features:

- Distance between 2 characters
- Difference in hitpoint
- Difference in energy
- Difference of position in x-axis
- Difference of position in y-axis