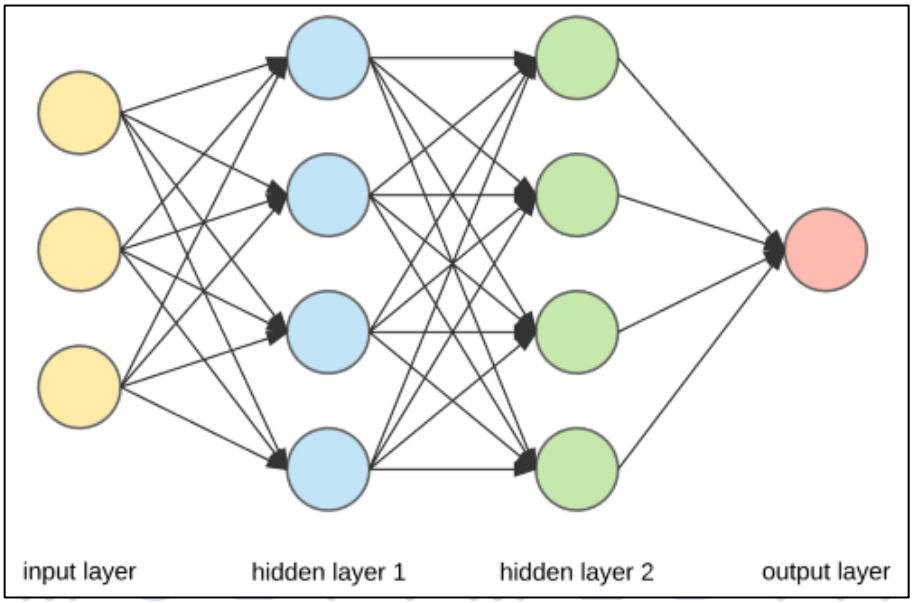
BAB 2
LANDASAN TEORI

2.1 Neural Network

Neural Network merupakan sebuah algoritma series yang memiliki kemampuan untuk mengenali hubungan di dalam sebuah data melalui sebuah proses yang bekerja hampir sama dengan bagaimana otak manusia bekerja, di dalam hal ini neural Network dapat dikatakan sebagai sistem neuron maupun organik dan *artificial*. Neural Network dapat beradaptasi untuk mengubah *input* yang akan diterima sehingga jaringan yang bekerja di dalamnya dapat membuat hasil yang sangat bagus tanpa harus merangkai ulang kriteria *output* yang ada. Sebuah *neural network* memiliki layer pada sebuah *node* yang saling berhubungan satu sama lain, setiap *node* memiliki persepsi yang mana hal ini mirip dengan *multiple linear regression*. Persepsi yang ada di dalam node tersebut akan memberikan sebuah sinyal yang telah dibuat oleh *multiple linear regression* ke dalam fungsi aktif yang non-linier [18]. Arsitektur Neural Network dapat dijabarkan sebagai berikut.



Gambar 2.1 Arsitektur neural network

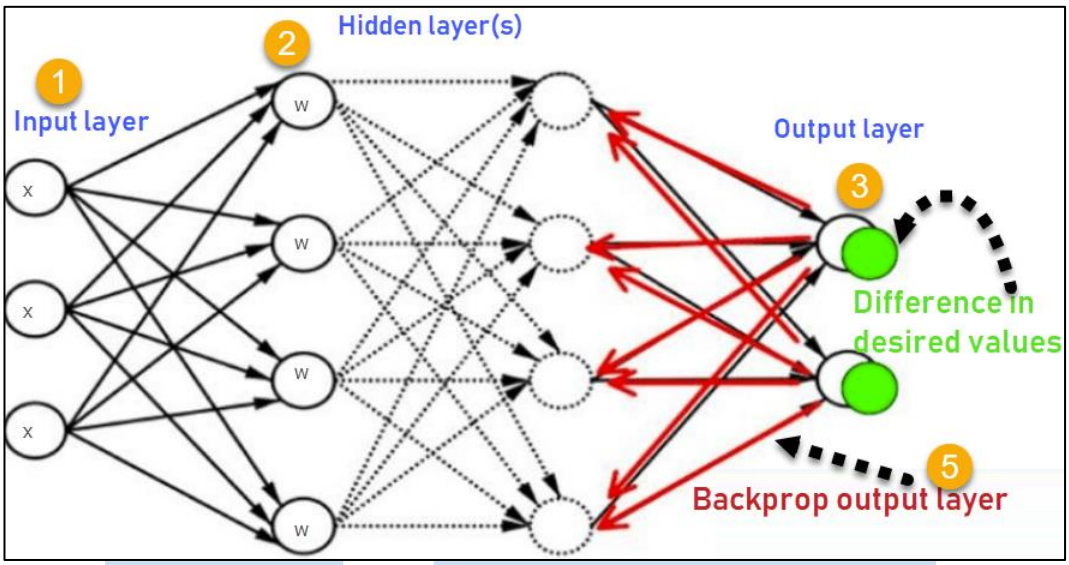
Sumber: [19]

Pada Gambar 2.1, menunjukkan arsitektur dari neural network yang terdiri dari 3 tipe lapisan yaitu *input layer*, *hidden layer*, dan *output layer*. Di mana *input layer* merupakan data inisial untuk *neural network*, kemudian *hidden layer* merupakan sebuah *layer* yang berada diposisi tengah di antara *input layer* dan *output layer* serta merupakan sebuah tempat di mana semua komputasi dijalankan, dan *output layer* merupakan hasil dari *input* yang diberikan [19].

2.2 Backpropagation

Backpropagation merupakan sebuah algoritma yang melakukan komputasi untuk *chain-rule* dengan urutan atau struktur operasi yang sangat spesifik dan sangat efisien. Algoritma ini membantu pengguna untuk melakukan komputasi bagian derivatif $\partial C/\partial w_i$ di saat yang bersamaan, algoritma ini juga menggunakan *one forward pass* didalam sebuah jaringan yang disertai oleh *one forward pass* lainnya. Maka dari itu, *cost* yang digunakan oleh *backpropagation* hampir sama dengan membuat *two forward passes* yang akan digunakan di sebuah jaringan [20]. Proses algoritma ini akan melakukan penyesuaian kembali setiap bobot serta bias melalui *error* yang telah didapatkan ketika melakukan *forward pass* [21].

Backpropagation bertujuan untuk memperkecil *cost function* dengan menyesuaikan bobot dan bias jaringan. Tingkatan dari penyesuaian tersebut ditentukan oleh gradien dari *cost function* dengan parameter yang sudah ada [22]. Hal tersebut membuat waktu komputasi untuk satu lapisan menjadi lebih efisien, berbeda dengan komputasi *native direct* yang melakukan komputasi gradien namun tidak menjelaskan bagaimana gradien tersebut digunakan melainkan menyamaratakan komputasi yang ada di dalam *delta rule* [23]. Kemudian, diagram dari *backpropagation* adalah sebagai berikut.



Gambar 2.2. Diagram backpropagation

Sumber: [23]

Penjelasan pada Gambar 2.2:

a. *Input X*, telah sampai melalui jalur yang sudah terhubung sebelumnya.
b. Model dari *input* tersebut menggunakan bobot asli dari nilai W. Bobot tersebut pada umumnya dipilih secara acak.
c. Kemudian akan dilakukan kalkulasi untuk *output* untuk setiap *neuron* dari *input layer* lalu menuju *hidden layer* dan menuju *output layer*.
d. Setelah itu perhitungan *error* akan dilakukan pada *output*.
e. Kemudian akan melakukan pengulangan kembali dari *output layer* ke *hidden layer* untuk menyesuaikan bobot sampai *error* berkurang.

Backpropagation juga memiliki beberapa kelebihan antara lain adalah mudah digunakan, cepat dan *simple*. Kemudian *backpropagation* tidak memiliki parameter untuk digunakan selain dari jumlah *input* yang masuk, juga merupakan metode yang cukup fleksibel dan standar yang dapat bekerja dengan baik. Lalu, *backpropagation* tidak memerlukan fitur spesial dari *function* untuk digunakan [23].

2.3 Recurrent Neural Network

Recurrent Neural Networks merupakan sebuah model yang sangat ekspresif untuk melakukan sebuah tugas yang sekuensial, model ini sangat efisien karena

tingginya sebuah dimensi tersembunyi yang dimilikinya bersamaan dengan dinamika non-linier yang dapat membantu model tersebut untuk mengingat dan mengelola informasi yang ada sebelumnya. Selain itu, RNN memiliki gradien yang cukup mudah untuk di komputasi menggunakan *backpropagation* seiring waktu berjalan, meskipun kualitas dari model ini sangat bagus namun model RNN ini gagal untuk menjadi sebuah alat *machine learning* utama dikarenakan kerumitan saat melakukan *training* secara efektif. Oleh karena itu, hubungan antara parameter dengan dinamika yang tersembunyi tidaklah stabil sehingga model ini tampak seperti model yang bermasalah me gradien yang dimilikinya [24].

Recurrent Neural Networks dapat didefinisikan sebagai tipe *neural network* yang sangat kuat dan kokoh serta merupakan algoritma yang sangat efisien di dalam penggunaannya, algoritma ini dapat dikatakan cukup lama atau kuno dibandingkan dengan algoritma *deep learning* lainnya. Algoritma ini memiliki ingatan internal yang memungkinkan untuk mengingat hal penting berdasarkan *input* yang diterima, hal ini membuat algoritma ini untuk memprediksi sangat tepat dengan apa yang akan terjadi untuk masa yang akan datang [25].

2.4 Simple Recurrent Network (SRN)

Simple recurrent network merupakan sebuah jaringan dengan *input* yang merepresentasikan sebuah informasi yang terdapat pada di dalam *environment*, sedangkan *output* yang dikeluarkan adalah prediksi dari model tentang informasi yang akan datang sebagai jawaban yang terungkap seiringnya waktu berjalan [26]. Model dari SRN dapat digambarkan sebagai perhitungan sebagai berikut [27]:

$$\underline{C}_t = W_c \underline{C}_{t-1} + W_{in} \underline{X}_t \, , \tag{2.1}$$

$$\underline{h}_t = f(\underline{C}_t), \tag{2.2}$$

Di mana subskrip t pada rumus 2.1 dan 2.2 merupakan unit dari indeks waktu, $\boldsymbol{W_c} \in \mathbb{R}^{N \times N}$ merupakan bobot matriks untuk vector *hidden state* yaitu $\underline{\boldsymbol{C}}_{\boldsymbol{t-1}} \in \mathbb{R}^{N}$, sedangkan $\boldsymbol{W_{in}} \in \mathbb{R}^{N \times M}$ adalah bobot matriks dari *input vector* yaitu $\underline{\boldsymbol{X}}_{\boldsymbol{t}} \in \mathbb{R}^{M}$, $\underline{\boldsymbol{h}}_{\boldsymbol{t}} \in \mathbb{R}^{N}$ yang terdapat didalam *output vector*, dan $f(\cdot)$ adalah *hyperbolic-tangent* atau fungsi *sigmoid*. Secara induksi $\underline{C}_t$ dapat ditulis menjadi

$$\underline{c}_t = W_c^t \underline{c}_0 + \sum_{k=1}^{t} W_c^{t-k} W_{in} \underline{X}_k \quad (2.3)$$

Yang di mana $\underline{\boldsymbol{c}}_{\boldsymbol{0}}$ adalah inisial dari *internal state* SRN. Secara ilmiah, $\underline{\boldsymbol{c}}_{\boldsymbol{0}}$ dapat di samakan dengan 0 (nol). Sehingga, perhitungan 2.3 menjadi seperti pada rumus 2.4

$$\underline{c}_t = \sum_{k=1}^{t} W_c^{t-k} W_{in} \underline{X}_k \quad (2.4)$$

Pada perhitungan (4), *output* yang dikeluarkan oleh SRN adalah sebuah fungsi yang meneruskan semua elemen di dalam *input sequence*. Ketergantungan yang terjadi antara *output* dan *input* memberikan SRN untuk dapat mempertahankan urutan pola yang semantik yang diterima dari *input*. Meskipun demikian, SRN merupakan sebuah sistem dengan memori yang memiliki kapasitas terbatas. Dapat dimisalkan $\gtrdot_{max}$ menjadi *value singular* terbesar dari $\boldsymbol{W_c}$. Maka akan mendapatkan seperti pada 2.5

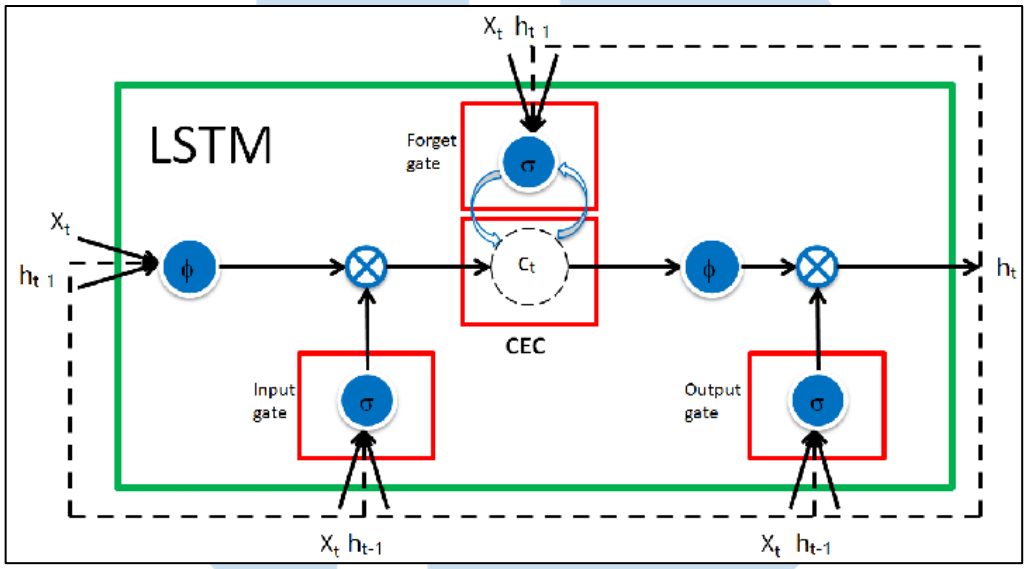
$$\left|W_c^{t-k} W_{in} \underline{X}_k\right| \leq \|W_c\|^{t-k} \left|W_{in} \underline{X}_k\right| = \sigma_{max}(W_c)^{t-k} \left|W_{in} \underline{X}_k\right|, k \leq t\,. \quad (2.5)$$

Di mana $\|\cdot\|$ menandakan norma matriks dan $|\cdot|$ menandakan norma vektor, kedua hal tersebut merupakan norma dari sebuah matriks. Kemudian $\sigma_{max}(\cdot)$ menandakan value *singular* terbesar, di mana pertidaksamaan tersebut diturunkan oleh norma matriks. Sehingga persamaan dapat diturunkan dengan fakta dari norma spektral (l^2 norma dari sebuah matriks) dari matriks persegi yang memiliki sama besar dengan *singular value.*

2.5 Long Short Term Memory (LSTM)

Long Short-Term Memory merupakan sebuah jaringan yang memiliki *extension* yang akan digunakan oleh *recurrent nerural network*, hal ini dapat dikatakan bahwa LSTM akan memperluas ingatan yang dimiliki oleh *recurrent neural network*. LSTM membantu *recurrent neural network* untuk mengingat input yang telah diterima selama beberapa periode yang cukup lama. Hal ini disebabkan oleh LSTM yang memiliki informasi di dalam ingatannya sama seperti memori

pada sebuah komputer [24]. Long Short-Term Memory juga dapat diartikan sebagai tipe dari *recurrent neural network* yang dapat mempelajari masalah prediksi secara sekuensial dan terstruktur. LSTM juga merupakan bagian kompleks area pada *deep learning* [28].



Gambar 2.3. Diagram sel LSTM

Sumber: [28]

Pada Gambar 2.3 menjelaskan bahwa φ, σ dan ⊗ menunjukkan fungsi tangen *hyperbolic*, fungsi *sigmoid* yang dapat membedakan operasi *singular value* sebagai σmax atau σmin dan fungsi *multiplication*. Sel pada LSTM memiliki modul *input gate, output gate, forget gate,* dan *constant error carousal* (CEC). Maka secara matematika LSTM dapat ditulis sebagai berikut.

$$\underline{C}_t = \sigma(W_f \underline{I}_t)\underline{c}_{t-1} + \sigma(W_i \underline{I}_t)\phi(W_{in} \underline{I}_t), \quad (2.6)$$

$$\underline{h}_t = \sigma(W_o \underline{I}_t)\phi(\underline{c}_t), \quad (2.7)$$

Yang di mana $\underline{\boldsymbol{c}}_{\boldsymbol{t}} \in \mathbb{R}^{\boldsymbol{N}}$ pada kolom vektor $\underline{\boldsymbol{I}}_{\boldsymbol{t}} \in \mathbb{R}^{(\boldsymbol{M+N})}$ merupakan rangkaian dari input yang baru yaitu $\underline{\boldsymbol{X}}_{\boldsymbol{t}} \in \mathbb{R}^{\boldsymbol{M}}$, dan output sebelumnya yaitu $\underline{\boldsymbol{h}}_{\boldsymbol{t-1}} \in \mathbb{R}^{\boldsymbol{N}}$ yang juga dapat dituliskan dalam bentuk $\underline{\boldsymbol{I}}_{\boldsymbol{t}}^{\boldsymbol{T}} = [\underline{\boldsymbol{X}}_{\boldsymbol{t}}^{\boldsymbol{T}}, \underline{\boldsymbol{h}}_{\boldsymbol{t-1}}^{\boldsymbol{T}}]$. Setelah itu, $\boldsymbol{W_f}$, $\boldsymbol{W_i}$, $\boldsymbol{W_o}$, dan $\boldsymbol{W_{in}}$ merupakan bobot dari matriks untuk *forget target, input gate,* dan *output gate* serta *input* yang akan dimasukkan. Dapat diasumsikan $\underline{\boldsymbol{c}}_{\boldsymbol{0}} = \underline{\boldsymbol{0}}$ adalah vektor *hidden state* dari LSTM yang dapat di turunkan menjadi

$$\underline{c_t} = \sum_{k=1}^{t} \underbrace{\left[\prod_{j=k+1}^{t} \sigma(W_f \underline{I}_j)\right]}_{\textit{forget gate}} \sigma(W_i \underline{I}_k)\phi(W_{in}\underline{I}_k) \tag{2.8}$$

Dengan menggunakan $\boldsymbol{f}(\cdot)$ pada perhitungan (2.8) sebagai fungsi *hyperbolic-tangent*, maka dapat dibandingkan *output* dari SRN dan LSTM sebagai berikut:

$$\underline{h}_t^{SRN} = \phi\left(\sum_{k=1}^{t} W_c^{t-k} W_{in} \underline{X}_k\right), \tag{2.9}$$

$$\underline{h}_t^{LSTM} = \sigma(W_o \underline{I}_t)\phi\left(\underbrace{\sum_{k=1}^{t}\left[\prod_{j=k+1}^{t} \sigma(W_f \underline{I}_j)\right]}_{\textit{forget gate}} \sigma(W_i \underline{I}_k)\emptyset(W_{in}\underline{I}_k)\right). \tag{2.10}$$

Dari rumus di atas dapat dikatakan bahwa W_c^{t-k} dan $\prod_{j=k+1}^{t} \sigma(W_f \underline{I}_j)$ memerankan peran sebagai *memory* yang sama untuk SRN dan LSTM. Telah ditemukan studi kasus yang spesial di mana panjang ingatan dari LSTM melebihi ingatan SRN terlepas dari parameter model SRN yaitu W_c , W_{in} seperti pada perhitungan (2.11)

$$\exists W_f \; s.t. \min\left|\sigma(W_f \underline{I}_j)\right| \geq \sigma_{max}\,(W_c), \forall \sigma_{max}(W_c) \in [0,1), \tag{2.11}$$

Kemudian menjadi

$$\left|\prod_{j=k+1}^{t} \sigma(W_f \underline{I}_j)\right| \geq \sigma_{max}(W_c)^{t-k}, t \geq k\,. \tag{2.12}$$

Berdasarkan perhitungan (2.5) dan (2.11), dampak dari *input* $\underline{I}_k$ pada *output* LSTM bertahan lama dari SRN, hal ini menyatakan bahwa selalu terdapat LSTM yang memiliki kapasitas ingatan yang panjang dari SRN.