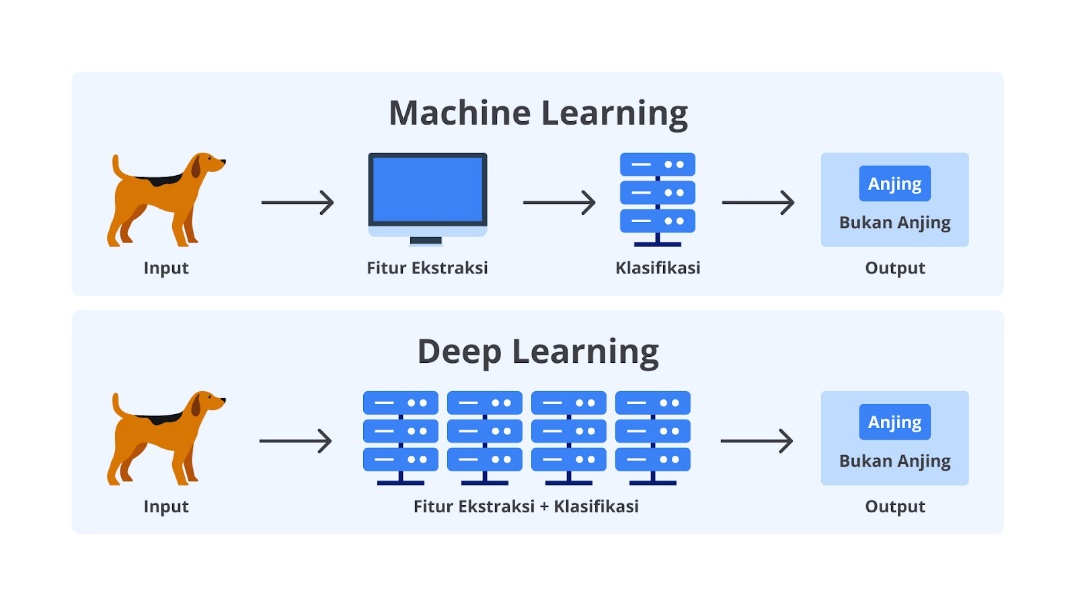
Kita sudah berada di penghujung deep learning untuk semua orang. Sampai sejauh ini, Anda diharapkan telah memahami konsep dasar deep learning hingga membuat model deep learning. Sekarang, mari kita rangkum secara saksama.

**Pengenalan Deep Learning**

Deep learning adalah bagian dari bidang keilmuan AI yang mengajarkan komputer untuk memproses data yang terinspirasi dari cara kerja otak manusia. Model deep learning dapat mengerjakan tugas yang lebih kompleks dari machine learning. Dengan kompleksitas yang cukup tinggi, model deep learning dapat mengenali gambar, teks, suara, dan data lainnya.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Anda pasti masih ingat pada modul Data untuk AI, kita mempelajari data tidak terstruktur berupa gambar, suara, dan bentuk lainnya. *Nah*, dengan menggunakan deep learning kita dapat membuat sebuah sistem AI yang dapat mengotomatiskan tugas-tugas yang biasanya membutuhkan kecerdasan manusia seperti mendeskripsikan gambar atau mengubah file suara ke dalam teks secara langsung seperti gambar di atas.

Lalu, mengapa deep learning ini menjadi bagian yang penting dari pengenalan AI? Seperti yang telah disinggung sebelumnya, deep learning merupakan model yang dapat melatih komputer untuk berpikir layaknya manusia sehingga dapat menggantikan tugas yang selama ini kita lakukan secara manual.

Deep learning terdiri dari 3 atau lebih lapisan atau biasa disebut *layers*. Ia berperan sebagai pengganti otak manusia sebagai jaringan saraf yang memungkinkan model mempelajari data yang sangat besar.

Lantas, apa itu jaringan saraf manusia yang ada pada layers tersebut? Pertanyaan tersebut akan segera terjawab karena kita akan mempelajari dan mengenali jaringan saraf atau artificial neural network yang berperan dalam pembangunan deep learning.

**Mengenal Artificial Neural Network**

ANN merupakan salah satu model ML yang multiguna, *powerful*, dan memiliki skalabilitas tinggi. Dengan kelebihan tersebut, ANN sangat ideal untuk dipakai dalam menangani masalah ML yang sangat kompleks, mulai dari mengklasifikasi miliaran gambar, mengenali ratusan bahasa dunia, merekomendasikan video ke ratusan juta pengguna, sampai belajar mengalahkan juara dunia permainan papan GO.

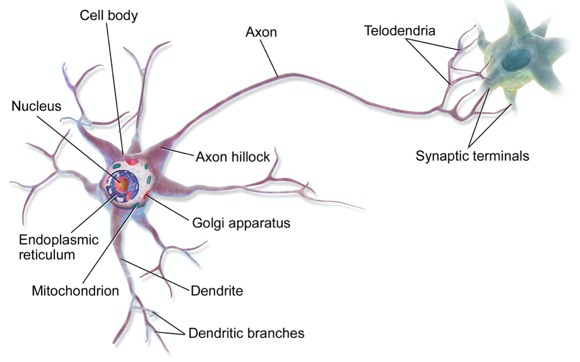
Namun, tahukah Anda apa itu ANN? Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan adalah sebuah model machine learning yang terinspirasi dari neuron/saraf yang terdapat pada otak manusia.

Sebelum mengenali jaringan saraf manusia pada komputer, sebaiknya kita mempelajari sedikit terkait saraf biologis (neuron) dalam otak manusia sesungguhnya.

National Institute of Neurological Disorders and Stroke[7] dalam tulisannya yang berjudul “*Brain Basics: The Life and Death of a Neuron*” menyatakan bahwa neuron atau saraf adalah pembawa pesan/informasi. Mereka menggunakan impuls listrik dan sinyal kimiawi untuk mengirimkan informasi antara area otak yang berbeda, serta antara otak dan seluruh sistem saraf.

Sebuah saraf terdiri dari 3 (tiga) bagian utama, yaitu akson, dendrit, dan badan sel yang di dalamnya terdapat nukleus. Nukleus berisi materi genetik dan bertugas mengontrol seluruh aktivitas sel. Akson adalah cabang yang terlihat seperti ekor yang panjang. Ia bertugas untuk mengirimkan pesan dari sel. Panjang akson berkisar antara beberapa kali lebih panjang dari badan sel, bahkan hingga 10 ribu kali lebih panjang dari badan sel; sedangkan dendrit adalah cabang-cabang pendek yang terlihat seperti cabang pohon yang tugasnya menerima pesan untuk sel.

Setiap ujung akson dari sebuah neuron terhubung dengan dendrit dari neuron lainnya. Neuron berkomunikasi satu sama lain dengan mengirimkan senyawa kimia yang disebut neurotransmitter. Ia melintasi ruang kecil (synapse) antara akson dan dendrit neuron yang berdekatan. Ketika sebuah neuron mendapatkan rangsangan, neuron tersebut akan mengirim sinyal ke neuron lainnya. Seperti ketika kita tidak sengaja menyentuh panci yang panas, saraf di  tangan kita mengirim sinyal ke saraf lain sampai ke otak dan kita merespon dengan cepat. Pengiriman sinyal antar neuron terjadi sangat cepat dengan kisaran beberapa milidetik saja.

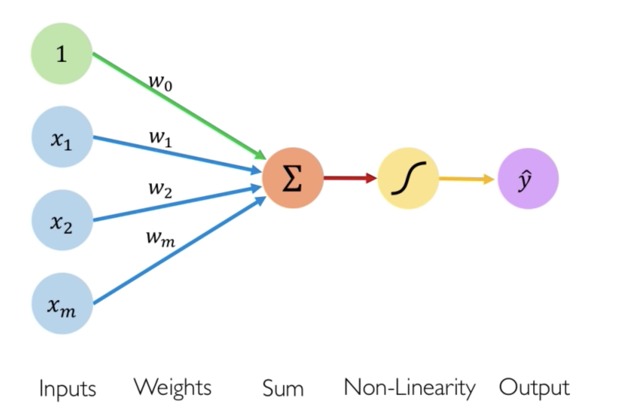
[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Cara kerja dari sebuah neuron terlihat sangat sederhana. Namun, neuron-neuron tersebut terorganisir dalam sebuah jaringan berisi miliaran neuron. Setiap neuron terhubung dengan beberapa ribu neuron lainnya. Dengan jumlah yang luar biasa besar tersebut, banyak pekerjaan kompleks yang dapat diselesaikan.

Nah, kita akan mempelajari neuron tersebut sebagai acuan untuk membuat sebuah jaringan saraf tiruan dalam komputer. Jaringan saraf tiruan ini memiliki komponen dasar bernama perceptron. Frank Rosenblatt dari Cornell Aeronautical Library adalah ilmuwan yang pertama kali menemukan perceptron pada tahun 1957 [8]. Perceptron pada jaringan saraf tiruan terinspirasi dari neuron pada jaringan saraf di otak manusia. Pada jaringan saraf tiruan, perceptron dan neuron merujuk pada hal yang sama.

Lantas bagaimana perceptron bekerja pada jaringan saraf tiruan?

Sebuah perceptron menerima masukan berupa bilangan numerik. Perceptron kemudian memproses masukan tersebut untuk menghasilkan sebuah keluaran. Agar lebih memahami cara kerja perceptron, kita akan menggunakan diagram di bawah.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

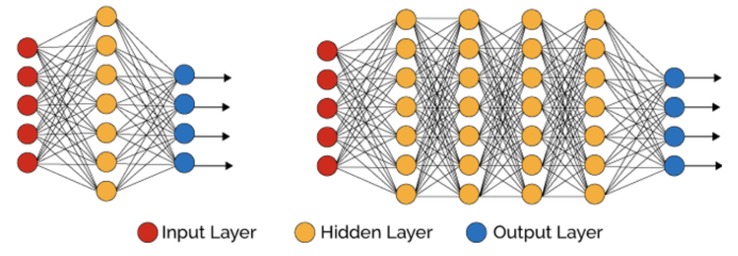
Berikut adalah proses yang menjelaskan bagaimana perceptron bekerja.

* Input menerima masukan berupa angka-angka.
* Setiap input memiliki bobot masing-masing. Bobot adalah parameter yang akan dipelajari oleh sebuah perceptron dan menunjukkan kekuatan node tertentu.
* Selanjutnya adalah tahap penjumlahan input. Pada tahap ini, setiap input akan dikalikan dengan bobotnya masing masing, lalu hasilnya akan ditambahkan dengan bias yang merupakan sebuah konstanta atau angka. Nilai bias memungkinkan Anda untuk mengubah kurva fungsi aktivasi ke atas atau ke bawah sehingga bisa lebih fleksibel dalam meminimalisasi eror. Penjelasan lebih lanjut tentang bias dapat Anda pelajari pada [tautan](https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98) berikut.

Hasil penjumlahan pada tahap ini biasanya disebut weighted sum.

* Berikutnya, aplikasikan weighted sum pada fungsi aktivasi atau disebut juga Non-Linearity Function. Fungsi aktivasi digunakan untuk memetakan nilai yang dihasilkan menjadi nilai yang diperlukan, misalnya antara (0, 1) atau (-1, 1). Fungsi ini memungkinkan perceptron dapat menyesuaikan pola untuk data yang non linier. Penjelasan lebih lanjut tentang fungsi aktivasi akan diulas pada paragraf di bawah.
* Setelah semua langkah di atas terlewati, akhirnya kita memperoleh output berupa hasil perhitungan sebuah perceptron dalam bentuk bilangan numerik.

Setelah mengetahui cara kerja perceptron, kita akan membahas tentang hidden layer. Sebuah hidden layer adalah dense layer yang berada di antara input layer dan output layer. Perhatikan ilustrasi berikut yang menunjukkan jaringan saraf di sebelah kiri memiliki 1 hidden layer, sedangkan jaringan saraf di sebelah kanan memiliki 4 buah hidden layer.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Dalam sebuah jaringan saraf tiruan, input layer dan output layer harus selalu ada, tetapi untuk hidden layer bisa ada beberapa atau tidak sama sekali. Hidden layer dan output layer sama-sama merupakan sebuah layer yang memiliki beberapa perceptron, sedangkan input layer adalah sebuah layer yang hanya menampung angka-angka.

Hidden layer diberikan nama hidden karena sifatnya yang tersembunyi. Pada sebuah sistem jaringan saraf, input dan output layer merupakan lapisan yang dapat kita amati, sementara hidden layer, tidak.

Pada sebuah jaringan saraf tiruan, semakin banyak jumlah hidden layer dalam sistem, semakin lama jaringan saraf tersebut memproduksi hasil, tetapi juga semakin kompleks masalah yang dapat diselesaikan.

**Gambaran Penerapan Deep Learning di Industri**

Deep learning sebagai bagian dari keilmuan machine learning sudah diimplementasikan pada berbagai bidang industri saat ini. Jika Anda ingat kembali pada materi data untuk AI disebutkan bahwa data tidak terstruktur dapat digunakan untuk membangun sebuah sistem berbasis AI. Untuk mengolahnya, kita dapat menggunakan model deep learning untuk mempelajari data tersebut sehingga dapat melakukan tugas berdasarkan data dan permasalahan yang dialami oleh developer.

Sebelum era deep learning, pendekatan yang dilakukan untuk tugas klasifikasi gambar melibatkan tim expert (ahli) yang menjelaskan bagaimana karakteristik atau fitur hewan tertentu. Misal, seekor burung memiliki sayap, bentuk paruh, dan bentuk kaki tertentu.

Kemudian, kita merancang filter pemrosesan gambar agar sesuai dengan kriteria ini. Misalnya, fitur edge detection untuk mengidentifikasi bentuk paruh burung dan mencocokkan fitur morfologi untuk melihat apakah hewan tersebut cocok dengan bentuk yang kita harapkan. Beberapa teknik pemrosesan gambar lain juga dibutuhkan dalam proses ini. Selanjutnya, berdasarkan beberapa kriteria, kita akan mendapatkan skor untuk menilai apakah gambar tersebut adalah seekor burung.

Sebuah proses yang panjang, bukan? Kita juga melakukan hal yang sama saat merancang dan menerapkan serangkaian aturan berbeda untuk hewan-hewan lainnya. Bayangkan usaha yang harus dilakukan untuk dapat menyelesaikan satu tugas klasifikasi. Tentunya sangat menantang, ‘kan?

Dengan mempelajari konsep deep learning ini akan mempermudah Anda untuk membangun sebuah sistem berbasis AI. Agar lebih paham terkait pengembangannya, mari kita simak contoh penerapan deep learning yang akan dibahas di bawah ini, mulai dari pengolahan citra, text, hingga audio.

**Pengolahan Citra/Custom Vision**

Mari kita awali dengan sebuah cerita. Bayangkan Anda sedang duduk di tepi pantai, menikmati senja dan mengamati segala hal yang terjadi di lingkungan sekitar. Di hadapan Anda, laut membentang, ombak berkejaran, sekelompok camar terbang rendah, sesekali mematuk ikan kecil yang muncul ke permukaan. Jauh di ufuk sana, matahari perlahan tenggelam, meninggalkan rona merah keemasan.

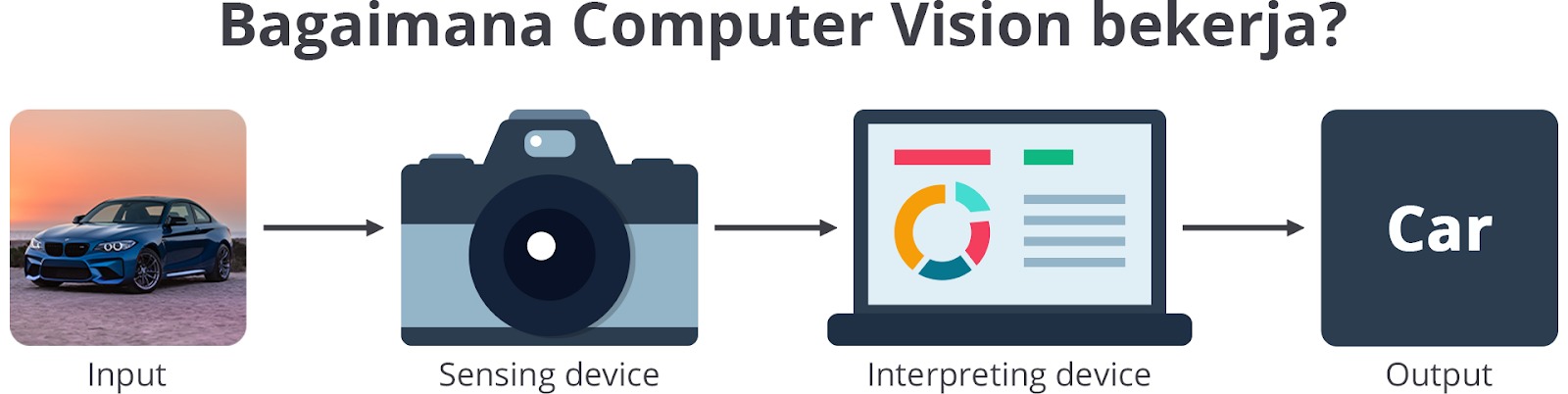
Saat Anda berada di sana dan menyaksikan itu semua, sesungguhnya ada dua sistem dalam tubuh yang sedang bekerja, yaitu mata dan sistem kognitif dalam tubuh. Mata berfungsi sebagai sensor yang menciptakan representasi pemandangan, sedangkan sistem kognitif yang akan memahami apa yang dilihat oleh mata.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Mata manusia dapat melihat pemandangan di sekelilingnya sekaligus mampu melakukan penyesuaian secara dinamis. Saat fokus pada daerah dengan tingkat kecerahan yang bervariasi, mata bisa mengimbangi. Selain itu, mata mampu mencakup sudut pandang yang lebih luas dan fokus pada objek dengan berbagai jarak secara bergantian. Kemampuan mata kita luar biasa, bukan?

Kemudian sistem kognitif memproses semua informasi yang dilihat oleh mata dan merepresentasikan ke dalam berbagai bentuk, warna, kecerahan, detil, dan keindahan. Inilah yang disebut sebagai “vision” atau sistem penglihatan manusia. Kemampuan penglihatan manusia (human vision capabilities) ini kemudian ditiru oleh komputer atau mesin sehingga lahirlah bidang computer vision.

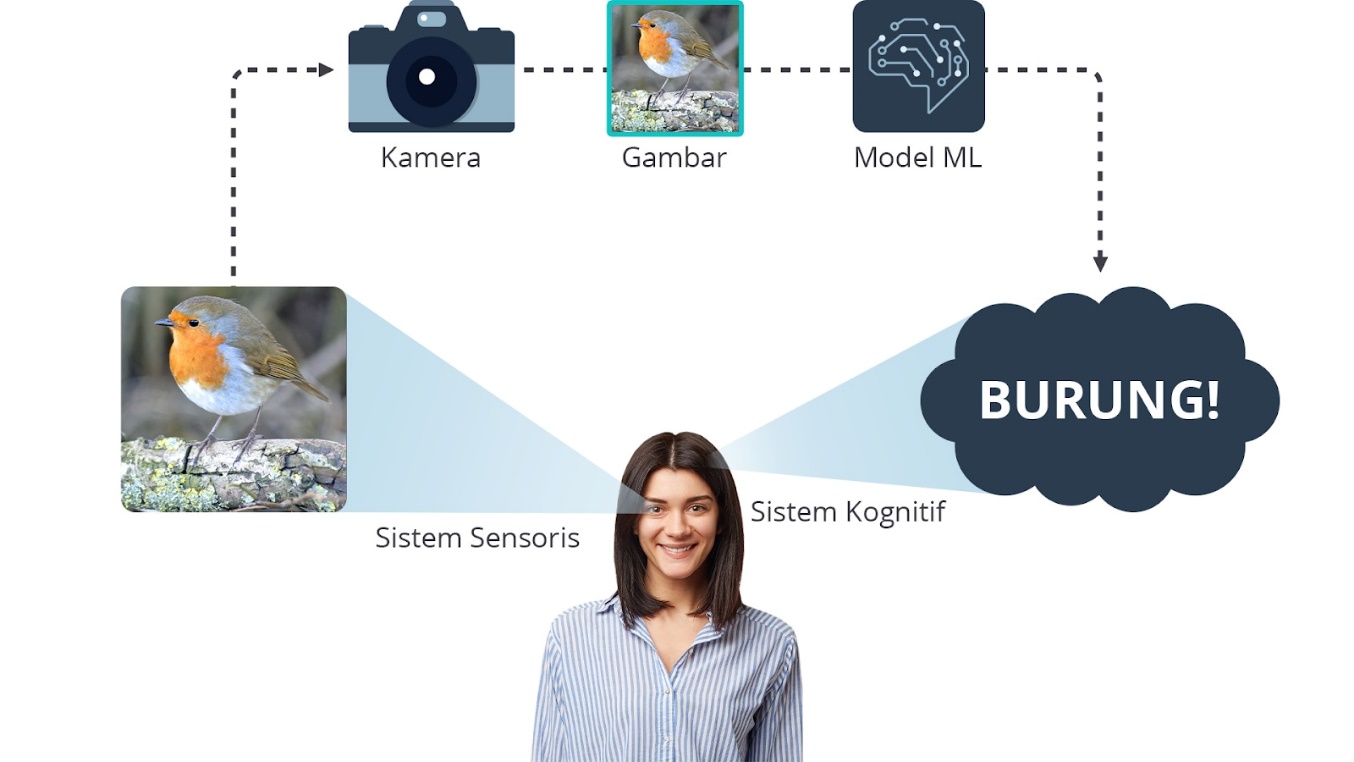
Bagaimana caranya semua itu bisa dilakukan? Computer vision bekerja dengan membangun metode untuk pembentukan gambar (meniru sistem sensorik manusia) dan persepsi mesin (meniru sistem kognitif manusia). Sistem sensorik manusia ditiru dengan melibatkan sistem sensor, seperti kamera, pengaturan desain, dan penempatannya; sedangkan pendekatan modern untuk meniru sistem kognitif manusia terdiri dari metode machine learning yang digunakan untuk mengekstrak informasi dari gambar,

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Jadi, computer vision merupakan bidang yang memungkinkan komputer atau sistem memperoleh informasi dari gambar digital, video, dan input visual lainnya. Bagi manusia dengan sistem penglihatan yang normal, tentu mudah untuk dapat langsung menginterpretasikan objek yang dilihatnya. Sayangnya, hal tersebut akan sulit dilakukan oleh mesin. Mesin membutuhkan sistem pemrosesan paralel yang mumpuni untuk memproses data dalam jumlah besar dan sistem kontrol yang kompleks untuk menyelesaikan permasalahan. Hal ini menjadikan computer vision sebagai salah satu bidang yang rumit sekaligus menantang.

Jika Anda mempelajari computer vision satu dekade ke belakang, metode yang digunakan untuk mengekstrak informasi dari objek digital tidaklah melibatkan machine learning. Pada masa itu, beberapa metode yang digunakan antara lain tentang denoising, edge detection, texture detection, dan morphological operation (shape-based). Kemajuan di bidang AI dan machine learning telah mengubah cara ini. Sebagai contoh, klasifikasi gambar sekarang bisa diselesaikan dengan pendekatan deep learning, yaitu teknik Convolutional Neural Network.

Pendekatan deep learning mengajarkan komputer untuk mengenali suatu objek dalam gambar dengan memberikan banyak data gambar beserta labelnya (solusi yang benar). Misalnya, pada permasalahan klasifikasi hewan, kita akan menunjukkan kepada komputer banyak gambar hewan seperti burung, kelinci, kucing, dan sebagainya. Dengan data pelatihan seperti itu, komputer belajar cara mengklasifikasikan gambar yang belum pernah ditemui sebelumnya.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Dengan cara di atas, proses pengenalan dan klasifikasi gambar tentu jadi lebih mudah dan cepat.

Anda mungkin bertanya-tanya, apa contoh penerapan computer vision di industri saat ini? Contohnya ada banyak sekali, tetapi pada modul ini kita akan membahas salah satu contoh penerapannya di bidang kesehatan.

Kemajuan computer vision di bidang kesehatan memungkinkan para tenaga medis menggunakan data citra medis (medical imaging data) untuk membantu memberikan diagnosis, pengobatan, dan prediksi penyakit. Dengan computer vision, tenaga medis dapat menafsirkan citra sinar-X, CT scan, MRI, dan citra mikroskopis secara lebih akurat. Meskipun computer vision tidak akan sepenuhnya menggantikan tenaga kesehatan, ia cukup berperan dalam melengkapi diagnosis untuk pasien.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Penerapan computer vision dalam pencitraan medis juga menghasilkan solusi untuk memprediksi dan menganalisis gejala gangguan pada gigi, neurologis dan neuropathy, memantau kehilangan darah untuk mengoptimalkan transfusi, mendeteksi COVID-19, dan lain sebagainya. COVID-Net dengan teknik deep neural network menunjukkan akurasi 90% dalam mendiagnosis COVID-19 berdasarkan gambar rontgen dada.

Selain itu, sebuah perusahaan bernama Gauss Surgical memproduksi alat untuk memonitor darah secara real-time. Monitor dilengkapi dengan aplikasi sederhana dengan teknik computer vision berbasis cloud. Alat ini bertujuan untuk memprediksi banyaknya darah yang hilang selama operasi, memaksimalkan proses transfusi darah, dan mendeteksi potensi pendarahan secara lebih akurat. Teknologi ini diperkirakan dapat menghemat sekitar 10 miliar dollar setiap tahunnya.

**Pengolahan Text**

Bayangkan Anda terbangun di pagi hari. Dengan mata masih setengah terpejam, Anda mengaktifkan perangkat cerdas di sisi tempat tidur kemudian bertanya kepadanya.

Anda: “*Jam berapa sekarang?*”

Asisten digital: “*Sekarang pukul enam pagi*”

Tidak seperti biasanya, pagi ini Anda merasa sangat kedinginan. Kemudian, sambil menarik selimut, Anda bertanya lagi.

Anda : “*Berapa suhu saat ini?*”

Asisten digital : “*Sekarang, suhu di Bandung 14 derajat celcius*”

Dengan suhu sedingin ini, rasanya enggan untuk beranjak dari tempat tidur. Melanjutkan tidur sambil meringkuk di dalam selimut yang hangat sepertinya ide bagus. Namun, Anda kemudian ingat ada janji temu klien hari ini tapi entah jam berapa. Anda lantas bertanya lagi.

Anda: “*Apa saja jadwalku hari ini?*”

Asisten digital menjawab dengan membacakan seluruh jadwal pada kalender digital Anda hari itu, termasuk jadwal meeting dengan klien pada pukul 7 pagi. Artinya, Anda tidak bisa lanjut rebahan dan harus segera bangkit dari tempat tidur untuk bersiap.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Anda mungkin familier dan pernah menggunakan asisten digital. Bahkan, sebagian dari Anda mungkin telah lama menggunakannya dalam kehidupan sehari-hari. Tentu saja, bahasa yang Anda gunakan saat berkomunikasi dengan asisten digital bukanlah bahasa pemrograman formal seperti Python atau C, melainkan bahasa alami seperti yang Anda gunakan sehari-hari untuk berkomunikasi dengan sesama manusia.

Permasalahannya, komputer hanya dapat memproses data dalam biner. Lalu, bagaimana kita membuat mesin memahami bahasa alami tersebut? Di sinilah bidang Natural Language Processing (Pemrosesan Bahasa Alami) berperan. NLP merupakan subbidang dari Artificial Intelligence (AI) untuk memproses, menganalisis, memahami, dan menghasilkan bahasa manusia. NLP termasuk bidang keilmuan AI karena pemrosesan bahasa dianggap sebagai bagian dari kecerdasan manusia. Penggunaan bahasa merupakan keterampilan paling menonjol yang membedakan manusia dengan makhluk lainnya.

NLP mencakup berbagai algoritma, metode, dan pemecahan masalah yang menggunakan teks sebagai input. NLP menghasilkan beberapa informasi sebagai output, seperti label, representasi semantik, dan sebagainya. Teknik NLP digunakan di setiap aplikasi cerdas yang melibatkan bahasa alami. Ia merupakan komponen penting dalam berbagai aplikasi perangkat lunak yang kita gunakan dalam kehidupan sehari-hari.

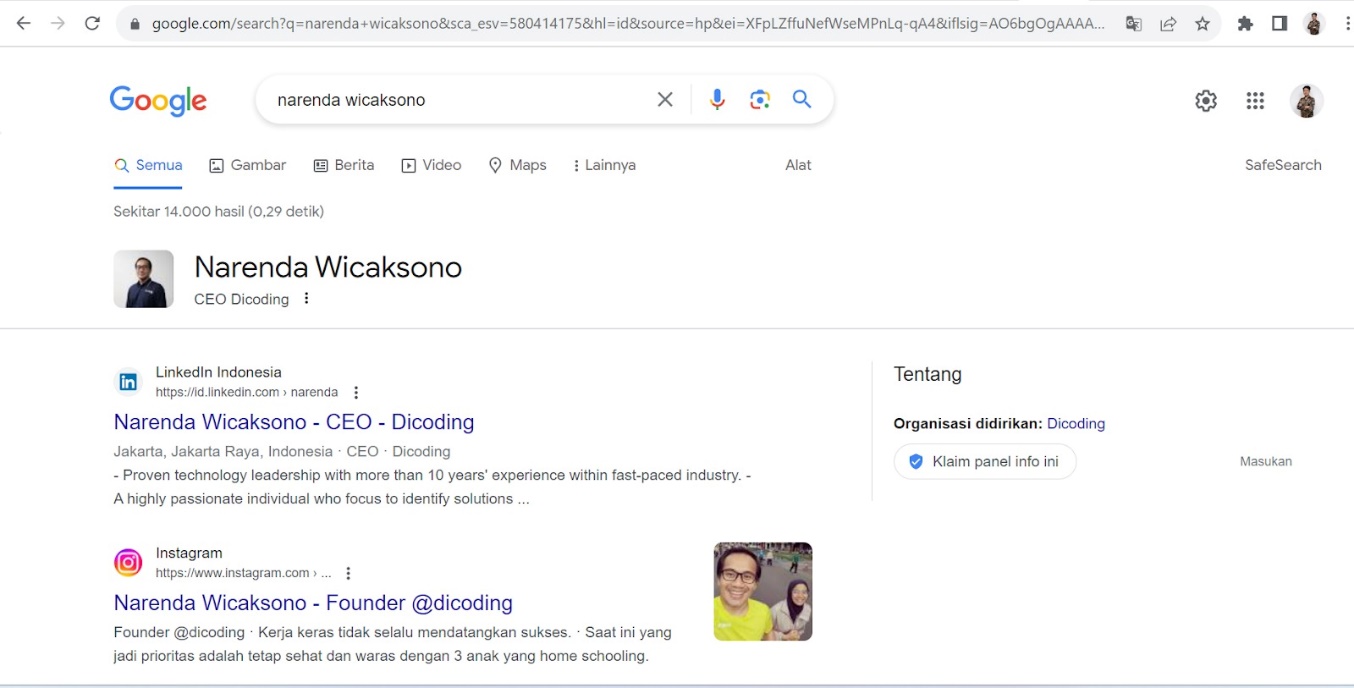
Besarnya peluang pengembangan NLP ini berdampak pada semakin banyaknya perusahaan menerapkan NLP dalam produk maupun bisnisnya. Hal ini berpengaruh pada permintaan akan pakar atau tenaga ahli di bidang NLP. Tentu ini peluang bagus bagi Anda yang tertarik untuk mendalami NLP. Semangat!

Nah, berbicara tentang penerapan NLP, kita bisa menemukan penerapan NLP pada banyak bidang. Sebagai contoh, bidang e-commerce, pendidikan, keuangan, hiburan, perawatan kesehatan, layanan pelanggan, dan marketing.

Pada bidang marketing, salah satu aplikasi NLP yang telah dibahas di materi sebelumnya adalah analisis sentimen untuk social media monitoring. Selain yang telah disebutkan tadi, NLP memiliki peran signifikan pada berbagai produk dan layanan. Salah satu contoh penerapan yang paling populer pada aplikasi NLP adalah mesin pencari.

Siapa yang tidak mengenal mesin pencari? Tempat kita mencari dan bertanya mengenai berbagai hal di internet. Mesin pencari modern seperti Google dan Bing seolah telah menjadi bagian tak terpisahkan dalam kehidupan daring umat manusia. Nah, di balik kecanggihan mesin pencari, ada peran penting NLP dalam berbagai tahap pengembangannya.

Salah satu contohnya adalah pada tahap analisis kueri (*query analysis*). Analisis kueri mengidentifikasi intensi pengguna saat mengetikkan kata kunci pada mesin pencari kemudian memberikan informasi yang relevan. Misalnya, jika kita mengetik nama seorang tokoh terkenal pada mesin pencari, akan muncul berbagai informasi penting yang relevan dengan tokoh tersebut. Selain itu juga muncul beberapa berita mengenai tokoh tersebut.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Selain itu, mesin pencari juga memiliki fungsi koreksi dan rekomendasi kueri. Fungsi ini biasanya muncul saat kita mengetikkan ejaan yang salah dalam mesin pencari. Ketika hal tersebut terjadi, mesin menunjukkan koreksi dengan label seperti “menampilkan hasil untuk (ejaan yang dikoreksi)” atau “telusuri (ejaan salah yang kita ketik)”. Fungsi ini tentu memudahkan proses pencarian dan mengoptimalkan pengalaman pengguna.

Sampai di sini, Anda telah memahami beberapa contoh penerapan dari deep learning, mulai dari data yang berupa citra hingga proses pengolahan bahasa alami. Dengan uraian tersebut, diharapkan Anda dapat memahami cara kerja deep learning dan dapat memiliki ide untuk membuat deep learning berdasarkan permasalahan yang ada di sekitar.

**Ekstensi Model**

Setelah Anda membuat sebuah model machine learning, tentu perlu melakukan export model tersebut menjadi ekstensi tertentu agar dapat digunakan pada platform yang diinginkan. Permasalahan yang muncul saat ini adalah tidak ada ekstensi yang cocok dengan berbagai platform, katakanlah Anda ingin men-deploy model yang Anda buat di web atau mobile device. Anda perlu membuat model machine learning dengan bentuk atau ekstensi tertentu agar bisa di-consume oleh platform tersebut.

Untuk menyelesaikan tersebut teachable machine menyediakan beberapa ekstensi seperti TensorFlow.js dan TensorFlow Lite sebagai pilihan export model machine learning. Pertanyaannya, apa perbedaan dari masing-masing ekstensi yang disediakan oleh teachable machine? Bagaimana cara kita menentukan ekstensi yang cocok untuk proyek yang sedang dibangun?

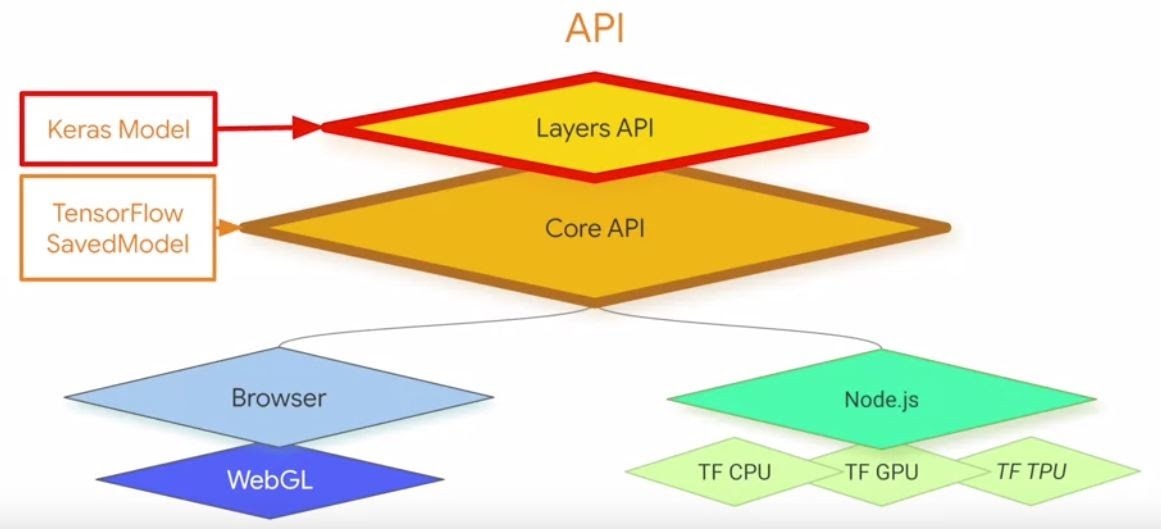
**TensorFlow.js**

TensorFlow.js memungkinkan kita untuk membangun aplikasi machine learning pada Web Browser. Ia adalah sebuah *framework* yang kompatibel dengan TensorFlow API. TensorFlow.js menggunakan model yang telah dibuat dengan mengubah format model menjadi JSON file.

Pada level atas TensorFlow.js API, kita akan dihadapkan dengan Layers API. Jika familier dengan penggunaan layer-layer pada Keras, kita dapat lebih mudah untuk menggunakan Layers API.

Di bawah layers API, ada Core API yang menangani model dari TensorFlow. Core API juga mengimplementasikan operasi graf pada level lebih kompleks, seperti deklarasi tensor (data input), operasi pada tensor, memori, eksekusi fungsi, dan lain-lain.

Selain itu CORE API bekerja dengan browser dan menggunakan WebGL untuk menggunakan resource yang mendukung proses training atau pengambilan hasil prediksi (inference). Pada Node.js, kita bisa membuat aplikasi server-side dan menggunakan resource yang tersedia, seperti CPU, GPU, atau TPU.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Terdapat beberapa keuntungan menggunakan TensorFlow.js seperti memudahkan kita dalam melakukan integrasi dengan teknologi web seperti user interface karena TensorFlow.js dapat ditulis dalam bahasa script. Selain itu, TensorFlow.js memudahkan kita pada beberapa utilitas yang dapat kita import seperti *document object model*(DOM) *canvas* yang memungkinkan kita mendapat data langsung dari input user di internet.

Jika ada kelebihan, pasti ada kekurangan. TensorFlow.js juga memiliki kekurangan, salah satunya yaitu penggunaan web browser sebagai *platform*machine learning menyebabkan permasalahan performa. Web browser biasanya merupakan aplikasi dengan *single* proses dan tidak bekerja intens dengan CPU. Namun, modern web browser sudah menyediakan API yang bisa memanfaatkan lokal hardware akselerator dengan menggunakan WebGL atau WebGPU. Dengan adanya API tersebut, TensorFlow.js dapat memberikan performa yang sama baiknya walaupun menggunakan browser.

Web browser merupakan platform yang cocok untuk men-deploy model dengan ekstensi TensorFlow.js. Selanjutnya, mari kita pelajari proses *deployment*model machine learning pada perangkat *mobile* dan *IoT*.

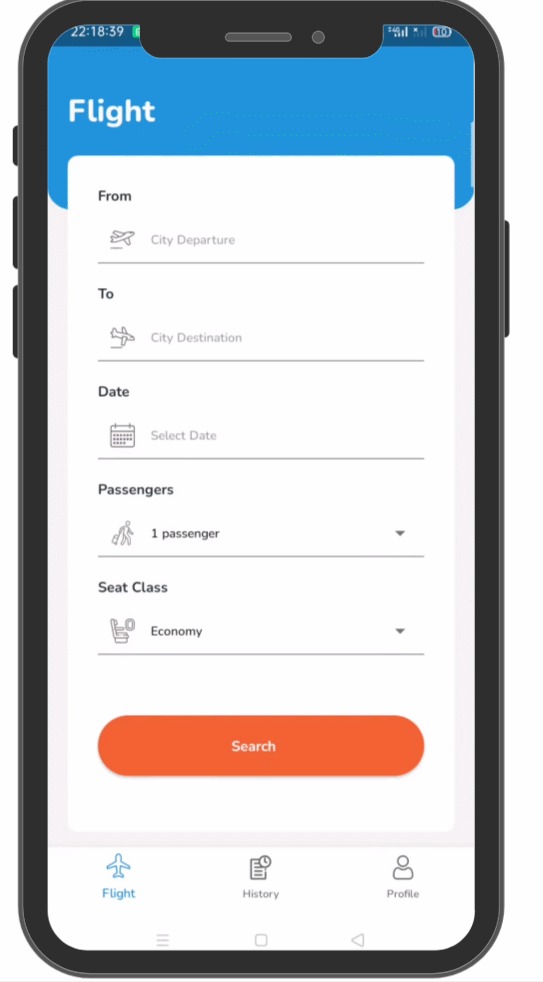
**TensorFlow Lite**

Sebelumnya Anda telah mempelajari proses deployment menggunakan TensorFlow.js yang cocok diimplementasikan pada web browser. Pada bagian ini, kita akan berkenalan dengan TensorFlow Lite. TensorFlow Lite (TF-Lite) merupakan sebuah framework yang dapat menjalankan model TensorFlow pada perangkat mobile dan IoT.

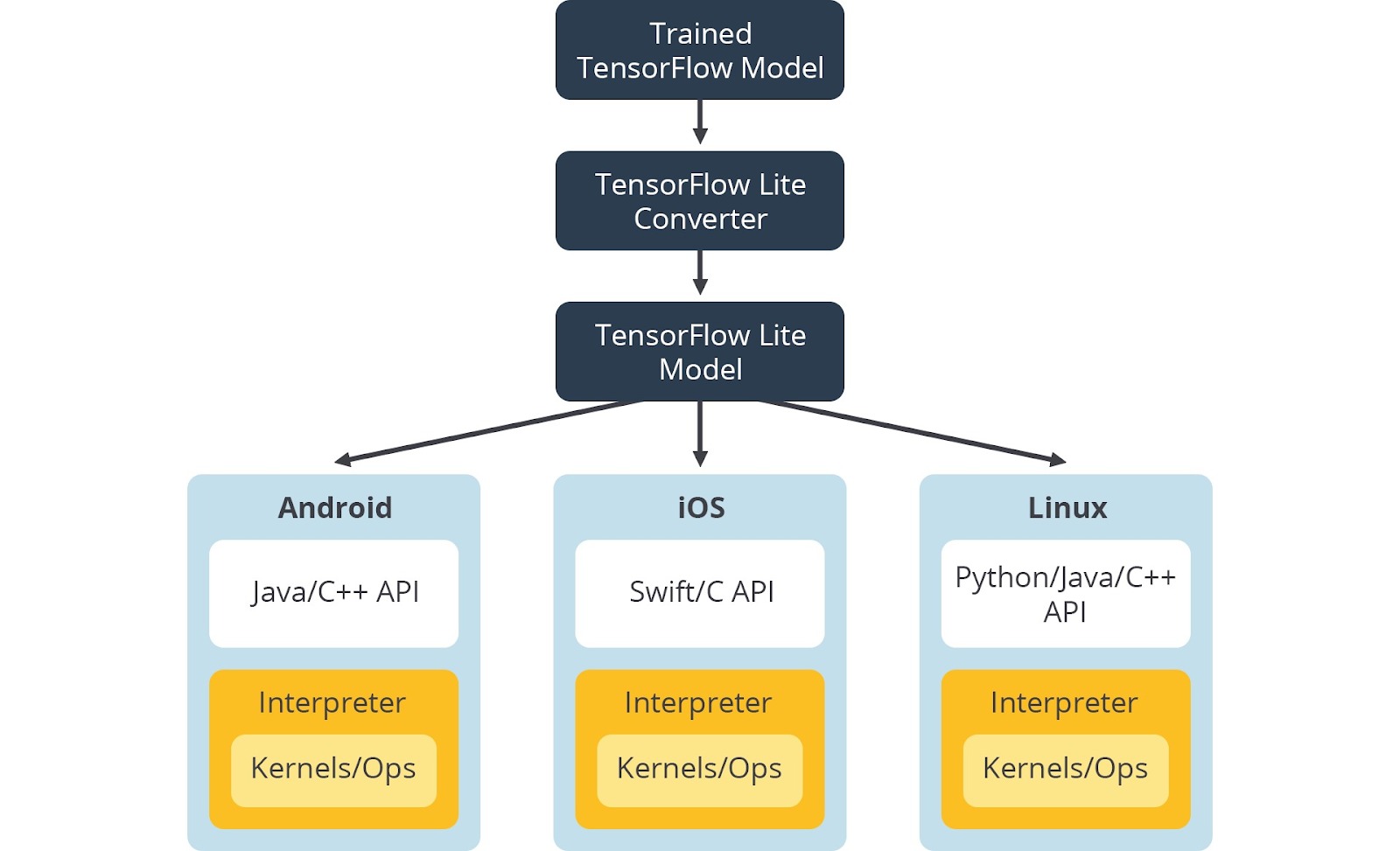
Machine learning pada perangkat mobile dapat memudahkan manusia dalam menyelesaikan tugas sehari-hari. Selain itu, implementasi model machine learning pada perangkat *mobile* menggunakan TF-Lite juga memiliki beberapa keuntungan seperti berikut.

1. Penggunaan TF-Lite tidak memerlukan server sehingga perangkat tidak harus terhubung ke internet untuk melakukan prediksi dan mampu menjaga privasi pengguna.
2. Memiliki Latency dan ukuran binary yang kecil sehingga dapat mengurangi konsumsi daya ketika melakukan prediksi.

Dengan adanya machine learning pada perangkat *mobile,* kita dapat melakukan beragam aktivitas dengan lebih mudah dan efisien. Agar semakin paham, mari kita bahas beberapa contoh penggunaan machine learning pada perangkat *mobile*.

1. Traveloka OCR merupakan salah satu contoh penerapan machine learning yang dapat mengenali identitas seseorang berdasarkan foto KTPnya. Fitur ini dapat membantu kita dalam mengisi identitas pada platform Traveloka agar mempercepat proses pengisian data dan menghindari kesalahan manusia pada proses pengisian secara manual.    
   [](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)
2. Google Translate Instant Camera Translation yang dapat menerjemahkan teks bahasa asing dengan foto. Fitur ini dapat membantu turis mancanegara dalam memahami bahasa asing ketika berlibur tanpa bantuan penerjemah sehingga mereka tidak perlu khawatir ketika berlibur ke negara asing.  
   [](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Sebelum menjalankan model TensorFlow pada perangkat mobile, kita perlu memahami arsitektur TF-Lite terlebih dahulu. Berikut merupakan ilustrasi sederhana arsitektur TF-Lite.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Setelah melatih model menggunakan teachable machine, Anda bisa memilih export model dengan pilihan “TensorFlow Lite”. Sekadar informasi, TF-Lite juga dilengkapi dengan API untuk berbagai macam bahasa pemrograman, seperti C, C++, Swift, Java, dan Python. Menarik, ‘kan?

**Proses di Balik Deep Learning**

**Pembelajaran yang Dilakukan Komputer**

Di balik proses pembelajaran yang dilakukan oleh komputer terdapat banyak sekali perhitungan matematis yang terjadi. Mari kita bahas satu per satu mulai dari input layer, hidden layer, hingga output layer.

Kita akan menggunakan studi kasus klasifikasi gambar fashion menggunakan MNIST dataset untuk mempelajari cara kerja dari setiap layer. Dataset tersebut memiliki 10 buah kelas yang terdiri dari baju, sepatu, tas, dan sebagainya.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Jika tidak menggunakan teachable machine, kita harus menggunakan TensorFlow untuk membuat model machine learning tersebut. Kita dapat menggunakan Keras dengan kode sebagai berikut.

1. mnist = tf.keras.datasets.fashion\_mnist
2. (x\_train, y\_train),(x\_test, y\_test) = mnist.load\_data()
3. X\_train, x\_test = x\_train / 255.0, x\_test / 255.0
4. model = tf.keras.models.Sequential([
5. tf.keras.layers.Flatten(input\_shape=(28, 28)),
6. tf.keras.layers.Dense(512, activation=tf.nn.relu),
7. tf.keras.layers.Dense(10, activation=tf.nn.softmax)
8. ])
9. model.compile(optimizer=tf.optimizers.Adam()),
10. loss=’sparse\_categorical\_crossentropy’,
11. metrics=[‘accuracy’])
12. model.fit(x\_train, y\_train, epochs=10)

Mari kita bahas kode di atas satu per satu. Hal yang paling pertama adalah kita perlu mempersiapkan data kemudian membaginya menjadi data latih dan data uji. Data fashion MNIST bisa kita dapatkan dengan mudah dari library datasets yang disediakan Keras.

1. Mnist = tf.keras.datasets.fashion\_mnist
2. (x\_train, y\_train),(x\_test, y\_test) = mnist.load\_data()
3. X\_train, x\_test = x\_train / 255.0, x\_test / 255.0

Dalam klasifikasi gambar, setiap piksel pada gambar memiliki nilai dari 0 sampai 255. Kita perlu melakukan normalisasi dengan membagi setiap pixel pada gambar dengan 255. Dengan nilai yang telah dinormalisasi, jaringan saraf dapat belajar dengan lebih baik.

1. X\_train, x\_test = x\_train / 255.0, x\_test / 255.0

Pada langkah berikutnya, kita mendefinisikan arsitektur dari jaringan saraf yang akan kita latih.

1. model = tf.keras.models.Sequential([
2. tf.keras.layers.Flatten(input\_shape=(28, 28)),
3. tf.keras.layers.Dense(512, activation=tf.nn.relu),
4. tf.keras.layers.Dense(10, activation=tf.nn.softmax)
5. ])

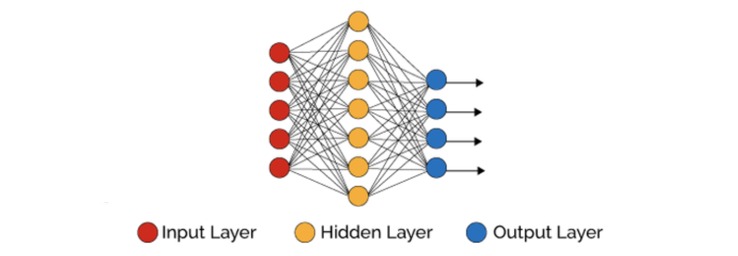
Seperti yang Anda lihat pada kode di atas, ketiga layer tersebut terdiri dari input layer, hidden layer, dan output layer. Apa arti dari masing-masing layer tersebut? Mari kita bedah semuanya agar Anda dapat lebih memahami proses pembelajaran yang terjadi pada machine learning.

Untuk membuat sebuah model di Keras, kita bisa memanggil fungsi tf.keras.models.Sequential([...]) dan menampungnya pada sebuah variabel. Model sequential pada Keras adalah tumpukan layer-layer yang melakukan proses perhitungan sehingga mendapatkan output yang sesuai dengan tugasnya.

Berikut definisi dari 3 layer utama pada model sequential.

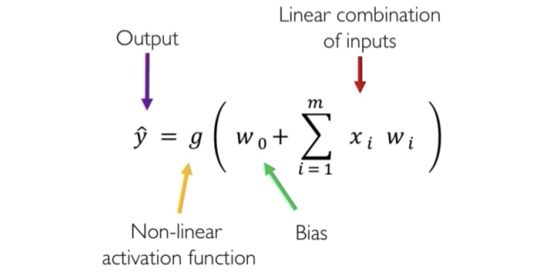
* **Input layer**  
  Layer yang memiliki parameter ‘input\_shape’. Input\_shape sendiri adalah resolusi dari gambar-gambar pada data latih. Dalam hal ini sebuah gambar MNIST memiliki resolusi 28x28 pixel sehingga input shape-nya adalah (28, 28). Sebuah layer Flatten pada Keras akan berfungsi untuk meratakan input. Meratakan di sini artinya mengubah gambar yang merupakan matriks 2 dimensi menjadi array 1 dimensi. Pada kasus kita sebuah gambar MNIST yang merupakan matriks 28x 28 elemen akan diubah menjadi larik/array satu dimensi sebesar 784 elemen.
* **Hidden layer**  
  Dense layer pada Keras merupakan layer yang dapat dipakai sebagai hidden layer dan output layer pada sebuah MLP. Parameter unit merupakan jumlah perceptron pada sebuah layer. Kita dapat menggunakan fungsi aktivasi relu (rectified linear unit) atau fungsi aktivasi lain untuk hidden layer kita.
* **Output layer**  
  Ia didefinisikan dengan membuat sebuah Dense layer. Jumlah unit menyesuaikan dengan jumlah label pada dataset. Untuk fungsi aktivasi pada layer output, gunakan fungsi aktivasi Sigmoid ketika hanya terdapat 2 kelas/label pada dataset. Untuk dataset yang memiliki 3 kelas atau lebih, gunakan fungsi aktivasi Softmax. Fungsi aktivasi softmax akan memilih kelas mana yang memiliki probabilitas tertinggi. Untuk data fashion MNIST kita akan menggunakan fungsi aktivasi softmax karena terdapat 10 kelas.

Agar Anda lebih memahami proses terjadinya pembelajaran machine learning, perhatikan gambar di bawah ini.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Gambar di atas menunjukkan jumlah layer yang direpresentasikan oleh neuron. Jika kalian ingat, proses pembelajaran di atas sama halnya dengan perceptron yang telah kita bahas pada modul ini. Setiap neuron yang terhubung mewakili perhitungan dari sebuah perceptron. Proses pembelajaran ini terdiri dari 5 komponen, yaitu bobot atau *weights*(Wi) dan *bias*(W0), penjumlahan atau *sum*(∑), fungsi aktivasi atau *non linearity function* (⎰), dan *output*(y).

Fungsi matematis dari proses pembelajaran ini dapat kita lihat di bawah ini.

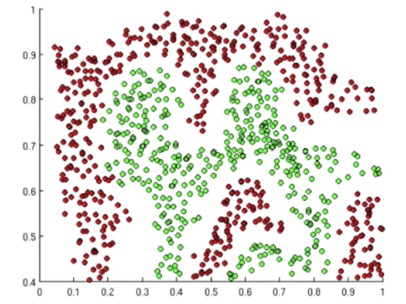
[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Rumus tersebut merupakan notasi matematis yang menjelaskan proses yang kita bahas sebelumnya. Berikut merupakan keterangan dari rumus di atas:

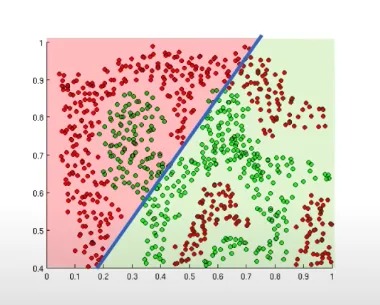
* ŷ = output/keluaran
* w0= bias
* w1= weight/bobot
* x1= input
* g = fungsi aktivasi

Sehingga dari rumus di atas dapat disimpulkan bahwa output/keluaran (ŷ) merupakan hasil perhitungan dari bias (w0) ditambah dengan total dari input (x1) dikalikan dengan bobot (w1). Kemudian, nilai akhir dari perhitungan tersebut dikalikan dengan fungsi aktivasi (g) sehingga mendapatkan sebuah nilai output/keluaran

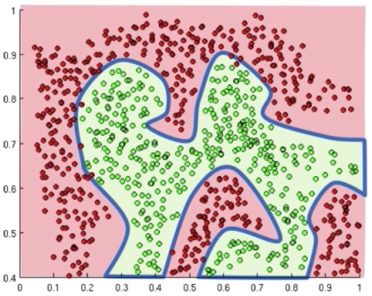
Fungsi aktivasi (g) pada perceptron bertugas untuk membuat jaringan saraf mampu menyesuaikan pola pada data non linier. Seperti yang sudah pernah dibahas sebelumnya, mayoritas data yang terdapat di dunia nyata adalah data non linier seperti di bawah ini.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Tanpa fungsi aktivasi, jaringan saraf hanya bisa mengenali pola linier seperti garis pada regresi linier. Hasilnya bisa Anda lihat melalui gambar berikut.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Fungsi aktivasi (g) itulah yang memungkinkan jaringan saraf dapat mengenali pola non-linier tanpa memperhatikan kompleksitas atau sebaran data yang ada. Sehingga dengan penggunaan fungsi aktivasi (g) menghasilkan batasan sebaran data sebagai berikut.

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Setelah membuat arsitektur machine learning, model kita belum bisa melakukan apa pun. Agar model bisa belajar, kita perlu memanggil fungsi compile pada model dan menspesifikasikan optimizer dan loss function. Dua hal ini memiliki peran yang sangat penting ketika kita membangun model machine learning. Loss function berfungsi untuk menghitung perbedaan antara output prediksi dengan output sesungguhnya, sedangkan optimizer berfungsi untuk menyesuaikan nilai parameter sehingga dapat meminimalisir kesalahan pada saat pembelajaran dilakukan.

Salah satu optimizer yang biasa digunakan adalah Adam. Selanjutnya untuk loss function kita dapat menggunakan sparse categorical entropy pada kasus klasifikasi 3 kelas atau lebih. Untuk masalah 2 kelas, loss function yang lebih tepat adalah binary cross entropy. Parameter metrics berfungsi untuk menampilkan metrik yang dipilih pada proses pelatihan model.

1. model.compile(optimizer=tf.optimizers.Adam()),
2. loss=’sparse\_categorical\_crossentropy’,
3. metrics=[‘accuracy’])

Setelah membuat arsitektur MLP dan menentukan optimizer serta loss functionnya, kita dapat melatih model kita pada data training. Parameter epoch merupakan jumlah berapa kali sebuah model melakukan propagasi balik.

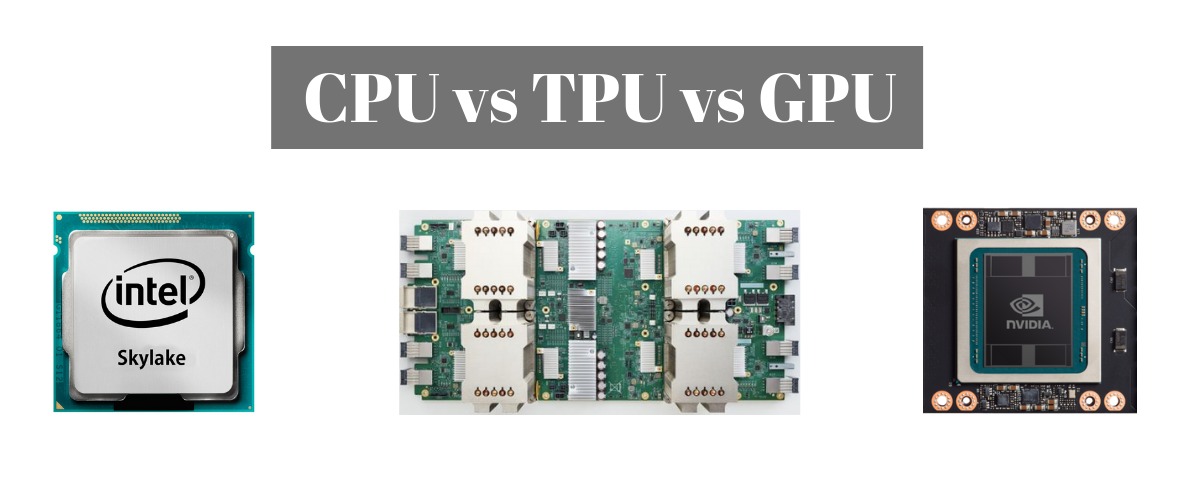
1. model.fit(x\_train, y\_train, epochs=10)

Begitulah cara kita membuat sebuah model deep learning dengan tanpa bantuan teachable machine. Tentu sangat menyenangkan, bukan? Namun, Anda tidak perlu risau melihat proses pembuatan machine learning tersebut karena sejatinya tugas tersebut akan dikelola oleh machine learning engineer.

**Perangkat Keras yang Digunakan**

Anda telah mempelajari banyak hal terkait pembangunan AI hingga deep learning, tetapi tahukah Anda bagaimana model yang kita buat dapat melakukan tugasnya seperti mengklasifikasikan kelompok umur, atau memprediksi sebuah gambar? Komputer memerlukan perangkat keras untuk menjalankan model machine learning karena tugas-tugas yang dilakukan olehnya seringkali melakukan komputasi yang berat dan memerlukan operasi matematika yang intensif.

Ada beberapa perangkat keras yang dapat mendukung komputer untuk menjalankan model machine learning beberapa contohnya seperti *Central Processing Unit*(CPU), *Graphical Processing Unit*(GPU), dan *Tensor Processing Unit* (TPU).

[](https://www.dicoding.com/academies/653/tutorials/35498?from=35493)

Mari kita bahas apa perbedaannya dan kelebihan dari masing-masing perangkat keras tersebut. Materi ini dibuat agar Anda dapat menentukan perangkat keras yang cocok untuk menjalankan model machine learning yang telah dibuat sebelumnya.

Saat kami menyebutkan beberapa contoh perangkat keras yang dapat membantu menjalankan model machine learning, mungkin Anda akan bertanya “bagaimana cara menentukan perangkat keras yang cocok untuk model yang telah dibuat?” Pertanyaan yang bagus!

Sebetulnya, jika menjalankan machine learning pada komputer lokal dan memiliki perangkat keras yang mendukung, Anda tidak perlu memikirkan hal ini karena GPU akan secara otomatis melakukan segala komputasinya. Namun, permasalahan muncul ketika Anda akan menjalankan model machine learning pada mesin yang terbatas atau ketika Anda akan menyimpan model pada sebuah layanan cloud computing.

Mengapa ketika kita menyimpan model machine learning pada layanan cloud computing harus memperhatikan perangkat keras yang digunakan? Hal ini disebabkan mayoritas layanan tersebut memiliki batasan spesifikasi sehingga mempengaruhi biaya yang kita keluarkan untuk menjalankan model machine learning.

Mari kita kembali ke pertanyaan utama kita, “Bagaimana cara menentukan perangkat keras yang cocok untuk model yang telah dibuat?” Jika Anda membuat machine learning yang memerlukan pelatihan model yang sangat intensif secara komputasi seperti pelatihan jaringan saraf tiruan yang besar, GPU atau TPU bisa menjadi pilihan yang lebih baik daripada CPU. Namun, ketika Anda memiliki model machine learning yang lebih sederhana tanpa melibatkan jaringan saraf seperti klasifikasi menggunakan decision tree, k-means clustering, dan lain sebagainya, Anda dapat menggunakan CPU.

Sampai di sini, bagaimana menurut Anda mengenai pembangunan machine learning ini? Proses pengembangan machine learning tidak sebatas membuat model saja karena kita harus mempelajari hingga model tersebut dapat mengerjakan tugasnya. Berita baiknya, Anda sudah mempelajari semua itu, sekaligus mengetahui perangkat keras yang dapat digunakan.