BAB III
Landasan Teori

Bab ini akan membahas berbagai teori yang melandasi penulis dalam penelitian yang akan dilakukan.

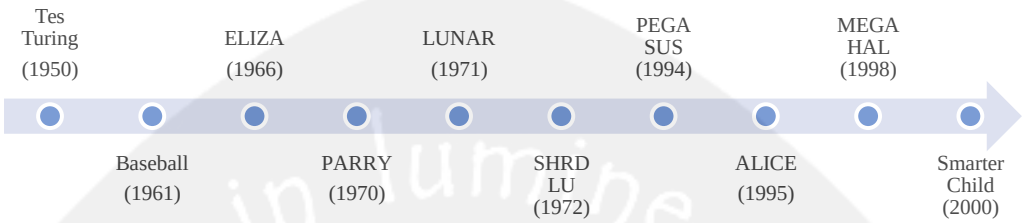
3.1. Aplikasi *Chatbot*

Joseph Weizenbaum menciptakan ELIZA sebagai *chatbot* pertama. Sistem ELIZA dibuat untuk meniru bahasa psikoterapis dan sistemnya didasarkan pada pencocokan kata kunci. Setelah ELIZA, *chatbot* berikutnya adalah PARRY. Berbeda dengan ELIZA, PARRY dimodelkan sebagai pasien paranoid saat dia berbicara ke terapisnya dengan mengandalkan pencocokan pola dan beberapa trik, seperti bercerita, jika tidak ada jawaban yang ditemukan.

Beberapa aplikasi *chatbot* muncul selain bidang kedokteran. BASEBALL dapat menjawab pertanyaan tentang liga bisbol Amerika Serikat dan LUNAR dapat menjawab pertanyaan tentang geologi batuan dan tanah di bulan. Keduanya mirip dengan *chatbot* ELIZA namun didasarkan pada *knowledge base system*. Setelah itu, muncul SHRDLU yang sistemnya dapat memindahkan berbagai balok yang berwarna dan berbentuk. Aplikasi *chatbot* juga diterapkan untuk pemesanan penerbangan yang bernama PEGASUS, di mana pengguna dapat mencari dan memesan penerbangan *American Airlines*.

Ketidakmampuan yang kelihatan jelas dari *chatbot* ELIZA dan PARRY adalah responnya yang dapat diprediksi, berlebihan, dan kurang memiliki kepribadian. Oleh karena itu, pada tahun 1995 *chatbot* ALICE (*Artificial Linguistic Internet Computer Entity*) dibuat untuk mengatasi hal tersebut dan sistemnya berbasiskan pencocokan pola dengan bahasa XML, sehingga sangat mudah untuk menulis program *chatbot* sendiri tanpa pengetahuan pemrograman. Akan tetapi, teknik pencocokan pola tampaknya tidak mengarah pada terobosan dalam kecerdasan buatan. Arah baru penelitian *chatbot* adalah pengenalan sistem MegaHAL yang didasarkan pada teknik pembelajaran mesin. Setelah itu, aplikasi *chatbot* semakin berkembang melalui pesan instan dan jaringan SMS

dengan SmarterChild, yang merupakan dasar dari aplikasi *virtual assistant* saat ini. Berikut adalah *timeline* dari aplikasi *chatbot*.



Gambar 3.1. *Timeline* dari Aplikasi *Chatbot* (Wetstein, 2017)

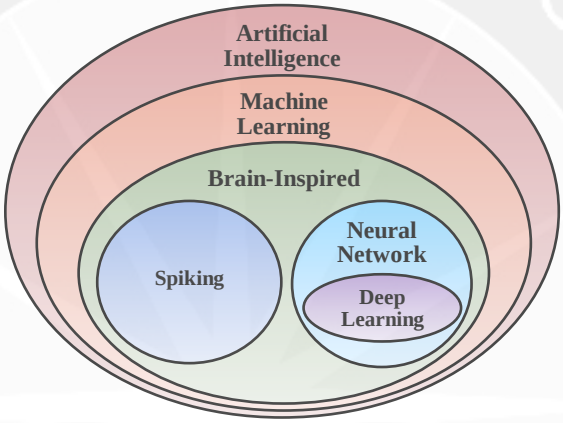
3.2. *Natural Language Processing*

Natural Language Processing (NLP) adalah cabang dari kecerdasan buatan yang termotivasi dari teknik komputasi untuk analisa otomatis dan representasi bahasa manusia dengan tujuan memudahkan manusia untuk menjalin komunikasi dengan mesin. Jika mengikuti tren saat ini, penelitian NLP terbaru semakin bertambah dan berfokus pada penggunaan metode baru dari *deep learning* dengan praktik yang paling populer adalah pembelajaran yang diawasi (*supervised learning*). NLP telah menghasilkan beragam aplikasi yang berguna bagi manusia, seperti terjemahan mesin, kategorisasi teks, deteksi spam, ekstraksi informasi, peringkas dokumen, sistem percakapan, dan aplikasi medis (Khurana *et al.*, 2017).

3.3. *Deep Learning*

Deep learning adalah kelas dari pembelajaran mesin yang bekerja dalam banyak layer pemrosesan untuk ekstrasi dan transformasi dengan tujuan mempelajari representasi yang hirarkis dari data statistik non-linier. Akhir-akhir ini *deep learning* telah menghasilkan teknologi mutakhir diberbagai bidang pekerjaan, seperti klasifikasi (*supervised*) dan analisis pola (*unsupervised*) terutama untuk memahami data seperti gambar, suara, dan teks.

Alasan penting yang mendasari populernya *deep learning* saat ini adalah akurasi, efisiensi, dan fleksibilitas yang didukung oleh meningkatnya kemampuan perangkat keras untuk membantu proses komputasinya yang berat. *Deep learning* menawarkan cara untuk memanfaatkan sejumlah besar data dengan sedikit rekayasa tangan karena secara otomatis mesin melakukan ekstraksi fitur secara rinci, dimana hal itu memerlukan banyak waktu dan upaya yang intens. *Deep learning* berada pada titik potong yang bisa dilihat pada ilustrasi Gambar 3.2.

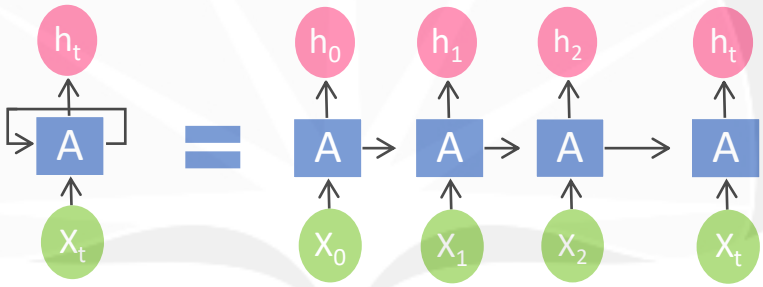


Gambar 3.2. Ruang Lingkup *Deep Learning* (Sze *et al.*, 2017)

3.4. *Recurrent Neural Network* (RNN)

Recurrent Neural Network (RNN) merupakan salah satu kelas dari *deep neural network* yang diawasi. Pelatihan RNN dalam tingkat yang diawasi membutuhkan *dataset* pelatihan dari pasangan input-target dengan tujuan meminimalkan perbedaan nilai *loss* pasangan itu dengan mengoptimalkan bobot jaringan (Salehinejad *et al.*, 2017). RNN dibentuk dari neuron buatan dengan satu atau lebih umpan balik yang berulang. Pada setiap langkah waktu, neuron akan menerima data, melakukan komputasi, dan menghasilkan keluaran. RNN menangkap dinamika yang kaya dari keadaan tersembunyi untuk konteks jangka panjang, sehingga membentuk model yang ekspresif dan sangat kuat untuk tugas-tugas yang sekuens, seperti pengenalan suara, sintesis ucapan, visi mesin, generasi deskripsi video, dan rangkaian teks.

RNN memiliki tiga lapisan yaitu layer *input*, layer tersembunyi yang berulang, dan layer *output* (Salehinejad *et al.*, 2017). Layer input memiliki unit input, terkoneksi penuh ke unit tersembunyi yang ada di layer tersembunyi. Unit tersembunyi itu terhubung satu sama lain secara berulang. Layer tersembunyi bisa didefinisikan sebagai "memori" atau ruang keadaan yang berdimensi tinggi dengan dinamika non-linier untuk mengingat dan memproses informasi masa lalu. Keadaan tersembunyi akan merangkum semua informasi unik yang diperlukan sebagai keadaan terakhir dari jaringan, melalui serangkaian langkah waktu. Informasi itu lalu terintegrasi, sehingga mampu menentukan perilaku jaringan di masa depan dan melakukan prediksi yang akurat di layer *output*. Arsitektur RNN bisa dilihat pada ilustrasi Gambar 3.3.



Gambar 3.3. Arsitektur RNN Sederhana (Christopher Olah via Github, 2015)

3.5. Mekanisme Gerbang RNN

Salah satu kelemahan RNN adalah pembelajaran jangka panjang dengan *gradient descent* bisa menghasilkan masalah menghilang atau meledaknya gradien (Salehinejad *et al.*, 2017). Pada saat pembelajaran, ketika nilai gradien dipropagasikan kembali, nilai itu terus dikalikan dengan bobot yang lebih kecil dari satu (<1.0). Oleh karena itu, nilai gradien yang dihasilkan saat iterasi lama kelamaan cenderung mengecil atau mengalami kehilangan nilai. Nilai gradien secara eksponensial juga bisa membesar, karena nilai gradien dikalikan dengan bobot yang lebih besar dari satu (>1.0). Maka perlu mencegah nilai gradien tersebut menjadi angka-angka yang terlalu kecil atau terlalu besar.

Salah satu cara mengatasi masalah menghilang atau meledaknya gradien adalah memodifikasi arsitektur model dengan memasukkan unit gerbang yang dirancang khusus untuk menyimpan informasi selama waktu periode yang lama. Mekanisme gerbang yang paling dikenal saat ini adalah *Long-Short Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). LSTM mampu menangani penghafalan dan pengingatan kembali untuk jangka panjang, khususnya data yang sangat besar. LSTM pada prinsipnya dapat menggunakan unit memorinya untuk mengingat informasi yang jaraknya jauh dan melacak berbagai atribut teks yang sedang diproses (Karpathy *et al.*, 2016). Sementara itu, GRU memiliki parameter yang lebih sedikit dari LSTM, sehingga cocok untuk data yang sedikit, agar tidak terjadi *overfitting*. Selain itu, GRU memberikan konvergensi yang lebih cepat dan hasilnya bisa disandingkan dengan LSTM (Chung *et al.*, 2014).

Persamaan variabel dari LSTM yaitu x_t sebagai input vektor ke LSTM, F_t sebagai vektor aktivasi *forget gate*, I_t sebagai vektor aktivasi *input gate*, O_t sebagai vektor aktivasi *output gate*, H_t sebagai vektor *hidden state* atau dikenal sebagai vektor keluaran dari unit LSTM, C_t sebagai vektor *cell state*, W-U-b sebagai matriks bobot dan parameter vektor bias yang perlu dipelajari selama pelatihan. Bentuk persamaan LSTM yang ringkas adalah sebagai berikut.

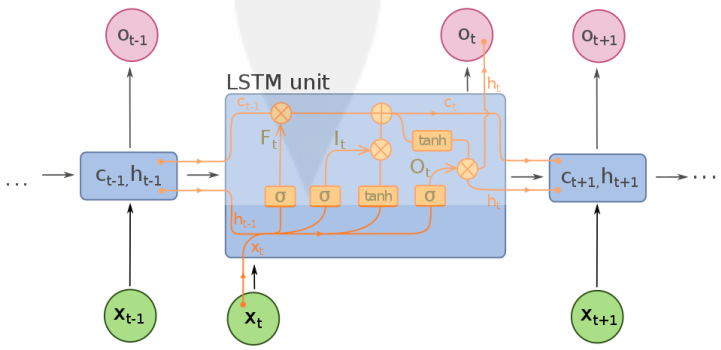
$$F_t = \sigma_g\,(W_f\,x_t + U_f\,h_{t-1} + b_f) \quad (1)$$

$$I_t = \sigma_g\,(W_i\,x_t + U_i\,h_{t-1} + b_i) \quad (2)$$

$$O_t = \sigma_g\,(W_o\,x_t + U_o\,h_{t-1} + b_o) \quad (3)$$

$$C_t = f_t \circ c_{t-1} + i_t \circ \sigma_g\,(W_c\,x_t + U_c\,h_{t-1} + b_c) \quad (4)$$

$$H_t = O_t \circ \sigma_h\,(c_t) \quad (5)$$



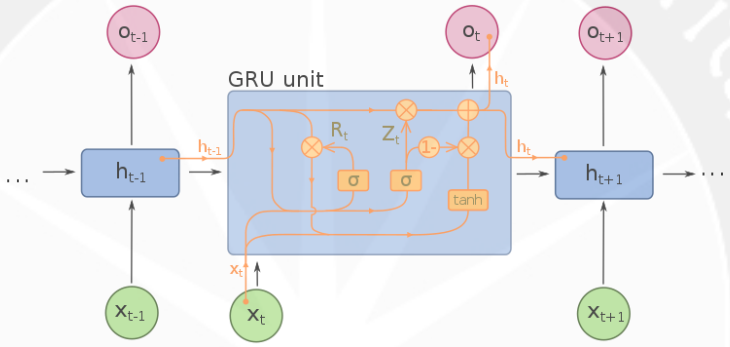
Gambar 3.4. Unit LSTM (François Deloche via Wikimedia Commons, 2017)

Persamaan variabel dari GRU yaitu x_t sebagai input vektor ke GRU, Z_t sebagai vektor *update gate*, R_t sebagai vektor *reset* gate, O_t sebagai vektor aktivasi *output gate*, H_t sebagai vektor *hidden state* atau dikenal sebagai vektor keluaran dari unit LSTM, W-U-b sebagai matriks bobot dan parameter vektor bias yang perlu dipelajari selama pelatihan. Bentuk persamaan GRU yang ringkas adalah sebagai berikut.

$$F_t = \sigma_g (W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$R_t = \sigma_g (W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$C_t = Z_t \circ h_{t-1} + (1-Z_t) \circ \sigma_g (W_h x_t + U_h (r_t \circ h_{t-1}) + b_h) \quad (3)$$



Gambar 3.5. Unit GRU (François Deloche via Wikimedia Commons, 2017)

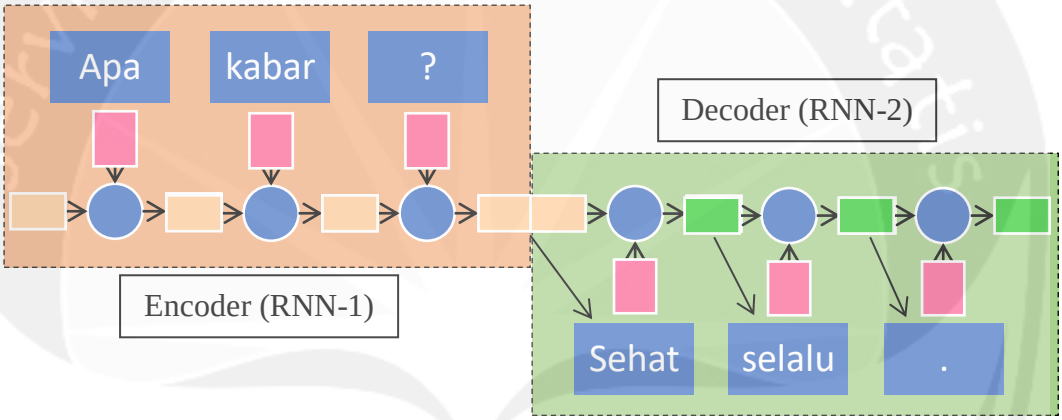
3.6. *Sequential to Sequential* (SEQ2SEQ)

Recurrent Neural Network (RNN) sejauh ini berguna untuk pemodelan informasi yang sekuensial seperti pada tugas-tugas bahasa alami. Model RNN yang populer saat ini adalah *sequential-to-sequential* (SEQ2SEQ). Model SEQ2SEQ dibentuk dari dua jaringan RNN, yaitu *encoder* dan *decoder*. Jaringan RNN pada *encoder* melakukan *encoding* untuk sekuens input ke dalam vektor konteks dengan ukuran tetap, sedangkan *decoder* menggunakan vektor konteks tadi sebagai "benih" untuk menghasilkan sekuens target. Oleh karena itu, model SEQ2SEQ sering juga disebut sebagai model *encoder-decoder*.

Aplikasi di bidang bahasa alami mulai berkembang karena kemampuan ingatan RNN untuk konteks jangka panjang. Model percakapan SEQ2SEQ mampu memprediksi kata berikutnya dengan diberikan kata atau kalimat konteks sebelumnya dari suatu percakapan. Hal itu akan menghasilkan model

generatif yang merangkai kata per kata dan membentuk kalimat yang benar-benar baru, dibandingkan dengan model *retrieval* yang hanya mengambil kalimat utuh pada pra-konstruksi repositori.

Penelitian arsitektur SEQ2SEQ mengalami kemajuan untuk menghasilkan model dengan akurasi yang tinggi. Perkembangan model SEQ2SEQ ada yang memakai gabungan dua gerbang yang berbeda (LSTM dan GRU) pada *encoder* model (Bay *et al.*, 2017). Ada juga yang memakai mekanisme gerbang yang *multilayer*, contohnya 2 layer LSTM pada *encoder* dan 2 layer LSTM pada *decoder*. Arsitektur sederhana SEQ2SEQ bisa dilihat pada ilustrasi Gambar 3.6.



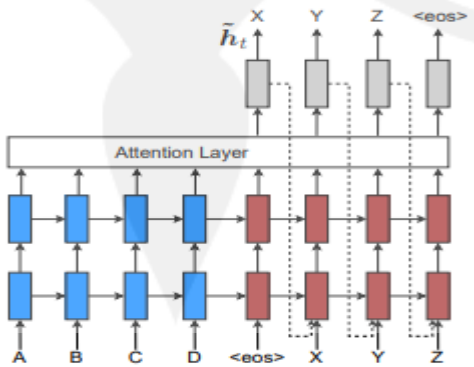
Gambar 3.6. Arsitektur SEQ2SEQ (CIS530 University of Pennsylvania, 2018)

3.7. Mekanisme Perhatian

Konsep perhatian telah mendapatkan popularitas baru-baru ini dalam pelatihan jaringan saraf dengan mengambil contoh dari mekanisme perhatian visual manusia. Perhatian visual manusia pada dasarnya memberi fokus pada wilayah tertentu dari gambar dengan "resolusi tinggi" sambil melihat gambar di sekitarnya dalam "resolusi rendah", dan kemudian menyesuaikan titik fokus itu dari waktu ke waktu. Mekanisme perhatian kemudian berkembang dan memungkinkan model untuk belajar keberpihakan antara modalitas yang berbeda (Luong *et al.*, 2015), misalnya, antara visual gambar dan keterangan gambar pada pembuatan *image caption*.

Pada tugas bahasa alami, input dikodekan ke dalam ukuran tetap, sehingga kalimat yang panjang tidak dapat digeneralisasi jauh lebih panjang daripada kapasitas tetapnya. Bahdanau (2015) lalu menerapkan mekanisme perhatian untuk menyelaraskan kata-kata pada tugas terjemahan mesin sehingga mengatasi kinerja buruk pada kalimat yang panjang. Mekanisme perhatian mampu memperluas vektor ukuran tetap itu dengan memungkinkan model secara otomatis mencari bagian dari sumber kalimat yang relevan untuk memprediksi kata target, tanpa harus membentuk bagian-bagian itu sebagai segmen padat secara eksplisit.

Mekanisme perhatian membantu jaringan mengingat aspek-aspek tertentu dari input dengan lebih baik. Mekanisme perhatian digunakan ketika menghasilkan setiap kata dalam *decoder* (Lopyrev, 2015). Untuk setiap kata keluaran, mekanisme perhatian menghitung bobot dari setiap kata input yang menentukan seberapa banyak perhatian yang harus diberikan pada kata input itu. Penjumlahan bobot maksimum adalah satu (< = 1) yang kemudian digunakan untuk menghitung rata-rata tertimbang dari layer tersembunyi terakhir yang dihasilkan setelah memproses setiap kata input. Rata-rata bobot ini, disebut sebagai konteks, kemudian dimasukkan ke layer *softmax* bersama dengan layer tersembunyi terakhir dari langkah saat ini dari proses *decoding*. Lapisan mekanisme perhatian (*attention layer*) pada jaringan SEQ2SEQ bisa dilihat pada ilustrasi Gambar 3.7.



Gambar 3.7. *Attention Layer* Pada Jaringan SEQ2SEQ (Luong *et al.*, 2015)

3.8. *Gradient Descent*

Gradient Descent (GD) adalah metode pengoptimalan yang sederhana dan populer dalam *deep learning*. Model *deep learning* biasanya memiliki parameter (bobot dan bias) dan fungsi biaya (*error*) untuk mengevaluasi seberapa baik set dari parameter tersebut. Ide dasar GD adalah menyesuaikan bobot model dengan mencari turunan dari fungsi biaya (*error*) sehubungan dengan masing-masing anggota matriks bobot di dalam model. GD memulai dari nilai awal parameter tersebut dan secara iteratif bergerak menuju nilai parameter yang meminimalisasi nilai dari fungsi biaya (*error*) tersebut.

Ada beberapa ekstensi dari metode pengoptimalan GD. Contoh ekstensi tersebut adalah *Stochastic Gradient Descent* (SGD), *Adaptive Gradient Algorithm* (AdaGrad), *Root Mean Square Propagation* (RMSProp), dan *Adaptive Moment Estimation* (Adam). SGD mempertahankan tingkat pembelajaran tunggal (*alpha*) untuk semua pembaruan bobot dan tingkat pembelajaran tidak berubah selama pelatihan. AdaGrad mempertahankan tingkat pembelajaran per-parameter sehingga meningkatkan kinerja pada masalah dengan gradien yang tersebar. RMSProp juga mempertahankan laju pembelajaran per-parameter yang disesuaikan berdasarkan rata-rata besarnya bobot gradien terbaru, seperti seberapa cepat bobot itu berubah, sehingga RMSProp baik digunakan untuk masalah pembaruan bobot terbaik secara sekuensial (online) dan non-stasioner. Adam lalu menggabungkan keuntungan dari dua ekstensi itu untuk menghitung laju pembelajaran individu yang adaptif untuk parameter yang berbeda dari perkiraan momen pertama seperti RMSProp, dan momen kedua gradien (Kingma *et al.*, 2014). Pada praktiknya, Adam saat ini direkomendasikan sebagai algoritma yang digunakan secara *default*, dan seringkali bekerja dengan lebih baik daripada RMSProp dan AdaGrad.

3.9. *Framework Deep Learning*

Pelatihan RNN membutuhkan kemampuan komputasi yang tinggi dan banyak implementasi *deep learning* memakai *General Purpose Graphics*

Processing Unit (GPGPU) untuk memanfaatkan kemampuan komputasi pararel supaya proses komputasi *deep learning* menjadi lebih cepat. Kerangka kerja *deep learning* menawarkan blok-blok pembangunan untuk merancang, melatih, dan memvalidasi jaringan melalui antarmuka pemrograman tingkat tinggi. Hal itu dapat dilakukan dengan memakai sebuah *library* yang dapat mendukung beragam jenis perangkat keras yang berbeda. Misalnya, suatu program yang dibangun dengan Keras diatas kerangka kerja Tensorflow, bisa berjalan pada CPU maupun GPU tanpa tanpa perlu mengubah kode program itu.

Saat penelitian ini berlangsung, sudah ada beberapa kerangka kerja *deep learning* yang populer serta didukung oleh komunitas dari pengembang dan perusahaan teknologi, seperti Theano, Tensorflow, Torch, Microsoft Cognitive Toolkit, Neon, Chainer, dan lain sebagainya. Kerangka kerja tersebut menyediakan fitur yang serupa dengan menawarkan dukungan luas untuk algoritma *deep learning*, memiliki banyak panduan dan dokumentasi, dan bahkan menawarkan visualisasi dari proses pelatihan model. Pada akhirnya kerangka kerja tersebut mempunyai tujuan umum, yaitu untuk manajemen komputasi yang intens dimana melibatkan proses iterasi yang cukup banyak dengan jumlah data yang besar.

3.10. Tensorflow

Tensorflow merupakan kerangka kerja *machine learning* yang dibangun secara terbuka (*opensource*) yang didukung oleh perusahaan teknologi Google. Secara umum Tensorflow adalah antarmuka pemrograman untuk perhitungan numerik memakai grafik aliran data. Node-node yang ada di dalam grafik mewakili operasi matematika, sedangkan tepi grafik mewakili array data multidimensi (*tensor*) yang mengalir di antara mereka. Komputasinya dapat dijalankan dengan sedikit atau tanpa perubahan pada beragam sistem yang heterogen, mulai dari perangkat seluler hingga perangkat komputasi seperti kartu GPU. Sistem Tensorflow itu fleksibel dan dapat digunakan untuk mengekspresikan berbagai macam algoritma, termasuk pelatihan dan inferensi untuk model *deep learning* (Abadi *et al.*, 2016) dan telah digunakan untuk

berbagai bidang penelitian, seperti pengenalan suara, visi komputer, robot, pengambilan informasi, pemrosesan bahasa alami, ekstraksi informasi geografis, dan penemuan obat-obatan secara komputasional. Oleh karena itu, Tensorflow termasuk sebagai *library* yang paling populer dengan jumlah pengembang dan dukungan komunitas yang besar.