Mengkombinasikan Metode Q-Learning dan Backpropagation dalam Permainan Flappy Bird

Resnia Trya Muslima
Fakultas Sains dan Teknologi
Universitas Al Azhar Indonesia
Jakarta, Indonesia
resniatrya07@gmail.com

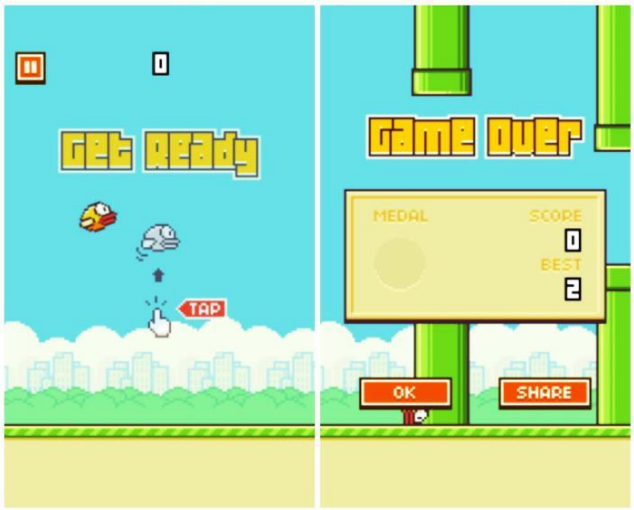
***Abstract*—Makalah ini memiliki tujuan untuk mengkombinasikan *Algoritma Q-Learning* dan *Backpropagation* dalam kasus agen learning dalam *Game Flappy Bird*. Untuk memprediksi fungsi nilai dari setiap tindakan maka dilakukanlah kombinasi antara *Algoritma Q-Learning* dan *Backpropagation*. Pendekatan fungsi nilai ini digunakan untuk mengurangi waktu belajar dan mengurangi bobot yang tersimpan dalam memori. Maka dari itu, hasilnya menunjukkan bahwa dengan mengkombinasikan *Algoritma Q-Learning* dan *Backpropagation* dapat mengurangi waktu belajar agen untuk memainkan *Game Flappy Bird* jika dibandingkan dengan penggunaan *Algoritma Q-Learning* regular saja yang menyimpan dalam memori.**

Kata Kunci—Game; Flappy Bird; Algortima; Q-Learning; Backpropagation;

I. PENDAHULUAN

Permainan (*game*) adalah merupakan salah satu implementasi dalam bidang ilmu komputer. Perkembangan permainan yang sangat pesat telah menjadi mode tersendiri di dunia, sehingga permainan sudah dapat dimainkan di*gadget*, salah satunya adalah *Flappy Bird*. *Flappy Bird* adalah permainan yang mengharuskan pemainnya selalu mengontrol dengan melihat ketinggian seekor burung dengan mengetuk layar agar dapat melewati celah diantara dua pipa sebanyak mungkin yang selalu berdatangan. Seekor burung akan mati jika menabrak dan jatuh ke bawah tanah karena adanya gaya gravitasi dengan ini maka permainan telah berakhir.

Permasalahan sering terjadi di awal, karena ada dua skenario yang berbeda. Skenario pertama adalah *Flappy Bird* standar, skenario yang kedua, untuk meningkatkan kesulitan permainan dengan cara pipa berubah untuk bisa bergerak ke atas dan ke bawah secara spesifik kecepatan yang menyebabkan kebanyakan agen berakhir di awal. Dari hal ini mengharuskan agen untuk mempelajari gerakan yang kompleks dan membutuhkan waktu lebih lama daripada mempelajari bagaimana cara lolos dari pipa berikutnya. Pict.1 Tampilan Utama dan *Game Over* pada permainan *Flappy Bird*



Pict.1 Tampilan Utama dan Game Over Game Flappy Bird
Sumber : http://mashable.com/2014/02/04/flappy-bird-developer/#P_BRJDATHsqa

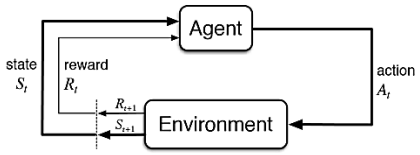
Ditinjau dari permasalahan tersebut, sudah ada penelitian yang menggunakan metode *Q-learning*. Sebagai salah satu metode dalam Reinforcement Learning yang menghasilkan rancangan keadaan (*state*) yang lebih efisien. Namun, pembelajaran dengan *Q-learning* masih membutuhkan waktu pembelajaran yang lama dan jumlah memori yang besar untuk menyimpan bobot. Maka metode *Q-learning* akan dikombinasikan dengan *Backpropagation*, untuk mendapatkan hasil yang lebih efisien dan efektif.

II. LANDASAN TEORI

A. *Reinforcement Learning*

Reinforcement learning adalah suatu metode pembelajaran untuk memetakan setiap *state* terhadap *action* yang dipilih untuk memaksimalkan *reward* yang diterima. Setiap *state* dan *action* yang dipetakan diberi nilai yang representasikan sebagai sebuah table. Suatu agen dituntut untuk menemukan sendiri *action* apa yang menghasilkan *reward* paling besar dengan cara mencobanya. *Agent* (agen) adalah program yang berjalan pada prosesor (cluster prosesor) yang

mengimplementasikan algoritma *reinforcement learning*. *Enviroment* (lingkungan) adalah representasi digital dari dunia untuk tempat agen beroperasi.



Pict.2 interaksi antara agen dan lingkungan
Sumber : https://stackoverflow.com/questions/47047250/tensorflow-reinforcement-learning-with-variable-character-level-text-input

Karakteristik lain dari *reinforcement learning* adalah mempertimbangkan masalah lalu mengarahkan pada tujuan saat berinteraksi dengan lingkungan yang tidak pasti. Agen menerima *state* (representasi dari *environment*) dan memilih *action*. Agen akan selalu berusaha untuk memaksimalkan *reward* yang diterima dari waktu ke waktu.

B. Q-Learning

Q-learning adalah pengembangan dari Temporal Difference yang juga biasa dikenal *off-policy TD control*. *Q-learning* merupakan salah satu terobosan paling penting dalam *Reinforcement Learning*. Tidak seperti *TD-learning* yang memperbaharui *value function* berdasarkan keadaan selanjutnya, Q-learning memperbaharui *value function* berdasarkan nilai *action-value function* terbesar di keadaan selanjutnya.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

Pict.3 Q-learning melakukan update terhadap action-value function
Sumber : https://randomant.net/reinforcement-learning-concepts/

Daftar Notasi

α = Learning rate, $0 < \alpha \leq 1$, menentukan ukuran laju dimana nilai yang lama akan digantikan dengan nilai baru.

γ = Discount rate, $0 \leq \gamma \leq 1$, menentukan nilai dari reward di masa depan, semakin kecil nilai γ, maka agent akan semakin mementingkan reward dekat, bukan reward di masa depan.

$Q(S_t, A_t)$ adalah *action-value function* pada keadaan ke -*t* (S_t) dan aksi ke -*t* (A_t). R_{t+1} adalah *reward* yang diperoleh pada waktu ke -*t*+1, sedangkan $\max_\alpha Q(S_{t+1}, \alpha)$ adalah nilai untuk suatu aksi α.

C. Algoritma Q-Learning

Pada proses pembelajaran metode Algoritma *Q-learning* diawali dengan menginisialisasi nilai *action-value function Q(S,A)* dan proses perulangan pemilihan aksi A serta nilai *action-value function* yang diperbaharui sampai kondisi pembelajaran yang digunakan sudah terpenuhi. Seperti gambar Pict.4 telah ditunjukkan algoritma *Q-Learning* secara lengkap dalam bentuk *procedural*.

Algoritma Q-Learning

Inisialisasi $Q(S,A)$
Untuk setiap peristiwa dilakukan perulangan:
  Menginisialisasi S (dari *environment*)
  Untuk setiap langkah dilakukan perulangan:
    Dipilih *action A* pada *state S* (contohnya dengan ε-*greedy*)
    Dilakukan *action A*, lalu didapatkan *reward R* dan *next state S'*
    Action-value function diperbaharui pada *state S* dan *action A*
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
    Diperbaharui *next state* menjadi *current state* $S \leftarrow S'$
  Sampai S adalah akhir dari peristiwa
Sampai proses pembelajaran selesai

Pict.4 Algoritma *Q-Learning*

D. Backpropogation

Backpropagation adalah salah satu algoritma *supervised learning* yang digunakan dalam *Artificial Neural Networks*. *Supervised learning* adalah metode pembelajaran yang melibatkan prediksi *output* dari *input* baru yang diberikan. *Backpropagation* mencari kombinasi bobot untuk meminimalkan kesalahan output untuk dianggap menjadi solusi yang benar. Dalam hal ini, *Backpropagation* memiliki dua tahap, sebagai berikut:

1. *Feed-forward*, yaitu suatu proses pelatihan suatu pola yang akan disatukan ke setiap unit di input layer, lalu keluaran yang dihasilkan akan ditransmisikan ke lapisan selanjutnya, dan terus sampai output layer.

2. *Backpropagation*, yaitu suatu proses penyesuaian pada setiap bobot berdasarkan keluaran yang diharapkan, agar menghasilkan kemungkinan terjadinya galat (eror) sangat kecil, mulai dari bobot yang terhubung ke neuron keluaran, dan terus mundur sampai ke input layer.

E. Algoritma Backpropogation

Algoritma Backpropagation diawali dengan inisialisasi bobot w1 dan w2 untuk setiap pola dan dilakukan proses perulangan *feed forward* dan *backpropagation* sampai kondisi saat berhenti telah dipenuhi. Gbr.5 Alur atau Algoritma Backpropagation yang diawali dengan w1 dan w2. Nilai $w1_{(i,j)}$ adalah nilai bobot *w*1 yang dapat menghubungkan neuron $x_{(i)}$ menuju neuron $y_{(j)}$. Sama seperti $w1_{(i,j)}$, $w2_{(j,k)}$ memiliki bobot yang menghubungkan neuron $y_{(j)}$ menuju neuron $z_{(k)}$. Parameter $t_{(k)}$ adalah target nilai dari neuron *k* untuk suatu pola. Paramaeter α adalah nilai dari learning rate, sedangkan $\Delta w1_{(i,j)}$ adalah besarnya perubahan suatu bobot *w*1 dari neuron $x_{(i)}$ menuju neuron $y_{(j)}$ dan $\Delta w2_{(j,k)}$ adalah besarnya perubahan bobot *w*2 yang menghubungkan neuron $y_{(j)}$ menuju neuron $z_{(k)}$. Dari Gbr.5 dapat terlihat bahwa nilai $y_{(j)}$ dan $z_{(k)}$ dihasilkan dari sebuah fungsi yaitu *activation function f(x)*.

Algoritma *Backpropagation*

Dilakukan proses inisialisasi semua bobot w1 dan w2 dengan nilai kecil secara berulang:

Diulangi untuk setiap pola:
Atur pola ke input unit x
Feed-forward
Dihitung output y dari setiap neuron di hidden layer

$$y\_net_{(j)} = \sum_{i=1}^{a+1} w1_{(i,j)} x_{(i)}$$

$$y_{(j)} = f(y\_net_{(j)})$$

Dihitung output dari setiap neuron di output layer

$$z\_net_{(k)} = \sum_{j=1}^{b+1} w2_{(j,k)} y_{(j)}$$

$$z_{(k)} = f(z\_net_{(k)})$$

Backpropagation
Dihitung perhitungan semua bobot w2

$$\delta\_o_{(k)} = (t_{(k)} - z_{(k)}) f'(z\_net_{(k)})$$

$$\Delta w2_{(j,k)} = \alpha . \delta\_o_{(k)} . y_{(j)}$$

Dihitung perubahan semua bobot w1

$$\delta\_h\_net_{(j)} = \sum_{k=1}^{c} w2_{(j,k)} \delta\_o_{(k)}$$

$$\delta\_h_{(j)} = \delta\_h\_net_{(j)} f'(y\_net_{(j)})$$

$$\Delta w1_{(i,j)} = \alpha \delta\_h_{(j)} x_{(i)}$$

Dilakukan perubahan nilai bobot w1 dan w2

$$w2_{(j,k)} = w2_{(j,k)} + \Delta w2_{(j,k)}$$

$$w1_{(i,j)} = w1_{(i,j)} + \Delta w1_{(i,j)}$$

Sampai kondisi henti terpenuhi

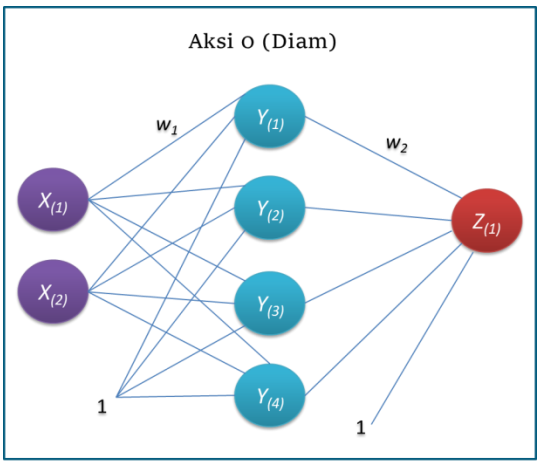
Gbr.5 Algoritma *Backpropagation*

III. KOMBINASI Q-LEARNING DENGAN BACKPROPAGATION

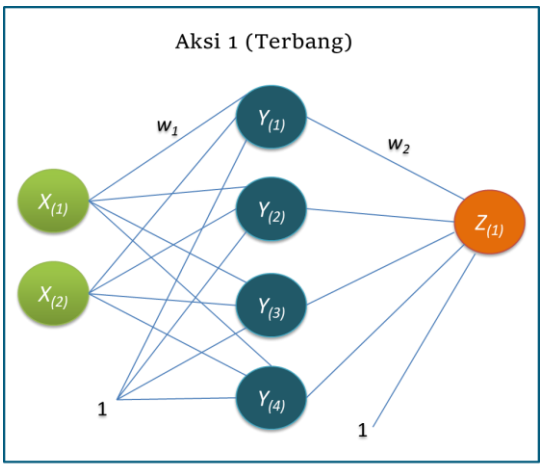
Game Flappy Bird memiliki ukuran keadaan yang besar. Dapat terlihat dari posisi burung dan posisi pipa yang selalu berubah setiap waktu. Perubahan dari beberapa *pixel* sangat mempengaruhi keadaan tersebut. Dengan begitu, makalah ini akan mengkaji seberapa besar pengaruh penggunaan *backpropagation* sebagai *value function approximation* pada proses pembelajaran agen flappy bird yang belajar untuk melewati pipa dengan menggunakan algoritma *Q-learning*. Berikut adalah penjelasan dari tahap kombinasi *Q-Learning* dan *Backpropagation*.

1) *State (Keadaan):* Keadaan dalam permainan ini adalah
   - $S_{(1)}$ = Selisih antara burung dan pipa pada bagian bawah di depannya. Kondisi ini didapat dari kordinat *y* pipa bagian bawah yang ada di depannya dikurang kordinat *y* burung lalu dibagi dengan 100.
   - $S_{(2)}$ = Jarak antara burung dan pipa di depannya. Jarak didapatkan dari nilai kordinat *x* pipa di depannya dikurangi kordinat *x* burung lalu dibagi dengan 100.

   Kedua nilai dibagi dengan 100 sebelum dijadikan masukan ke *ANN* agar perubahan bobotnya tidak terlalu jauh. Nilai 100 dipilih karena ukuran *frame* bernilai ratusan *pixel* sehingga pada akhirnya akan menjadi nilai satuan.
2) *Aksi:* Dalam permainan terdapat dua aksi yaitu saat burung terbang dan diam. Nilai aksi didefinisikan sebagai nilai biner yaitu, terbang = 1 dan diam = 0.
3) *Reward (Kejadian):* Kejadian yang terdapat dalam permainan ini ada tiga kejadian yaitu (-1) = ketika burung mati, (+0.1) = ketika burung hidup, dan (+1) = ketika *score* (nilai) bertambah. Penentuan reward ini dipengaruhi dengan *value action function* yang digunakan. Dengan itu digunakan fungsi *sigmoid bipolar :*
   - $f(x) = (2/(1+e^{-x}))-1$
   - $f'(x) = ½ .(1+f(x))(1-f(x))$

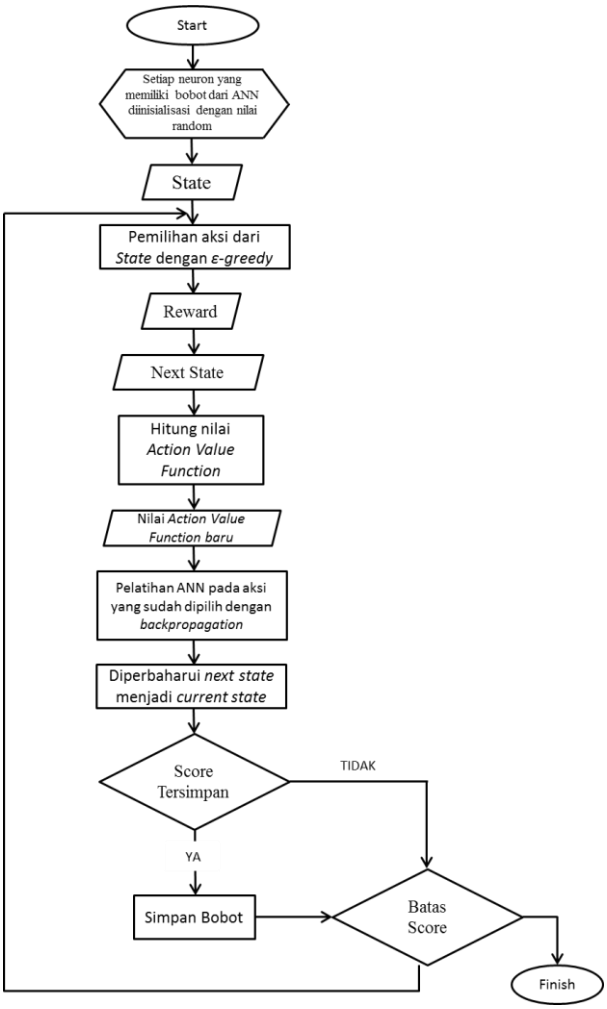


Pict.6 Artificial Neural Network (ANN) Aksi Diam



Pict.7 Artificial Neural Network (ANN) Aksi Terbang

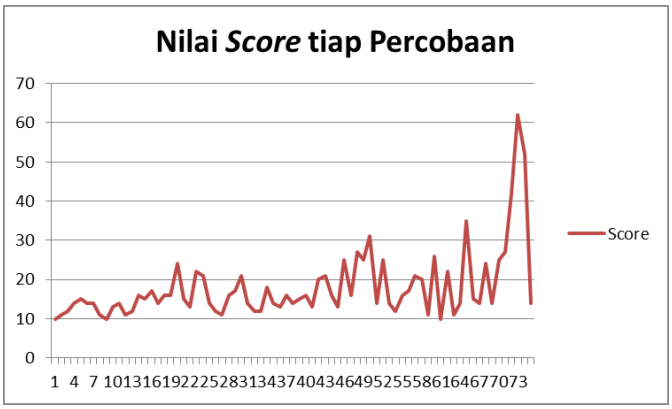
Dari Pict.6 dan Pict.7 dapat dilihat pada gambar untuk satu ANN tiap kemungkinan aksi. Dari kedua ANN memiliki masing – masing memiliki dua masukan unit (input) dari setiap keadaan, memiliki empat neuron tersembunyi (hidden layers), dan memiliki satu neuron keluaran (output) sebagai nilai dari *action value function*.Berikut adalah proses dari kombinasi *Q-Learning* dan *Backpropagation*



Gbr.8 Proses Pembelajaran agen dengan mengkombinasi *Q-Learning* dan *Backpropagation*

Berikut adalah detail penjelasan dari proses pembelajaran *Q-learning* yang dikombinasikan dengan *backpropagation*:

- Pada proses awal agen menginisialisasi bobot dari masing – masing ANN dengan mengambil nilai acak antara 0 sampai 1.
- Setelah agen mendapatkan nilai acak, dapat dilanjutkan dengan melihat keadaan agen tersebut. Keadaan yang didapatkan akan menunjukkan posisi vertikal dan horizontal dari agen.
- Langkah selanjutnya adalah agen dapat memilih aksi yang akan dilakukan dengan ε-*greedy*. Nilai ε yang digunakan adalah 0.001, yang berarti bahwa agen akan selalu memilih aksi dengan nilai *action-value function* terbesar. Pemilihan nilai tersebut digunakan karena pada kasus ini aksi akan diminta sangat cepat tiap *frame*-nya. Jika ada pemilihan aksi secara acak (*random action*) dan aksi yang diambil salah, maka akan sangat mengganggu proses pembelajaran. Agen akan sulit mencari solusi yang optimal.
- Agen akan mendapat *reward* berupa poin untuk setiap kondisi pipa yang dilewati. Setelah poin bertambah, dilanjutkan dengan keadaan berikutnya (*next state*) dari aksi yang sudah dilakukan.
- Setelah dilanjutkan dengan *next state*, agen harus menghitung nilai *update action-value function* dengan menggunakan rumus pada Gbr.3 yang nantinya digunakan sebagai keluaran yang diharapkan pada proses *backpropagation*. *Learning rate* yang digunakan adalah 0.9. Nilai ini dipilih karena diasumsikan dibutuhkan perubahan yang besar pada setiap nilai untuk mencapai solusi optimal. Nilai *discount rate* yang digunakan adalah 0.9, bertujuan supaya agen lebih mementingkan *reward* di masa depan daripada *reward* yang dekat.
- Digunakan ANN *backpropagation* untuk memilih aksi yang akan digunakan yaitu terbang atau diam. Adapun keluaran yang diharapkan diperoleh dari nilai *update action-value function*. Untuk nilai *learning rate* masih digunakan 0.9.
- Setelah dipilih aksi dari agen, selanjutnya agen akan melakukan *update next state*. Adapun *update next state* yang didapat akan dijadikan sebagai *current state*.
- Terakhir adalah pengecekan kondisi penyimpanan bobot dan kondisi henti. Kondisi henti diambil berdasarkan *score* yang didapat. Ketika sudah mencapai batas *score*, maka proses pembelajaran akan dihentikan. Namun, sebelumnnya bobot sudah mencapai *score* tersimpan terlebih dahulu dan nantinya akan digunakan setelah proses pembelajaran selesai. Jika *score* tidak mencapai batas, maka akan kembali ke langkah 3.



Gbr.9 Hasil score dari setiap percobaan

Jika proses pembelajaran selesai, agen akan menggunakan bobot yang telah disimpan untuk proses pengujian. Agen tidak akan lagi memilih aksi dengan ε-*greedy*, tetapi akan selalu memilih aksi dengan nilai *action-value function* terbesar. Agen juga tidak lagi menghitung nilai *update* untuk *action-value function* dan tidak lagi melatih ANN.

Adapun waktu pembelajaran dari *Q-learning* yang dikombinasikan dengan *backpropagation* disajikan pada Gbr.10

Percobaan ke	Waktu (s)
1	441
2	484
3	542
4	332
5	674
6	612
7	541
8	442
9	344
10	678
Rata - rata	509

IV. KESIMPULAN

Berdasarkan proses pembelajaran ini dapat diambil kesimpulan bahwa mengkombinasi metode *Q-Learning* dan *Backpropagation* dapat membuat waktu pembelajaran agen untuk memainkan Flappy Bird dapat lebih cepat dan dapat mengurangi bobot yang disimpan di memori, jika dibandingkan dengan menggunakan *Q-Learning* saja. Walaupun dengan waktu pembelajaran yang lebih cepat dan bobot yang disimpan berkurang, tetapi *Q-learning* yang dikombinasikan dengan *backpropagation* memiliki kemampuan yang sama dengan *Q-learning* saja untuk memainkan permainan Flappy Bird. Dalam implementasinya, dibutuhkan proses inisialisasi ulang bobot ANN ketika proses pembelajaran tidak berjalan dengan baik.

ACTKOWLEDGMENT

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas selesainya makalah yang berjudul " Mengkombinasikan Metode *Q-Learning* dan *Backpropagation* dalam Permainan Flappy Bird". Atas dukungan moral dan materil yang diberikan dalam penyusunan makalah ini, maka penulis mengucapkan banyak terima kasih kepada Bapak Ali Akbar dan Ibu Winangsari, selaku dosen mata kuliah Kecerdasan Buatan yang telah memberikan bimbingan, saran, ide dan kesempatan untuk dapat menyelesaikan makalah ini. Terlepas dari semua itu, penulis menyadari sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya. Oleh karena itu dengan tangan terbuka penulis menerima segala saran dan kritik dari pembaca agar dapat memperbaiki makalah ini. Akhir kata penulis berharap semoga makalah ini dapat bermanfaat dan dapat menginspirasi mahasiswa dan mahasiswi untuk terus bersemangat dalam melakukan penelitian terhadap metode Q-Learning dan Backpropagation.

REFERENCES

[1] Ardiansyah and E. Rainarli. Implementasi q-learning dan backpropagation pada agen yang memainkan permainan flappy bird. URL https://ejnteti.jteti.ugm.ac.id/ index.php/JNTETI/article/view/287.

[2] Z. H. Kamil Bojanczyk and C. Xu. Final report fly, flappy, fly: A q-learning decision system for obstacle detection and avoidance. URL https://web.stanford.edu/class/cs221/2017/restricted/pfinal/kamilb/final.pdf.

[3] Y. Lin. Using deep q-network to learn how to play flappy bird. URL https://github.com/yenchenlin/DeepLearningFlappyBird.

[4] C. Pearce. Computer science and game development. URL https://camd.northeastern.edu/gamedesign/academic-programs/bs-in-computer-science-and-game-design/.

[5] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. 1998.

[6] M. Y. Yi Shu, Ludong Sun and Z. Zhu. Obstacles avoidance with machine learning con-trol methods in flappy birds setting. URL http://cs229.stanford.edu/projects2014. html.

[7] J. Bennett. The algorithm behind the curtain: Reinforcement learning concepts (2 of 5). URL https://randomant.net/reinforcement-learning-concepts/.