BAB 2
LANDASAN TEORI

Pada bagian ini dijabarkan literatur-literatur yang mendasari penelitian ini. Literatur di bawah ini didapatkan dari penelitian-penelitian yang pernah dilakukan sebelumnya, dan diharapkan menjadi penjelasan untuk teori-teori yang akan digunakan dalam penelitian ini.

2.1 Machine Learning

Pembelajaran mesin atau *Machine Learning* (ML) adalah teknologi yang berfokus melakukan tugas dalam mempelajari sekumpulan data yang diberikan tanpa di program secara eksplisit. Pelatihan dalam ML bergantung dengan data yang diberikan untuk menyelesaikan permasalahan tertentu, oleh karena itu ML banyak digunakan dalam sistem prediksi, klasifikasi paket jaringan, analisis, sentimental, pengenalan ucapan, diagnosis medis, industri keuangan, pemrosesan sinyal, analisis kelelahan resah, pertanian, dan lainnya [2].

2.2 Deep Learning

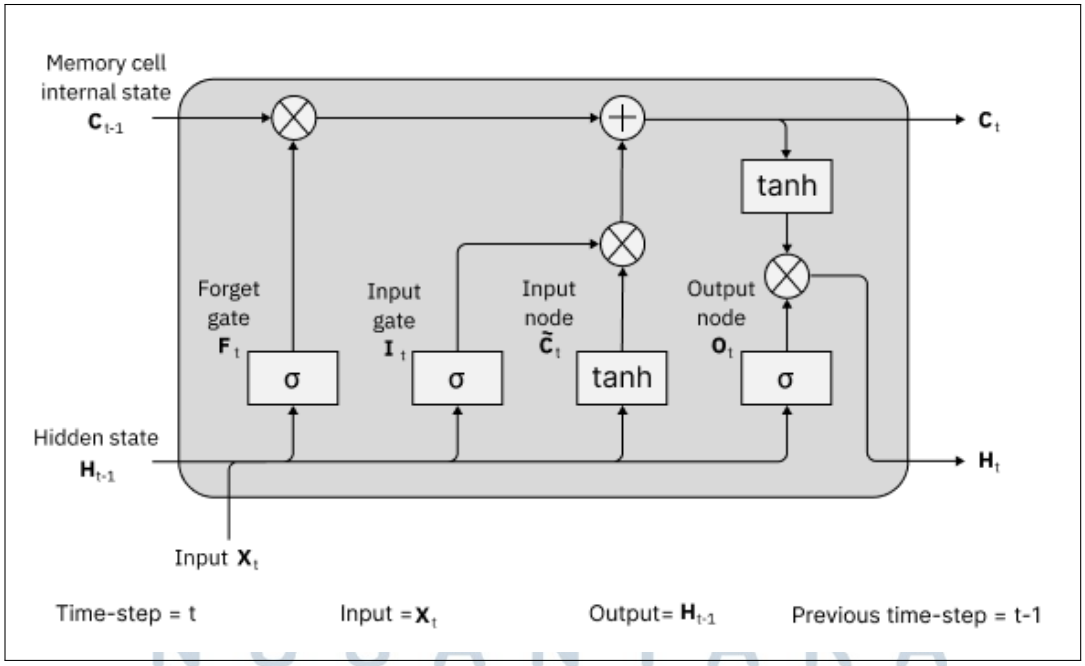
Deep Learning (DL) adalah subbidang ML yang dapat menyelesaikan masalah kompleks dengan melibatkan penggunaan jaringan saraf (*Neural Network*) tanpa memerlukan *feature engineering* secara manual, maka dari itu algoritma DL dapat belajar secara otomatis dan meningkatkan hasilnya dari data. Jaringan saraf tersebut memiliki struktur dan berfungsi seperti otak manusia yang memiliki karakteristik dalam menggunakan desain dari banyak lapisan node yang saling berhubungan (*deep neural networks*) [9, 10]. DL memanfaatkan *Graphics Processing Unit* (GPU) atau *Visual Processing Unit* (VPU) dalam proses pembelajaran secara paralel dibanding dengan penggunaan prosesor *Central Processing Unit* (CPU) [10].

2.3 Long Short-Term Memory

Long Short-Term Memory atau biasa nya disebut dengan LSTM adalah berbasis dari *Recurrent Neural Network* (RNN) yang mempelajari ketergantungan dalam prediksi urutan data dengan panjang variabel. LSTM dikembangkan oleh Hochreiter dan Schmidhuber pada tahun 1997. *Neural Network* LSTM terdiri dari *input layer*, satu atau lebih sel memori, dan *output layer*. Jumlah *neuron* pada lapisan *input* sama dengan jumlah *explanatory variabel*. Karakteristik pada LSTM *network* terdapat pada *layer* yang tersembunyi dari apa yang disebut sel memori. Masing-masing sel memori memiliki tiga gerbang yang menahan dan menyesuaikan *state cell* C_t, yaitu *Forget gate*(f_t) , *Input gate* (i_t), dan *Output gate* (o_t) [11, 12].

Gerbang pada LSTM bekerja sebagai filter, dan masing-masing memiliki tujuan yang berbeda:

- Forget gate: mendefinisikan informasi apa yang dihapus oleh *state cell*.
- Input gate: menentukan informasi apa yang ditambahkan oleh *state cell*.
- Output gate: menentukan informasi apa yang digunakan oleh *state cell*.



Gambar 2.1. Arsitektur memori LSTM.

Berikut adalah persamaan pada setiap gerbang dalam arsitektur blok memori LSTM yang dapat di lihat pada Gambar 2.1:

- Input node
  Pada tahap ini dikhususkan untuk memperbarui komponen blok *input* x_t yang menggabungkan *input* saat ini dan *output* dari LSTM *unit* $h_{(t-1)}$ tersebut pada iterasi terakhir.

$$\tilde{C}_t = \tanh(W_{xc}x_t + W_{hc}h_{(t-1)} + b_c) \tag{2.1}$$

- Input gate
  Pada tahap ini, *Input gate* yang menggabungkan nilai sel saat ini x_t, *output* pada LSTM *unit* $h_{(t-1)}$, dan nilai sel $h_{(t-1)}$ di iterasi terakhir akan di *update*.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{(t-1)} + b_i) \tag{2.2}$$

- Forget gate
  Pada tahap ini, LSTM *unit* menentukan informasi apa yang harus dihapus dari *cell state* sebelumnya $h_{(t-1)}$. Maka dari itu, *activation values* f_t pada *Forget gate* t dihitung berdasarkan x_t *output* saat ini $h_{(t-1)}$ dan *state* $h_{(t-1)}$ dari sel memori dari *time-step* sebelumnya (t-1).

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{(t-1)} + b_f) \tag{2.3}$$

- Output gate
  Pada tahap ini menghitung nilai sel, yang menggabungkan *current input* x_t, hasil dari LSTM $h_{(t-1)}$, dan nilai sel h_t iterasi sebelumnya.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{(t-1)} + b_o) \tag{2.4}$$

- Cell gate
  Pada tahap ini menghitung nilai sel, yang menggabungkan *Input node* $\tilde{C}_t$, *Input gate* i_t, dan *Forget gate* f_t dengan nilai sel sebelumnya.

$$C_t = f_t \odot C_{(t-1)} + i_t \odot \tilde{C}_t \tag{2.5}$$

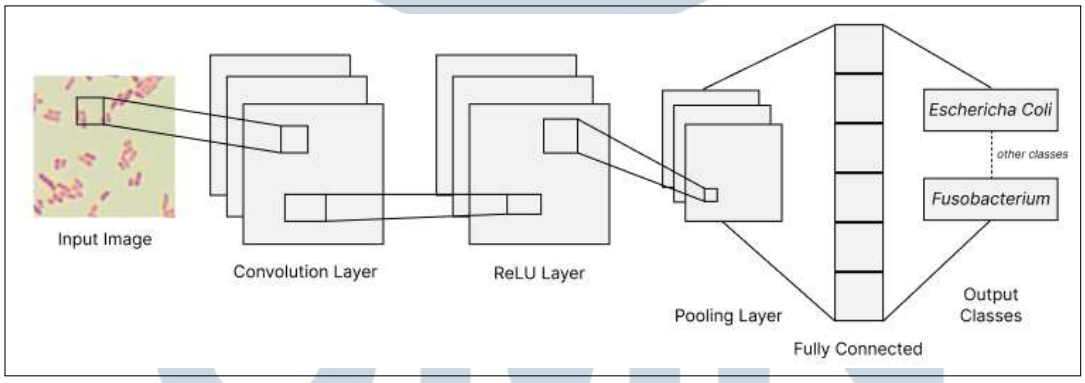
- Hidden gate

  Pada tahap ini menghitung blok *output*, yang menggabungkan nilai sel saat ini dengan nilai *Output gate* saat ini dengan rumus berikut:

$$h_t = \tanh(C_t) \odot o_t \tag{2.6}$$

2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu algoritma yang paling terkenal dan umum digunakan dan diterapkan dalam banyak bidang *Computer Vision*, *Speech processing*, *Face Recognition*, dan lainnya. CNN dapat mengidentifikasikan fitur yang relevan secara otomatis tanpa pengawasan manusia. Struktur desain algoritma CNN berbentuk seperti area korteks visual pada otak manusia dan hewan yang akan memproses informasi dalam bentuk rangsang visual. Pada umumnya, CNN terdiri dari beberapa *hidden layer* yand dapat di lihat pada Gambar 2.2, yaitu *convolutional layer*, *pooling layer*, *ReLU layer*, dan *fully connected layer* [9, 10].

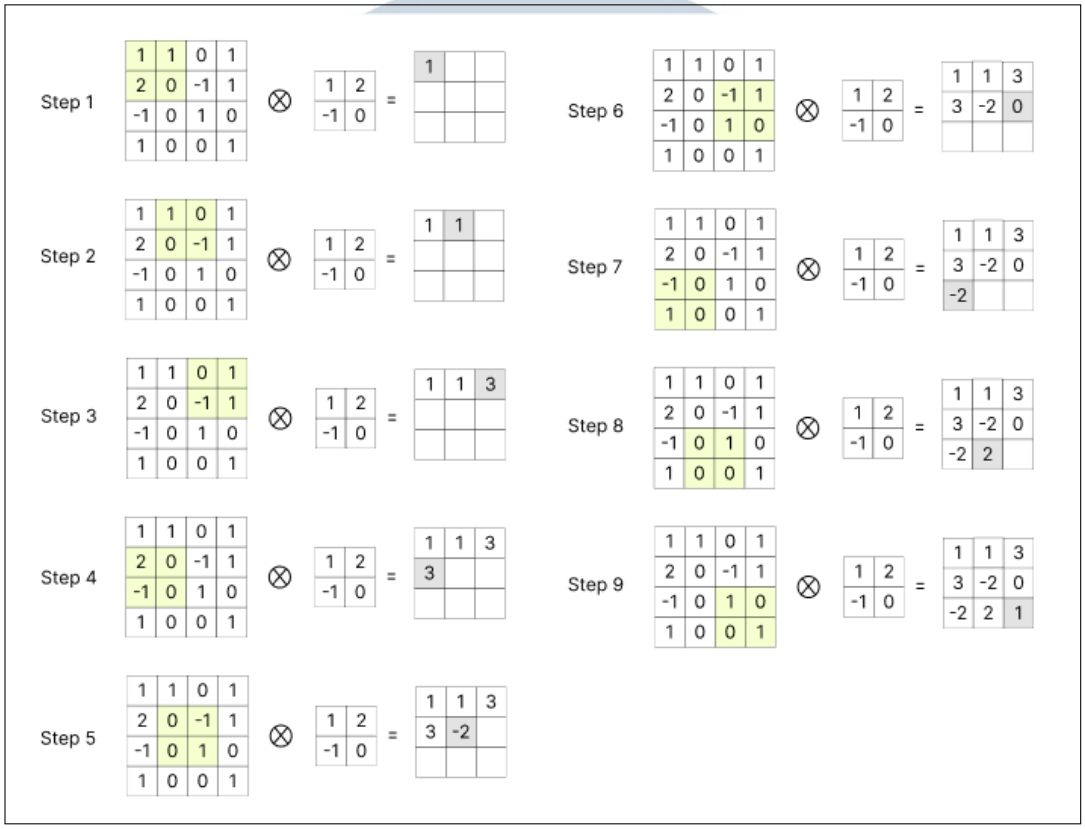


Gambar 2.2. Contoh Arsitektur CNN dalam klasifikasi gambar.

2.4.1 Convolution Layer

Dalam CNN, *Convolutional layer* menerima *input* gambar multi saluran (multi-channeled) dengan format vektor. Misalnya, format gambar RGB yang memiliki tiga saluran, sementara itu format gambar abu-abu (grayscale) memiliki satu saluran. Pada Gambar 2.3 adalah contoh proses kerja dalam *Convolutional layer* gambar abu-abu 4x4 dan menggunakan *kernel* 2x2 dengan nilai acak. *Kernel* akan bergerak secara horizontal dan vertikal ke seluruh gambar berulang kali

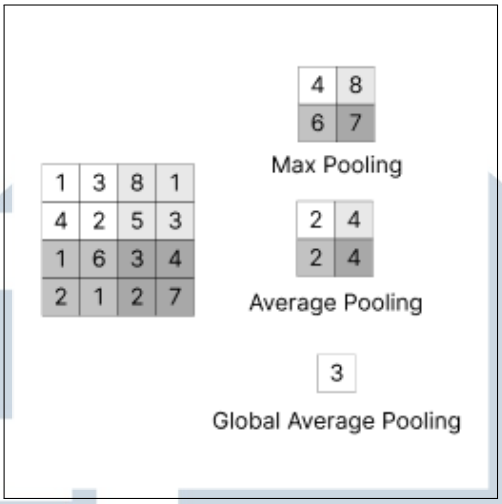
dan juga menghitung secara bersamaan dengan menggunakan perkalian titik (Dot Product) diantara vektor *input* gambar dan *kernel* [9, 10].



Gambar 2.3. Proses perhitungan dalam *Convolutional layer*.

2.4.2 Pooling Layer

Pooling layer menghasilkan fitur berukuran kecil dari fitur berukuran besar. Dalam *pooling layer* terdapat metode yang berbeda yaitu *tree pooling*, *gated pooling*, *average pooling*, *min pooling*, *max pooling*, *global average pooling* (GAP), dan *global max pooling*. Pada Gambar 2.4 adalah contoh metode *pooling layer* yang lebih sering digunakan, yaitu *average pooling*, *max pooling*, dan *global average pooling* (GAP) [9, 10].



Gambar 2.4. Tiga tipe *pooling*.

2.4.3 ReLU Layer

Dalam CNN, fungsi *Rectified Linear Units* (ReLU) yang paling umum digunakan sebagai fungsi aktivasi $f(x) = max(0,x)$. Fungsi ReLU digunakan untuk mengubah nilai *input* menjadi bilangan bulat positif yang meningkatkan sifat nonlinearitas fungsi keputusan [9, 10].

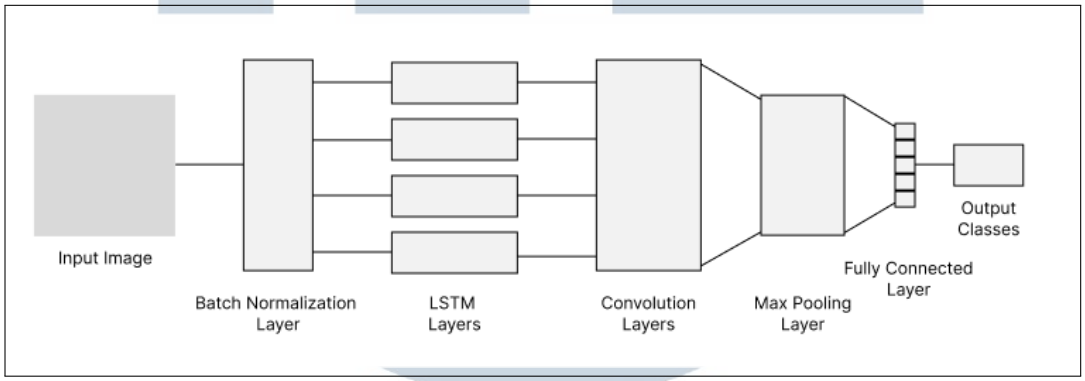
2.4.4 Fully-Connected Layer

Fully-Connected Layer digunakan sebagai pengklasifikasi CNN, setiap *neuron* terhubung dengan semua *neuron* pada *layer* sebelumnya. *Fully connected layer input* berasal dari *output* dari *convolutional layer* atau *pooling layer* yang terakhir [9, 10].

2.5 LSTM-CNN Hybrid Model

Pada Gambar 2.5 mempresentasikan arsitektur model LSTM-CNN. Pada Tahap awal *input* gambar melalui *Batch Normalization Layer* terlebih dahulu. *Batch Normalization Layer* menerapkan transformasi yang mempertahankan keluaran rata-rata mendekati 0 dan standar deviasi keluaran mendekati 1. Selanjutnya, hasil *output* dimensi dari *Batch Normalization Layer* akan diubah menjadi dimensi yang dapat digunakan dalam sel LSTM sebelum berlanjut ke LSTM *layer*. Fungsi aktivasi sel LSTM menggunakan tanh (*hyperbolic tangent*) dan diberikan *dropout rate* untuk mencegah *overfitting* [7].

Dari karakteristik LSTM, LSTM akan mengingat ketergantungan nilai jangka panjang dan bentuk gambar dalam pola tertentu. Hasil LSTM adalah informasi yang telah kalkulasi di berdasarkan gerbang dan mengumpulkannya informasi melalui sel-sel memori yang bergabung sendiri secara linear sebagai perantara untuk mendapatkan output dari lapisan tersembunyi. Hasil dari LSTM berlanjut dalam *Convolutional Layer* dan mengekstrak fitur penting dari *input*. Fungsi aktivasi dalam *Convolutional Layer* menggunakan *Rectified Linear Unit* (ReLU), kemudian lanjut ke *Max Pooling Layer* dan diberikan *dropout layer* sebelum memasuki *Fully-Connected Layer*[7].



Gambar 2.5. Arsitektur LSTM-CNN *Hybrid Model*.

2.6 Confusion Matrix

Confusion Matrix adalah ringkasan berbentuk tabel yang digunakan untuk menilai kinerja model klasifikasi, masing-masing kelas terdapat jumlah prediksi benar dan salah dalam nilai hitungan. Terdapat empat jumlah *tuple* yang berbeda, yaitu *true positives* (TP) yang menghasilkan jumlah *tuple* aktual yang berlabel positif oleh model, *true negatives* (TN) yang menghasilkan jumlah *tuple* aktual yang berlabel negatif oleh model, *False Positives* (FP) yang menghasilkan jumlah *tuple* negatif yang salah dilabeli oleh model, dan *False Negatif* (FN) yang menghasilkan jumlah *tuple* positif yang salah dilabeli oleh model. *Accuracy* dan *error rate* digunakan untuk klasifikasi data dengan jumlah data pada setiap kelas relatif sama, sementara itu *precision* dan *recall* digunakan untuk klasifikasi data yang tidak seimbang [10].