DASAR-DASAR DEEP LEARNING DAN CONTOH APLIKASINYA

Perkembangan ilmu pengetahuan dan teknologi dalam beberapa tahun terakhir sangat cepat, khususnya ilmu pada bidang AI (*Artificial Inteligence*). Salah satunya adalah metode yang ada pada AI yaitu *Deep Learning*. Deep learning merupakan subbidang *machine learning* yang algoritmanya terinspirasi dari struktur otak manusia. Struktur tersebut dinamakan *Artificial Neural Networks* atau disingkat ANN. Pada dasarnya, ia merupakan jaringan saraf yang memiliki tiga atau lebih lapisan ANN.

Dalam buku ini, akan dijelaskan dasar-dasar tentang metode Deep Learning serta contoh penerapan aplikasi Deep Learning dengan beberapa pemrograman. Buku diawali dengan pembahasan Konsep AI secara umum, metode *Deep Learning*, Algoritma *Deep Learning*, Arsitektur CNN, Penge-nalan *Jetson Nano*, dan contoh aplikasi penerapan Deep Learning. Buku ini juga membahas penerapan Deep Learning menggunakan salah satu perangkat yaitu NVIDIA *Jetson Nano*, yang cocok digunakan untuk para pengembang aplikasi AI. Dengan NVIDIA *Jetson Nano* ini, tidak hanya perusahaan besar, pengembang perorangan, peneliti, bahkan pelajar di sekolah maupun perguruan tinggi bisa mengembangkan perangkat pintar dengan lebih mudah.

Penerbit Mitra Cendekia Media

IKAPI

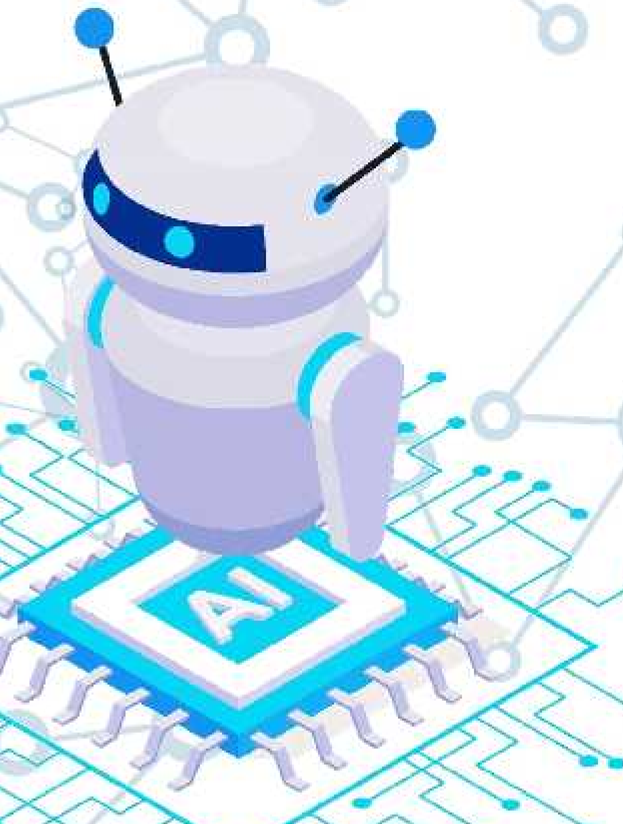
PENDIDIKAN

DASAR-DASAR DEEP LEARNING DAN CONTOH APLIKASINYA

Emil Naf'an, S.Kom., M.Kom.,dkk.

Emil Naf'an, S.Kom., M.Kom.,
Fajrul Islami, S.Kom., M.Kom.,
Gushelmi, S.Kom., M.Kom.

DASAR-DASAR DEEP LEARNING DAN CONTOH APLIKASINYA



DASAR-DASAR
DEEP LEARNING
dan Contoh Aplikasinya

UU No 28 Tahun 2014 tentang Hak Cipta

Fungsi dan Sifat Hak Cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

DASAR-DASAR
DEEP LEARNING
dan Contoh Aplikasinya

Emil Naf'an, S.Kom., M.Kom.
Fajrul Islami, S.Kom., M.Kom.
Gushelmi, S.Kom., M.Kom.

CV. MITRA CENDEKIA MEDIA

DASAR-DASAR DEEP LEARNING
DAN CONTOH APLIKASINYA

Emil Naf'an, S.Kom., M.Kom, Fajrul Islami, S.Kom., M.Kom
Gushelmi, S.Kom., M.Kom

Editor:
Rendi Fernandes

Desainer:
Mifta Ardila

Sumber Gambar Kover:
Freepik.com

Penata Letak:
Rendi Fernandes

Proofreader:
TIM MCM

Ukuran:
xvi, 105 hlm., 14,8x21 cm

ISBN :
000-000-00000-0-0

Cetakan Pertama:
Juli 2022

Anggota IKAPI: 022/SBA/20
PENERBIT MITRA CENDEKIA MEDIA)

Kapalo Koto No. 8, Selayo, Kec. Kubung, Kab. Solok
Sumatra Barat – Indonesia 27361
HP/WA: 0812-7574-0738
Website: www.mitracendekiamedia.com
E-mail: mitracendekiamedia@gmail.com

DAFTAR ISI

Daftar Gambar

Daftar Tabel

PRAKATA

Deep learning merupakan subbidang machine learning yang algoritmanya terinspirasi dari struktur otak manusia. Saat ini, teknik deep learning sangat populer di kalangan praktisi data dan menarik perhatian banyak pihak. Hal ini karena teknologi *deep learning* telah diterapkan dalam berbagai produk berteknologi tinggi seperti *self-driving* car. Selain itu, ia juga ada di balik produk dan layanan yang kita gunakan sehari-hari. Contohnya antara lain, asisten digital, Google Translate, dan *voice-activated device* (perangkat cerdas yang bisa diaktifkan dengan suara).

Pada buku ini kita akan membahas mengenai deep learning mulai dari dasar-dasar *Deep Learning* serta contoh aplikasi Deep learning. Adapun BAB pembahasan dalam buku yaitu:

BAB I : Pada BAB akan dijelaskan teori-teori yang berkaitan dengan *Deep Learning*. Mulai dari pengenalan *Artificial Intelligence* dan teori *Deep Learning* secara umum.

BAB II : BAB ini berisi penjelasan tentang salah satu model *Deep Learning* yang sedang populer saat ini yaitu metode *Teachable Machine*.

BAB III : Pada BAB ini dijelaskan satu perangkat yang bisa digunakan untuk penerapan pengembangan Aplikasi dengan AI, yaitu perangkat Jetson Nano.

BAB IV : Contoh Aplikasi *Teachable Machine* akan dijelaskan pada BAB IV ini. Dimana contoh-contoh penerapan meng-gunakan *Teachable Machine* akan diujikan disini.

BAB V : Ini merupakan BAB terakhir yang dibahas pada buku, yaitu contoh-contoh aplikasi *Deep Learning* menggunakan Jetson Nano akan dijelaskan pada BAB ini.

Buku ini disajikan secara efektif dan efisien, sehingga penerapan contoh aplikasi langsung bisa dengan mudah dipahami oleh pembaca. Semoga buku ini bisa menjadi rujukan yang tepat dan praktis dalam mempelajari dan mendalami konsep tentang *Deep Learning*.

Padang, 22 Juni 2022

Penulis,

Emil Fajrul Islami Gushelmi

BAB I
Dasar-Dasar Deep Learning

A. Apa itu AI ?

Pengertian AI dari beberapa sumber dapat diketahui dari 4 pendapat pakar-pakar berikut:

- Otomasi aktivitas yang berhubungan dengan proses berpikir, pemecahan masalah dan pembelajaran (Bellman,1978).
- Studi tentang kemampuan mengindera dengan menggunakan model komputasi. (Charniak+McDermo tt, 1985).
- Studi bagaimana cara melakukan sesuatu sehingga menjadi lebih baik (Rich+Knight, 1991).
- Cabang dari ilmu komputer yang fokus pada otomasi perilaku yang cerdas. (Luger+Stubblefield,1993)

Secara garis besar AI dapat dibedakan menjadi 4 kategori yaitu:

- *Thinking Humanly*

  Pendekatan ini dilakukan dengan dua cara yaitu pertama melalui introspeksi, mencoba menangkap pemikiran kita sendiri saat kita berfikir. "*how do you know that you understand*?".

  Yang kedua yaitu melalui penelitian-penelitian dari segi psikologi.

- *Acting Humanly* (*The Turing test approach*, 1950)

  Pendekatan ini pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer yang berintelijensia

(bot Cerdas) untuk menguji apakah komputer tersebut mampu mengelabui seorang manusia/ interrogator melalui komunikasi berbasis teks jarak jauh. Tentunya komputer tersebut harus memiliki kemampuan, *Natural Language Processing*, *Knowledge Representation*, *Automated Reasoning*, *Machine Learning*, *Computer Vision*, *Robotics*.

- *Thinking rationally*
  Pada pendekatan ini terdapat dua masalah yaitu pertama tidak mudah membuat pengetahuan informal, lalu menyatakan dalam formal term dengan notasi-notasi logika. Yang kedua terdapat perbedaan besar antara dapat memecahkan masalah "secara prinsip" dan memecahkannya "dalam dunia nyata".

- *Acting rationally* (*The Rational agent approach*)
  Pendekatan ini membuat inferensi logis yang merupakan bagian dari suatu rational agent. Karena untuk melakukan aksi secara rasional adalah dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa aksi yang dilakukan akan mencapai tujuan atau tidak. Sampai saat ini, pemikiran manusia yang diluar rasionalitas, yakni refleks dan intuitif (berhubungan dengan perasaan) belum dapat sepenuhnya ditirukan oleh komputer. Kedua definisi diatas dirasa kurang tepat untuk saat ini. Jika menggunakan definisi ini, maka banyak produk AI saat ini yang tidak layak disebut sebagai piranti cerdas. Definisi AI yang paling tepat saat ini adalah *acting rationally*.

B. Fondasi AI

Manusia dibekali kecerdasan yang luar biasa. Pada usia 3 tahun, dia sudah mampu mengenali berbagai macam benda walaupun hanya terlihat sebagian. Ketika melihat sebagian ekor cicak, maka dia akan dengan mudah mengenali bahwa hewan tersebut adalah cicak yang sedang bersembunyi dibalik bingkai lukisan. Pada usia dewasa, kecerdasannya terus berkembang dengan pesat, mulai dari kecerdasan kognitif, emosional dan spiritual. Sampai saat ini belum ada satu mesinpun yang mampu menyamai kecerdasan manusia secara keseluruhan. Selama bertahun-tahun, para ilmuwan berusaha mem-pelajari kecerdasan manusia. Dari pemikiran para ilmuwan tersebut, maka lahirlah AI sebagai cabang ilmu yang berusaha memahami kecerdasan manusia. Dukungan perkembangan teknologi, baik hardware maupun *software* yang sangat beragam. Hingga saat ini AI telah menghasilkan banyak piranti cerdas yang sangat berguna bagi kehidupan manusia. Hingga saat ini pula AI terus dipelajari dan dikembangkan secara meluas maupun mendalam.

C. Sejarah AI

Istilah AI pertama kali dikemukakan pada tahun 1956 dikonferensi Darthmouth. Berikut adalah rangkuman singkat terkait tahapan sejarah perkembangan AI:

- Era komputer elektronik (1941)
  Telah ditemukan alat sebagai komputer elektronik yang dikembangkan di USA dan Jerman. Komputer tersebut memerlukan ruangan yang luas dan ruang AC yang terpisah dan melibatkan konfigurasi ribuan kabel. Penemuan ini menjadi dasar pengembangan program yang mengarah ke AI.

- Masa Persiapan AI (1943-1956)
  Warren McCulloch & Walter Pitts berhasil membuat suatu model sel syaraf tiruan (1943). Dan Norbert Wiener membuat penelitian mengenai prinsip teori feedback (1950). Sedangkan John McCarthy (bapak AI) melakukan penelitian bidang Automata, JST dan pembelajaran intelijensia dengan membuat program yang mampu berfikir.

- Awal Perkembangan AI (1952-1969)
  Kesuksesan Newell dan Simon dengan program "*General Problem Solver*". Program ini digunakan menyelesaikan masalah secara manusiawi. McCarthy mendemokan bahasa pemrograman tingkat tinggi yaitu LISP di MIT AI Lab. KemudianNathaniel Rochester dari IBM dan mahasiswa-mahasiswanya mengeluarkan program AI yaitu "Geometry Theorm Prover"yang mampu membuktikan suatu teorema (1959).

- Perkembangan AI melambat (1966-1974)
  Program AI yang bermunculan hanya mengandung sedikit atau bahkan tidak mengandung sama sekali pengetahuan pada subjeknya. Banyaknya permasalahan yang harus diselesaikan oleh AI, karena terlalu banyaknya masalah yang berkaitan, maka tidak jarang terjadi kegagalan ketika membuat program AI. Ada beberapa batasan pada struktur dasar yang digunakan untuk menghasilkan perilaku intelijensia, contohnya dua masukan data yang berbeda tidak dapat dilatih untuk mengenali kedua masukan yang berbeda.

- Sistem berbasis pengetahuan (1969-1979)
  Ed Feigenbaum, dkk, membuat program untuk meme--cahkan masalah struktur molekul (Dendral Programs) yang berfokus pada segi pengetahuan kimia. Dan Saul Amarel dalam proyek "*Computer in Biomedicine*" membuat program dari segi pengetahuan diagnosa medis.

- AI menjadi sebuah industri (1980-1988)
  Pada saat menjadi sebuah industry AI menemukan *expert system* (R1) yang mampu mengkonfigurasi sistem-sistem komputer. *Booming* industri AI juga melibatkan banyak perusahaan besar yang menawarkan software tools untuk membangun sistem pakar.

- Kembalinya Jaringan Syaraf Tiruan (1986-sekarang)
  Pada masa tahun ini Hopfield mengembangkan teknik mekanika statistik untuk mengoptimasi jaringan syaraf tiruan(1982). David Rumelhart & Geoff Hinton menemukan algoritma *back-propagation*. Algoritma ini berhasil diimplementasikan pada bidang ilmu komputer dan psikologi (1985).

D. Apa Itu *Deep Learning*?

Deep learning adalah sebuah *artificial intelligence* yang dapat meniru proses kerja otak manusia. Teknologi ini sangat efektif untuk mengolah data mentah dan menciptakan pola untuk keperluan pengambilan keputusan. Deep learning sendiri merupakan bagian dari *machine* learning yang memiliki jaringan tersendiri. Ia mampu mengenali pola dan informasi tanpa pengawasan dari data yang tidak terstruktur atau tidak berlabel. Nah, karena kemampuannya ini, teknologi *deep learning* juga dikenal sebagai *deep neural learning* atau *deep network learning*. Selain digunakan pada berbagai aplikasi raksasa, deep learning meru-

pakan teknologi utama di balik mobil tanpa pengemudi. Ia memungkinkan kendaraan untuk mengenali tanda berhenti dan membedakan pejalan kaki dari tiang lampu. Teknologi ini juga kunci dari kinerja *voice control* dalam perangkat sehari-hari seperti smartphone, tablet, TV dan *speaker hands-free*.

Deep learning merupakan subbidang *machine learning* yang algoritmanya terinspirasi dari struktur otak manusia. Struktur tersebut dinamakan *Artificial Neural Networks* atau disingkat ANN. Pada dasarnya, ia merupakan jaringan saraf yang memiliki tiga atau lebih lapisan ANN. Ia mampu belajar dan beradaptasi terhadap sejumlah besar data serta menyelesaikan berbagai permasalahan yang sulit diselesaikan dengan algoritma machine learning lainnya.

E. Jenis-Jenis Algoritma *Deep Learning*

Deep learning adalah sebuah teknologi yang mampu bekerja menggunakan beberapa algoritma tertentu. Sejatinya, tidak ada algoritma *deep learning* yang dianggap sempurna. Sebab, masing-masing jenis memiliki kapabilitas yang berbeda. Maka dari itu, *application developer* harus memilih jenis algoritma yang paling sesuai dengan kebutuhan mereka. Untuk memilih algoritma yang tepat, ada baiknya kamu pahami masing-masing jenis algoritma *deep learning*. Berikut pemaparannya merujuk pada *Simpli Learn*.

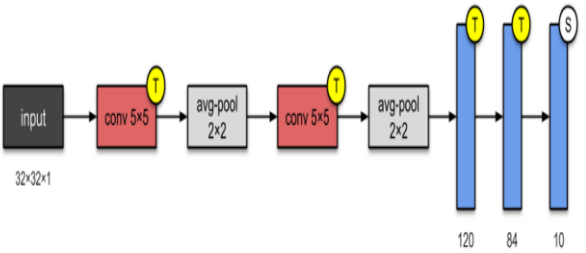
1. Convolutional Neural Networks (CNN)

   CNN, yang juga dikenal sebagai ConvNets, adalah salah satu algoritma *deep learning* yang bisa kamu manfaatkan. Ia terdiri dari beberapa lapisan dan sering digunakan untuk pemrosesan gambar dan deteksi objek. CNN pertama kali dikembangkan pada tahun 1988

ketika ia masih disebut sebagai LeNet. Pada masa itu, teknologi tersebut digunakan untuk mengenali karakter seperti kode pos dan angka. CNN kini banyak digunakan untuk mengidentifikasi citra satelit, memproses citra medis, memperkirakan deret waktu, dan mendeteksi anomali. CNN memiliki beberapa arsitektur yang bisa digunakan untuk memproses data. Berikut beberapa arsitektur pada CNN :

- Lenet

  LeNet-5 adalah salah satu arsitektur paling sederhana. Ini memiliki 2 *convolutional* dan 3 lapisan yang sepenuhnya terhubung. Lapisan penyatuan rata-rata seperti yang kita kenal sekarang disebut lapisan *sub-sampling* dan memiliki bobot yang dapat dilatih (yang bukan merupakan praktik merancang CNN saat ini). Arsitektur ini memiliki sekitar 60.000 parameter. Arsitektur Lenet dapat dilihat pada gambar 1.1 :

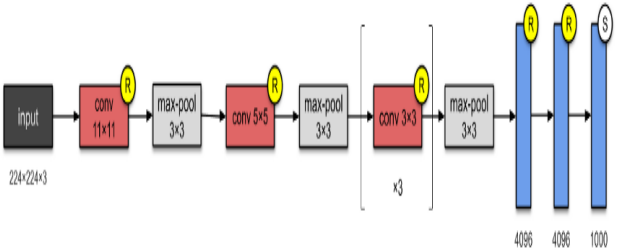


Gambar 1.1. Arsitektur Lenet

- Alexnet

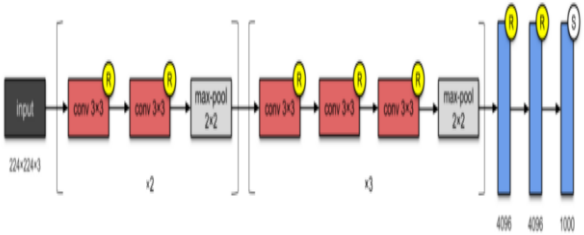
  Dengan parameter 60 juta, AlexNet memiliki 8 lapisan-5 konvolusi dan 3 terhubung sepenuhnya.

AlexNet baru saja menumpuk beberapa lapisan lagi ke LeNet-5. Pada saat publikasi, penulis menunjukkan bahwa arsitektur mereka adalah "salah satu jaringan saraf konvolusi terbesar hingga saat ini di himpunan bagian dari ImageNet". Arsitektur Alexnet dapat dilihat pada gambar 1.2 :



Gambar 1.2. Arsitektur Alexnet

- VGG-16
  VGG-16 merupakan model CNN yang memanfaatkan convolutional layer dengan spesifikasi *convolutional* filter yang kecil (3×3). Dengan ukuran convolutional filter tersebut, kedalaman *neural network* dapat ditambah dengan lebih banyak lagi *convolutional* layer. Arsitektur VGG-16 dapat dilihat pada gambar 1.3 :



Gambar 1.3. Arsitektur VGG-16

- Inception-V1

  Arsitektur 22-layer dengan parameter 5M ini disebut Inception-v1. Di sini, pendekatan Jaringan Dalam Jaringan banyak digunakan, seperti yang disebutkan dalam makalah ini. *Network In Network* diimplementasikan melalui modul Inception. Desain arsitektur modul *Inception* adalah produk penelitian tentang pendekatan struktur yang jarang. Setiap modul menyajikan 3 ide:

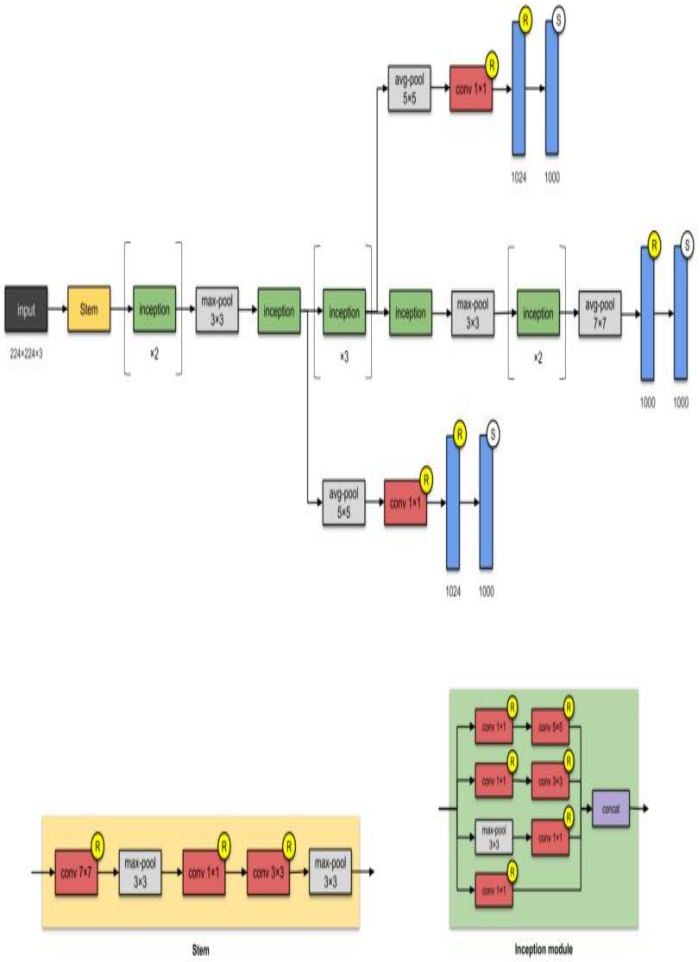
  a Memiliki menara konvolusi paralel dengan filter berbeda, diikuti dengan penggabungan, menangkap fitur berbeda pada 1×1, 3×3, dan 5×5, sehingga 'mengelompokkannya'. Ide ini dilatarbelakangi oleh Arora et al. dalam makalah Batas yang dapat dibuktikan untuk mempelajari beberapa representasi mendalam, menyarankan konstruksi lapis demi lapis di mana seseorang harus menganalisis statistik korelasi dari lapisan terakhir dan mengelompokkannya ke dalam kelompok unit dengan korelasi tinggi.

b. Konvolusi 1×1 digunakan untuk pengurangan dimensi untuk menghilangkan kemacetan komputasi.

c. Karena fungsi aktivasi dari konvolusi 1×1, penambahannya juga menambah nonlinier. Ide ini didasarkan pada makalah Network In Network. Lihat lampiran di sini.

d. Penulis juga memperkenalkan dua pengklasifikasi tambahan untuk mendorong diskriminasi pada tahap pengklasifikasi yang lebih rendah, untuk meningkatkan sinyal gradien yang disebarkan kembali, dan untuk memberikan regularisasi tambahan. Jaringan bantu (cabang yang terhubung ke pengklasifikasi tambahan) dibuang pada waktu inferens.

Arsitektur Inception-v1 dapat dilihat pada gambar 1.4 :



Gambar 1.4. Arsitektur Inception-v1

2. *Long Short Term Memory Network* (LSTM)
   LSTM adalah jenis Reccurent Neural Network (RNN) yang dapat mempelajari dan menghafal ketergantungan pola jangka panjang. Teknologi ini mampu mengingat seluruh informasi masa lalu dari periode-periode

tertentu. LSTM juga bisa menyimpan informasi dari waktu ke waktu. Mereka berguna untuk keperluan prediksi deret waktu karena ia bisa mengingat input sebelumnya. LSTM memiliki struktur seperti rantai di mana keempat lapisan di dalamnya saling berinteraksi dengan cara yang unik.

3. *Reccurent Neural Network* (RNN)
   Jenis algoritma deep learning berikutnya yang bisa kamu manfaatkan adalah *Reccurent Neural Network* (RNN). RNN memiliki koneksi yang bisa membentuk siklus terarah. Siklus tersebutlah yang memungkinkan output dari LSTM untuk diumpankan sebagai input ke fase terbaru. Setelah output dari LSTM menjadi input terbaru, ia dapat mengingat input sebelumnya karena kinerja memori internal. RNN biasanya digunakan untuk teks gambar, analisis deret waktu, *natural language processing*, pengenalan tulisan tangan, dan mesin translasi.

4. *Self Organizing Maps* (SOM)
   Jenis algoritma *deep learning* selanjutnya adalah SOM atau *Self Organizing Maps*. Sesuai namanya, teknologi ini mampu menginisiasikan data *visualization* secara mandiri. Kemampuan ini berfungsi untuk mengurangi dimensi data melalui jaringan saraf tiruan yang dapat bekerja secara otomatis. Data *visualization* ini ditujukkan untuk memecahkan permasalahan yang umumnya cukup sulit untuk diselesaikan manusia. SOM sendiri diciptakan untuk membantu pengguna memahami informasi berdimensi tinggi.

F. Contoh Penerapan *Deep Learning*

Penerapan teknologi *deep learning* dalam kehidupan sehari-hari kini sudah semakin luas. Berikut adalah beberapa contoh *deep learning* yang umum digunakan:

- Pengenalan Objek
  Teknologi ini digunakan untuk mengenali dan mendeteksi objek pada gambar dan video. Contohnya antara lain, fitur untuk menandai seseorang dalam sebuah foto di media sosial, pengenalan komponen elekronika.

- Biometrik Pengenalan Wajah
  Metode biometrik adalah metode keamanan yang dinilai paling aman. Sebab, untuk menembus sistem keamanan tersebut. Kita memerlukan data biometrik asli berupa sidik jari, bentuk wajah, atau bisa juga retina mata. Bisa juga untuk mengenali orang yang memakai masker dan orang yang tidak memakai masker. Proses pengenalan biometrik tersebut sebenarnya juga merupakan bentuk penerapan teknologi *deep learning*.

- *Virtual Assistants*
  Smartphone yang bisa mengenali suara dan bahasa saat menjalankan fitur *Virtual Assistant* sebenarnya merupakan contoh penerapan *deep learning*.

- Mobil Otomatis
  Sudah pernah mendengar tentang mobil otomatis tanpa awak dari Tesla? Agar mobil tidak menabrak dan punya kemampuan mengemudi layaknya seorang manusia, maka mesin perlu mengumpulkan banyak data yang berhubungan dengan rambu lalu lintas, perilaku pengguna jalan, hingga kemungkinan risiko di jalan. Hal ter-

sebut menjadi mungkin dengan bantuan teknologi deep learning yang ditanamkan dalam sistem mobil Tesla.

- *Chatbots*
  Saat ini, penggunaan chatbot sudah semakin masif. Penggunaan *chatbot* terbukti mampu meringankan pekerjaan manusia seperti industri *customer service*. Dengan *chatbot* yang ditingkatkan menggunakan teknologi *deep learning*, sistem akan terus mempelajari respons yang tepat saat menghadapi pelanggan.

- Penerjemahan
  Mirip dengan *Virtual Assistant*, contoh *deep learning* pada layanan penerjemahan pun mempelajari suara dan bahasa yang digunakan manusia. Mekanisme ini sangat memudahkan banyak orang.

BAB II
Pengenalan Teachable Machine

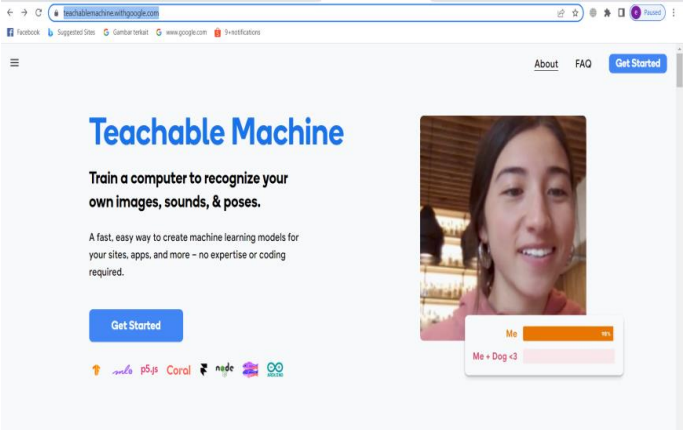
A. Apa itu *Teachable Machine*?

Teachable Machine ialah antarmuka yang memungkinkan siapapun mengajarkan algoritma dengan cara mengklasifikasikan kumpulan data. Jika ingin mengeksplorasi penggunaan antarmuka *machine learning* interaktif, *teachable machine* bisa digunakan untuk memperkenalkan pembelajaran mesin kepada siapa saja. Pada intinya, antarmuka tersebut dapat dibuat dan dapat diakses karena aplikasi tersebut dapat menggunakan audio atau gambar sebagai data input yang meningkatkan representasi alternatif yang dapat digunakan untuk mengaplikasikan konsep.

Teachable machine adalah *interface* berbasis web yang mengizinkan pengguna untuk melatih mereka dalam membuat model klasifikasi *machine learning* sendiri, tanpa menggunakan*coding*, hanya menggunakan *webcam*, gambar, ataupun suara dengan cepat. Sebelumnya sudah ada *Teachable Machine* 1.0 yang hanya bisa membuat atau mengenali sebuah pose dan mencocokannya dengan gambar yang sudah disediakan oleh Google dengan menggunakan *webcam*. Di versi 2.0 sudah bisa melatih AI hanya dengan mengklik satu tombol, tanpa *coding*, dan bisa mencocokan dengan sebuah suara ataupun pose. Sebagai contoh, jika kamu sedang duduk, AI akan tahu kamu sedang duduk, begitupun jika kamu sedang berdiri.

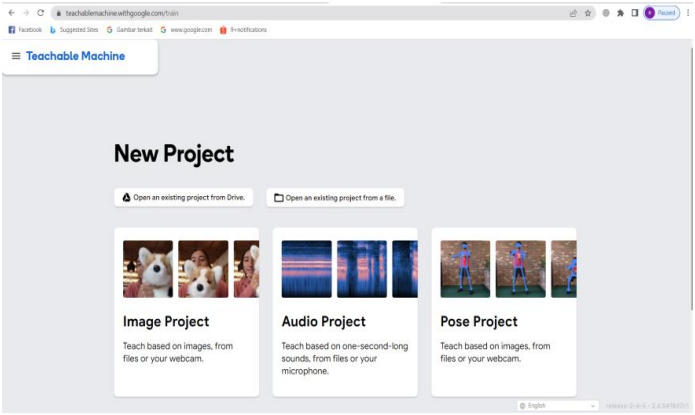
B. Jenis Model *Teachable Machine*

Untuk mengaktifkan *software* aplikasi *Teachable Machine* dapat dilakukan dengan cara masuk pada alamat URL : 'https://teachablemachine.withgoogle.com/', sehingga mu ncul tampilan seperti gambar 2.1 berikut ini.



Gambar 2.1. Tampilan Software Aplikasi Teachable Machine

Selanjutnya setelah tombol '*Get Started*' di-klik, maka akan muncul tampilan pada gambar 2.2.



Gambar 2.2. *Project* yang ditawarkan oleh *Software* Aplikasi *Teachable Machine*

Teachable Machine baru menyediakan 3 jenis model : *Image Project* untuk deteksi, kasifikasi *image*, *audio project* untuk *recognize audio*, dan pose *project* untuk *recognize* pose, seperti yang tampilan pada gambar 2.2.

- *Image project* : *project* berbasis image, pada image project kita bisa membuat aplikasi pengenalan objek berbasis gambar (*image*). Bisa juga untuk membuat klasifikasi objek dan aplikasi deep learning lainnya yang berbasis gambar (*image*)
- *Audio project* : pada audio *project* kita bisa membuat aplikasi pengenalan suara seperti : *speech recognize*. Contoh seperti aplikasi mengaktifan peralatan elektronik berbasis suara.

- *Pose Project* : Pada pose *project* kita bisa membuat aplikasi yang berbasis gerakan badan atau sikap dari sebuah gerakan. Contoh seperti aplikasi pengenalan tindakan seseorang yang akan mendatangkan hal-hal berisiko terhadap orang lain.

BAB III
PENGENALAN NVIDIA JETSON NANO

A. Apa itu Nvidia Jetson Nano ?

NVIDIA Jetson Nano adalah produk NVIDIA yang dapat diperlakukan sebagai komputer mini dengan penam-bahan perangkat keras dan perangkat lunak. Perangkat keras tambahan yang diperlukan seperti *mouse*, *keyboard*, dan *monitor*. Untuk perangkat lunak memerlukan micro SD sebagai media penyimpanan agar sistem operasi yang digunakan bisa dijalankan dengan baik.

NVIDIA Jetson Nano mulai diperkenalkan pada pertengahan Maret 2019. Produk ini ditujukan untuk aplikasi *Artificial Intelligence*, *Internet of Things* (IoT). Board NVIDIA Jetson Nano ini terdiri dari CPU dengan 1,43 GHz dan GPU dengan 128 cores dari generasi Maxwell.

Sejak diluncurkan ke publik pertama kali, terdapat beberapa model dari NVIDIA Jetson Nano, antara lain seperti terlihat pada Gambar 3.1. Model ini disebut sebagai NVIDIA Jetson Nano A02.

Gambar 3.1. NVIDIA Jetson nano A02

Saat ini NVIDIA telah merilis model baru, NVIDIA Jetson NanoB01. Secara teknis, kedua model memiliki CPU

dan GPU yang sama, tetapi beberapa perangkat / peripheral yang berubah. Bentuk fisik dari NVIDIA Jetson Nano B01 dapat dilihat pada Gambar 3.2.

Gambar 3.2. NVIDIA Jetson nano B01

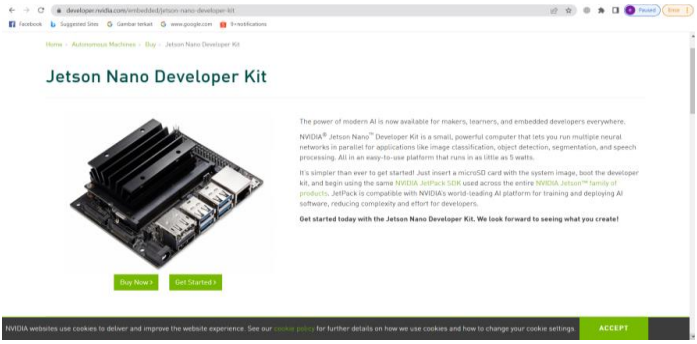
Adapun spesifikasi dari NVIDIA Jetson Nano dapat dilihat pada tabel berikut ini.

Table 3.1. Spesifikasi NVIDIA Jetson Nano

Fitur	**Keterangan**
GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD
Video Encode	4K @ 30 | 4x 1080p @ 30 |
Encode	9x 720p @ 30 (H.264/H.265)
VideoDecode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30
Decode	18x 720p @ 30(H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
I/O	GPIO, I2C, I2S, SPI, UART

Mechanical 69 mm x 45 mm, 260-pin edge connector

Informasi tentang Jetson Nano Developer Kit dapat dilihat pada https://developer.nvidia.com/embedded/jetson-nano-developer-kit. Setelah link url ini diklik, akan muncul tampilan seperti gambar 3.3 berikut ini.



Gambar 3.3. Laman Website Jetson Nano Developer Kit

Untuk memulai eksplorasi Jetson Nano, klik tombol '*Get Started*' sehingga muncul tampilan seperti gambar 3.4 berikut ini.

Gambar 3.4. Penjelasan Peripheral NVIDIA Jetson Nano

Keterangan nomor pada gambar 3.4 sebagai berikut.

1. Kartu microSD Sebagai Media Penyimpanan.
2. 40-pin header ekspansi.
3. Port Micro-USB untuk input tegangan 5V.
4. Port Ethernet Gigabit.
5. Port USB 3.0 (x4).
6. Port HDMI.
7. Konektor untuk Port Display.
8. Jack Barrel untuk tegangan input 5VDC.
9. Konektor kamera MIPI CSI-2.

Pada paket pembelian NVIDIA Jetson Nano, yang termasuk dalam kotaknya adalah :

- Modul NVIDIA Jetson Nano.
- Buku petunjuk berisi informasi singkat tentang modul tersebut.
- Kertas terlipat untuk alas modul NVIDIA Jetson Nano yang bisa dibuat berdiri.

Bahan / peralatan yang tidak masuk dalam paket pembelian namun dibutuhkan untuk menjadikan NVIDIA Jetson Nano sebagai komputer mini sebagai berikut :

- Kartu MicroSD dengan kapasitas minimal 32GB.
- Keyboard USB dan Mouse USB.
- Monitor (HDMI atau DP).
- Power Supply Micro-USB 5VDC ±2A.

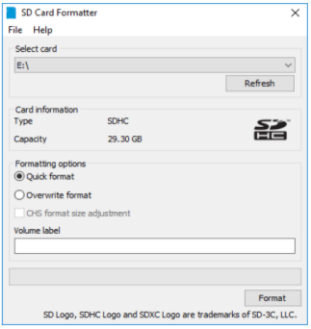
B. Menyiapkan Kartu microSD

Pada NVIDIA Jetson Nano, kartu microSD digunakan sebagai media tempat penyimpanan sistem operasi dan tempat penyimpanan data layaknya hardisk pada komputer (laptop). Sistem operasi diperoleh dari *image* sebesar 6.1 Gbyte yang bisa didownload pada laman NVIDIA dengan alamat URL: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#wr ite

Selanjutnya pindahkan image tersebut ke kartu microSD sesuai dengan sistem operasi yang digunakan. Pada contoh di buku ini hanya menggunakan ***sistem operasi Windows***. Untuk sistem operasi lain seperti **macOS** atau **Linux** silahkan ikuti petunjuk pada laman NVIDIA Jetson Nano tersebut.

Adapun langkah-langkah untuk Menulis (*Write*) *Image* Menggunakan Sistem Operasi Windows sebagai berikut.

- *Format* kartu microSD menggunakan SD Card *Formatter* dari SD *Association*.

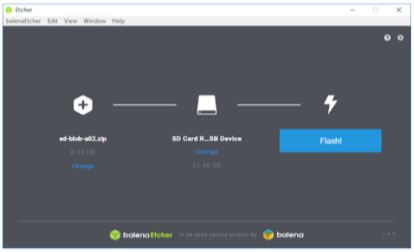


Gambar 3.5. Memformat kartu microSD menggunakan software SD Card Formatter

- Pilih drive yang ada kartu microSD.
- Kemudian pilih '*Quick format*'.
- Untuk 'Volume label' biarkan saja kosong.
- Kemudian klik '*Format*' dan jika ada pesan muncul, klik 'Yes' untuk memulai proses format.
- Selanjutnya gunakan aplikasi Etcher untuk menyimpan image Jetson Nano *Developer* ke microSD
- *Software Etcher* juga dapat didownload pada laman NVIDIA Jetson Nano tersebut. Tampilan *Etcher* dapat dilihat pada gambar 3.6.

Gambar 3.6. Tampilan *Software Etcher*

- Pilih '*Select image*' dan pilih image file dalam bentuk zip yang telah didownload sebelumnya.
- Kemudian masukkan kartu microSD. Jika ada pesan untuk menyarankan format microSD, harap diabaikan saja, karena kita sudah melakukan format microSD ini sebelumnya.
- Selanjutnya tekan '*Flash*!' seperti gambar 3.7. Proses menulis image ke kartu microSD memakan waktu lebih kurang 10 menit.

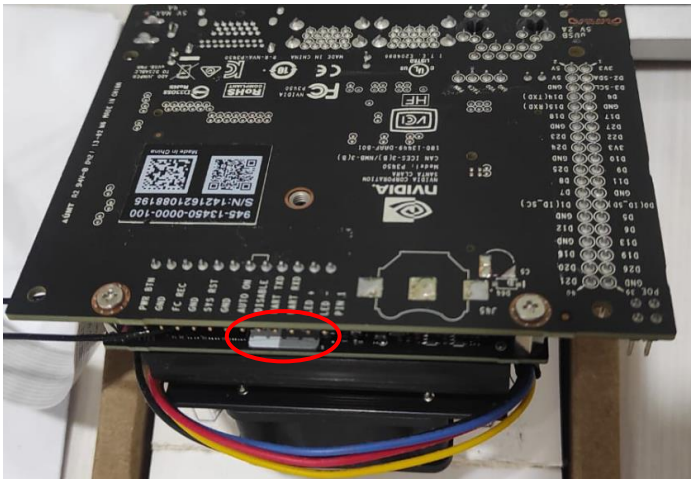


Gambar 3.7. Tampilan Saat Proses *Flash* Pada *Software Etcher*

- Jika proses sudah selesai, sistem operasi Windows akan menampilkan dialog yang menanyakan apakah kartu microSD akan diformat ulang. Abaikan saja dan pilih cancel. Hal ini karena sistem operasi yang tersimpan pada kartu microSD berbeda (berbasis Linux) sehingga tidak dikenal oleh sistem operasi Windows.
- Setelah proses ini tercapai, maka kartu microSD sudah siap digunakan pada board NVIDIA Jetson Nano.

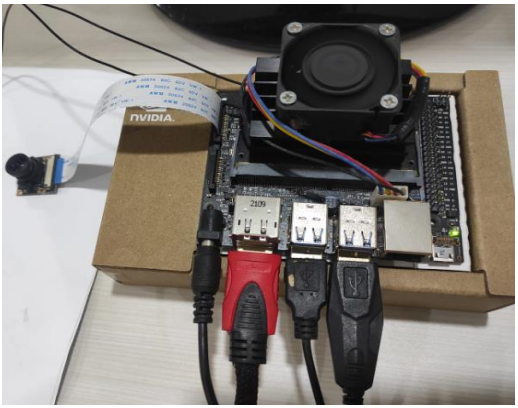
C. Menjalankan NVIDIA Jetson Nano

Setelah proses pengisian sistem operasi pada kartu microSD selesai dilakukan, langkah selanjutnya kita akan menjalankan NVIDIA Jetson Nano untuk pertama kalinya. Masukkan kartu microSD pada board NVIDIA Jetson Nano seperti pada gambar 3.8.



Gambar 3.8. Tampilan Posisi Kartu microSD Pada Board NVIDIA Jetson Nano

Langkah selanjutnya kita mencolokkan *keyboard*, *mouse*, dan *monitor* serta kamera CSI ke board NVIDIA Jetson Nano, seperti terlihat pada gambar 3.9. Untuk memberi daya pada perangkat NVIDIA Jetson Nano, dapat digunakan sumber catu daya dengan jack DC atau micro USB. Dalam hal ini kami menggunakan sumber catu daya dengan jack DC dengan adaptor daya 5V, 2A.



Gambar 3.9. Tampilan Colokkan *keyboard, mouse, monitor*, kamera CSI dan catu daya 5VDC

Setelah semua tersambung, colokkan sumber catu daya ke jala-jala PLN 220VAC. Langkah selanjutnya adalah melakukan konfigarsi sistem seperti membuat akun. Setelah proses konfigurasi selesai, kita akan melihat desktop NVIDIA Jetson Nano. Desktop ini berbasis Ubuntu Linux seperti terlihat pada gambar 3.10. Kita dapat melakukan aktivitas normal seperti yang dilakukan di komputer mana pun. Contohnya ; membuat dan mengedit file, menjelajah internet dan sebagainya.

Gambar 3.10. Tampilan Desktop NVIDIA Jetson Nano

1. Terminal

Image NVIDIA Jetson Nano dibuat dari Ubuntu, sehingga kita dapat menggunakan Terminal untuk melakukan tugas administrasi, seperti membuat/meng-edit file dan folder atau mengkompilasi dan menjalankan program. Sebagian besar, orang melakukan administrasi Linux dengan Terminal. Kita dapat menemukan Terminal NVIDIA Jetson dengan mengklik Cari di kiri atas. Ketik "*Terminal*" sehingga kita dapat melihat aplikasi Terminal, seperti yang ditunjukkan pada Gambar 3.11.

Gambar 3.11. Membuka Terminal di NVIDIA Jetson Nano

Setelah mengklik ikon Terminal, kita akan mendapatkan aplikasi Terminal, seperti yang ditunjukkan pada Gambar 3.12.

Gambar 3.12. Aplikasi terminal

Jika ingin menutup aplikasi Terminal, kita dapat mengetikkan perintah ini:

Exit

Gambar 3.13. Menutup Aplikasi terminal

Sekarang kita dapat melakukan tugas administrasi menggunakan Terminal.

2. *Restart* dan *Shut Down*

Terkadang kita ingin me-*reboot* sistem operasi NVIDIA Jetson Nano. Kita dapat mem-*boot* ulang secara manual. Caranya dengan mengklik ikon Pengaturan di kanan atas desktop NVIDIA Jetson Nano. Kita akan mendapatkan menu, seperti yang ditunjukkan pada Gambar 3.14. Pilih *Shut Down* dari menu.

Setelah melakukannya, akan mendapatkan dialog konfirmasi, seperti yang ditunjukkan pada Gambar 3.15. Ada dua opsi: *Shut Down* dan *Restart*. Pilih Restart untuk mem-*boot* ulang NVIDIA Jetson Nano.

Kita juga dapat mem-*boot* ulang NVIDIA Jetson Nano melalui Terminal. Kita dapat membuka Terminal dengan menekan tombol CTRL dan T secara bersamaan. Setelah membuka Terminal, ketikkan perintah ini:

$sudo reboot

Perangkat NVIDIA Jetson Nano akan reboot secara otomatis. Pastikan kita sudah menyimpan semua data sebelum melakukan *reboot*.

Gambar 3.14. Membuka menu untuk *Shut Down*

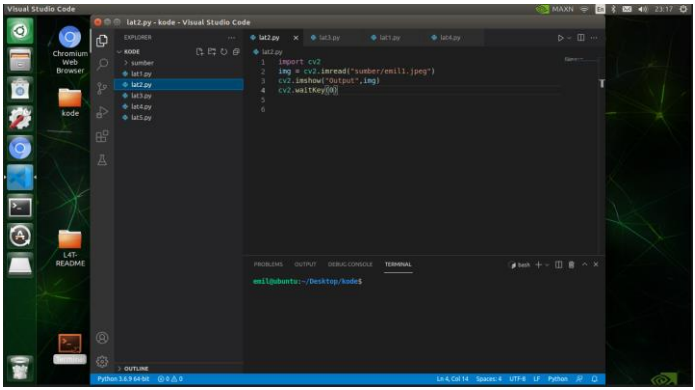
Gambar 3.15. Dialog konfirmasi untuk *reboot* dan *Shut Down*

Setelah proses *Shut Down*, jika kita tidak ingin menggunakan NVIDIA Jetson Nano lagi, maka kita dapat

mematikannya dengan cara mencabut colokan yang tersambung ke jala-jala PLN 220VAC.

D. Pemrograman NVIDIA Jetson Nano

Perangkat NVIDIA Jetson Nano dirancang untuk pengembang yang ingin membuat program pada *board*. Program tersebut seperti ; program C/C++, program *Python*, program *Node.js.* Untuk menulis kode tersebut dibutuhkan suatu editor. Terdapat beberapa editor yang bisa digunakan antara lain ; editor Nano, Visual Studio *Code* dari Microsoft.

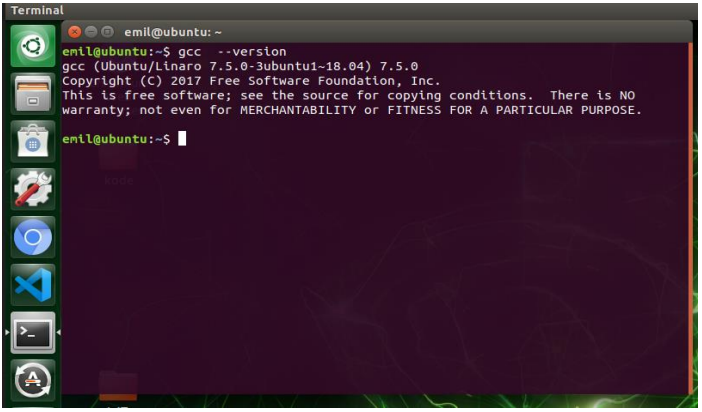


Gambar 3.16. Tampilan Visual Studio *Code* dengan Program Python

Kita dapat membuat program untuk C/C++ di NVIDIA Jetson Nano. *Image* NVIDIA Jetson Nano menyertakan kompiler GCC, sehingga kita dapat menggunakannya untuk mengkompilasi C/C++. Untuk memverifikasi versi GCC, buka Terminal dan ketik perintah ini:

$gcc --version

Kita akan melihat versi GCC di Terminal. Gambar 3.17 menunjukkan versi GCC pada NVIDIA Jetson Nano yang penulis gunakan.



Gambar 3.17. Versi *Checking* GCC

Kita juga bisa menggunakan bahasa pemrograman Python. Python adalah pemrograman skrip yang digunakan sebagian besar pengembang untuk membangun pemrosesan data dan implementasi ilmu data. Python juga dapat digunakan untuk membangun pemrograman perangkat keras. Ada berbagai pustaka Python yang menyediakan antarmuka drive perangkat keras.

Python 2.7 dan Python 3.x diinstal secara default melalui image NVIDIA Jetson Nano. Kita dapat memverifikasi ini dengan mengetikkan perintah ini di Terminal:

*$python --versio*n

Kita akan melihat Python 2.7.x terdaftar di Terminal. Untuk *Python* 3.x, dapat menggunakan perintah python3, sebagai berikut:

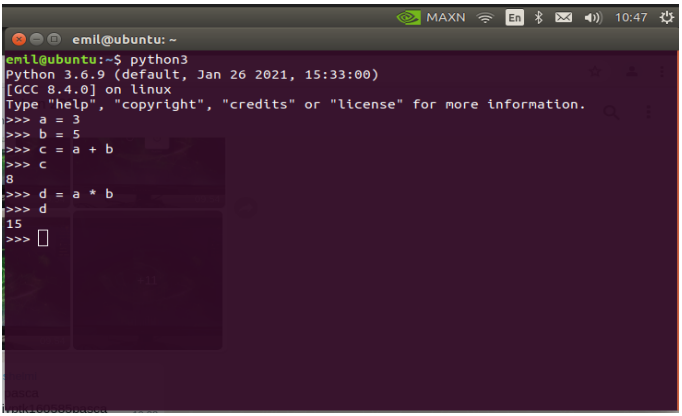
$python3 --version

$python3

Setelah eksekusi, kita akan melihat shell Python dan melihat >>> di Terminal, seperti yang ditunjukkan pada gambar 3.18. Di dalam *shell Python* >>>, kita akan coba menulis beberapa instruksi seperti di bawah ini.

```
>>> a = 3
>>> b = 5
>>> c = a + b
>>> c
>>> d = a * b
>>> d
```

Setelah instruksi selesai diketik dengan menekan tombol Enter, kita akan mendapatkan *output* dari *shell Python*, seperti yang ditunjukkan pada gambar 3.18.



Gambar 3.18. Menjalankan *Python Shell*

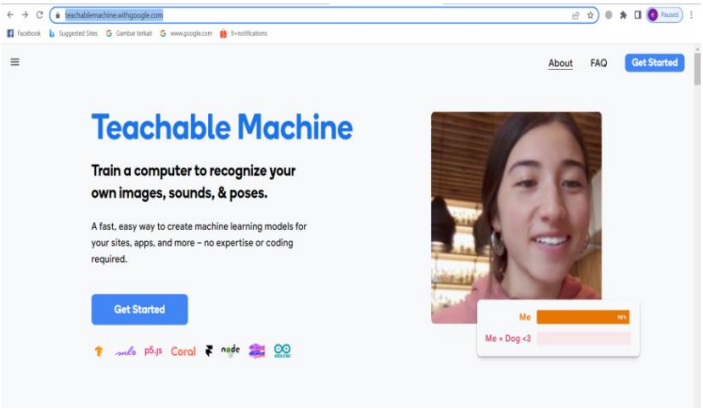
BAB IV

Contoh Aplikasi *Deep Learning* dengan *Teachable Machine*

A. Deteksi Penggunaan Masker pada Wajah

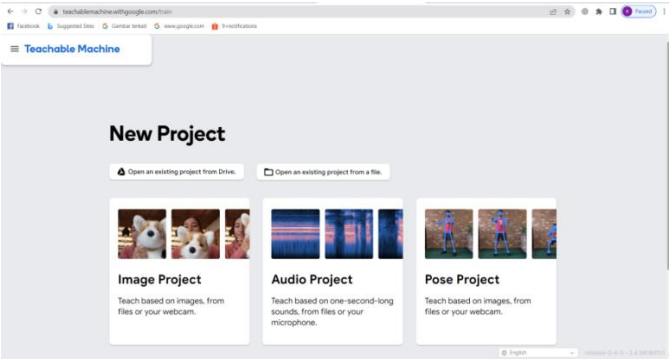
Untuk mendeteksi wajah menggunakan *Teachable Machine*, langkah yang harus dilakukan sebagai berikut.

- Aktifkan *Software* Aplikasi *Teachable Machine* pada alamat URL : 'https://teachablemachine.withgoogle.com/', sehingga muncul tampilan seperti gambar 4.1 berikut ini.



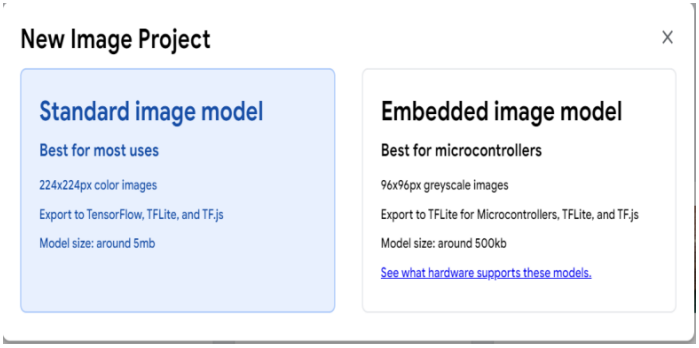
Gambar 4.1. Tampilan *Software* Aplikasi *Teachable Machine*

Selanjutnya setelah tombol '*Get Started*' di-klik, maka akan muncul tampilan pada gambar 4.2.



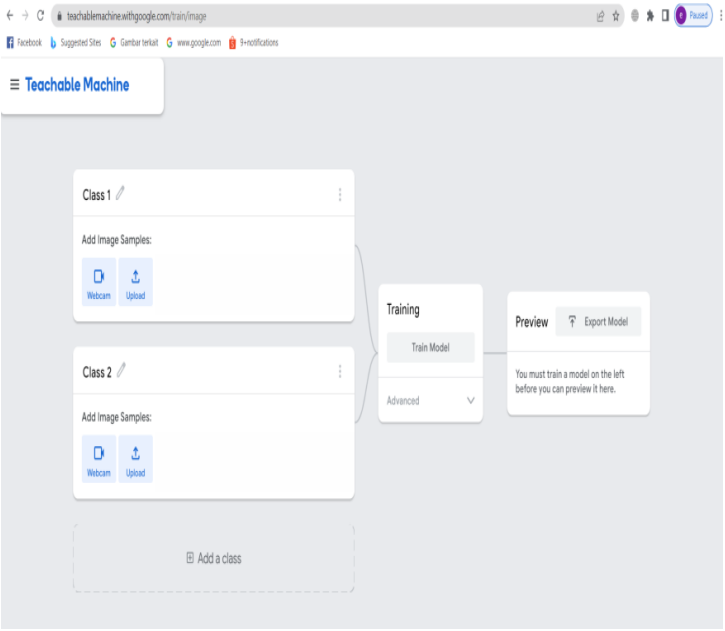
Gambar 4.2. *Project* yang ditawarkan oleh *Software* Aplikasi *Teachable Machine*

Klik pada *Image Project*, sehingga muncul tampilan pada gambar 4.3 berikut ini.



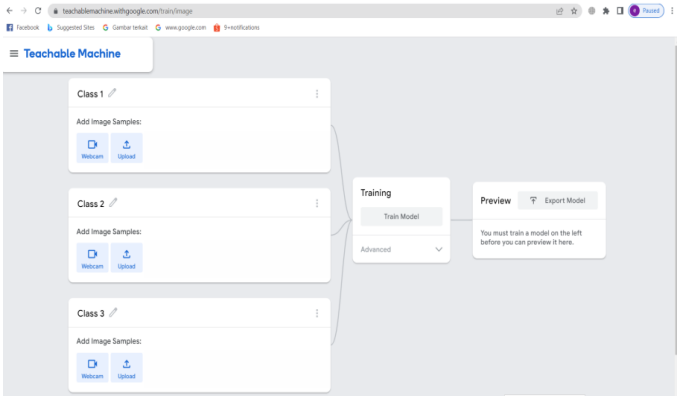
Gambar 4.3. Tampilan *New Image Project*

Di atas terdapat 2 (dua) pilihan, klik '*Standard image model*'. Hal ini karena kita ingin menguji gambar (*image*) standar dengan menggunakan komputer (PC/Laptop). Jika kita ingin menguji gambar (*image*) yang tersimpan dalam mikrokontroler, maka pilih '*Embedded image model*'. Setelah '*Standard image* model' di-klik akan muncul tampilan seperti terlihat pada gambar 4.4.



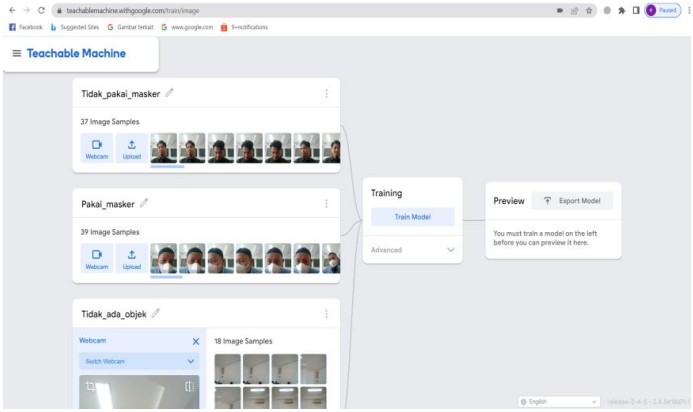
Gambar 4.4. Tampilan *Standard Image* Model Pada *Teachable Machine*

Pada gambar di atas terlihat ada 2 (dua) buah Class. Class tersebut bisa ditambahkan dengan meng-klik tombol '*Add a class*'. Pada setiap Class ini gambar (image) yang ingin dikenali (dideteksi) direkam, sehingga bisa menjadi sumber data (data set) bagi proses klasifikasi gambar (image) nantinya. Setelah di-klik tombol '*Add a class*', maka tampilannya menjadi seperti gambar 4.5 berikut.



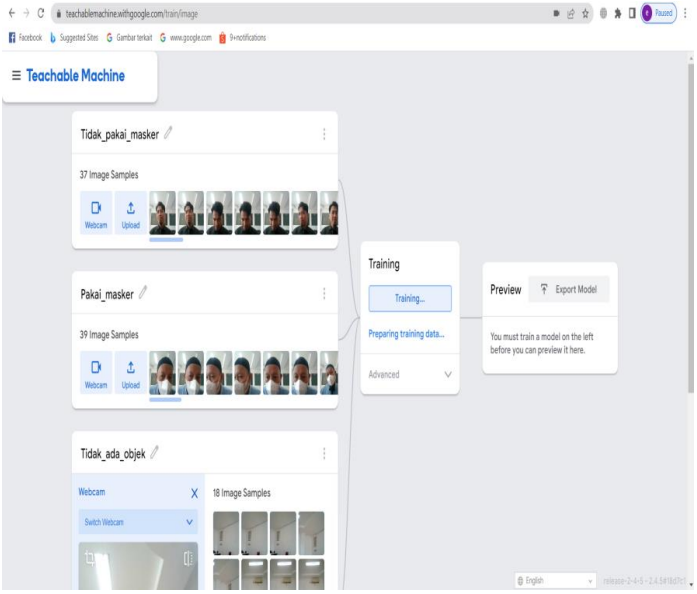
Gambar 4.5. Penambahan *Class* Pada *Teachable Machine*

Pada gambar 4.5 jumlah *Class* dijadikan 3 (tiga) buah, sesuai dengan tujuan yaitu mendeteksi ada atau tidaknya penggunakan/pemakaian masker. Nama *Class* tersebut bisa diubah sesuai kebutuhan, pada kasus contoh ini nama Class nya diubah menjadi : 'Tidak_pakai_masker', 'Pakai_masker' dan 'Tidak_ada_objek'. Tampilannya dapat dilihat pada gambar 4.6 berikut ini.



Gambar 4.6. *Class* Pada *Teachable Machine*

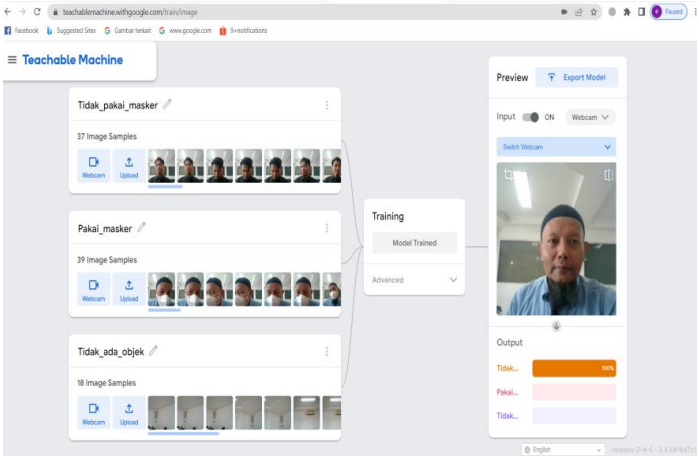
Pada gambar di atas, terlihat bahwa jumlah sampel gambar (*image*) berbeda-beda. Pada dasarnya, semakin banyak jumlah sampel gambar (*image*), maka proses pengenalan objek semakin akurat. Namun tentu saja ini akan membebani proses komputasi. Berdasarkan pengalaman penulis jumlah sampel minimal adalah 15 sampel dan maksimal 100 sampel. Sampel tersebut dapat berasal dari kamera (*webcam*) dan bisa juga berasal dari *upload* gambar (*image*) yang sudah disiapkan sebelumnya. Namun gambar (*image*) yang diupload harus berukuran 224x224 *pixel*. Untuk kedua proses ini, cukup dengan memilih salah satu kotak (*box*) dengan tulisan '*Webcam'* atau '*Upload'*.



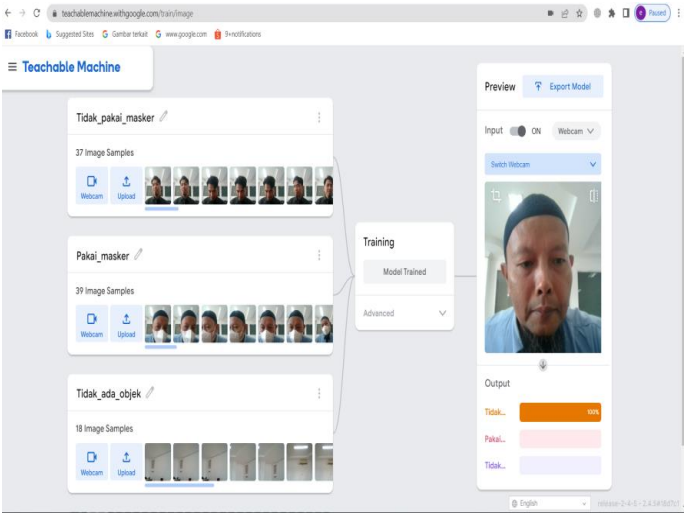
Gambar 4.7. Proses *Training* Gambar (*Image*)

Proses selanjutnya adalah melakukan pelatihan (training). Hal ini dilakukan dengan meng-klik tombol '*Train Model*'. Setelah tombol '*Train Model*' diklik, maka proses *training* dilakukan seperti terlihat pada gambar 4.7. Lama waktu proses Training tergantung pada jumlah sampel gambar (*image*) yang terdapat pada masing-masing *class*. Disamping itu juga tergantung pada kecepatan komputasi dari masing-masing komputer/laptop yang digunakan.

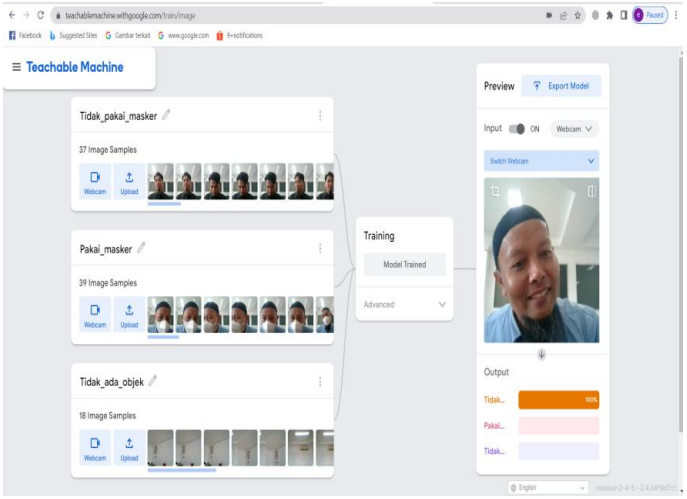
Setelah proses *traning* berhasil dilakukan, maka kamera akan aktif pada kotak (*box*) '*preview'*. Selanjutnya proses pengujian dilakukan. Pada gambar 4.8 sampai dengan gambar 4.11 dapat dilihat pengujian sampel orang pertama yang tidak menggunakan masker dengan hasil *capture* yang berbeda-beda. Hasil pengujian tersebut menunjukkan bahwa semua nilai 'Tidak_pakai_masker' adalah 100%.



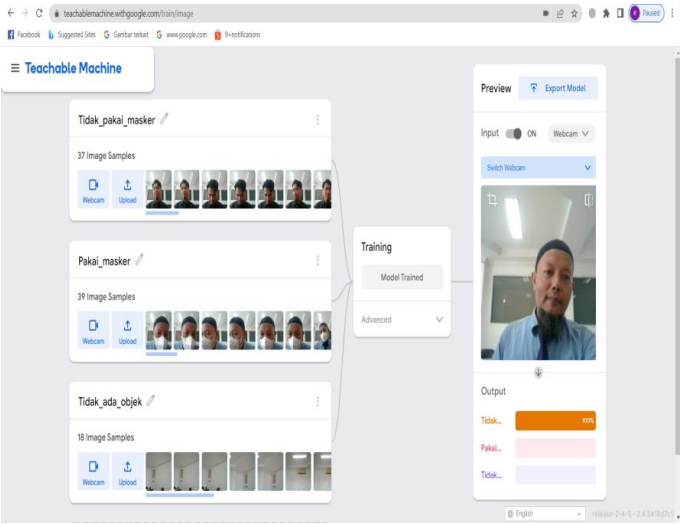
Gambar 4.8. Hasil Pengujian ke-1 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Pertama



Gambar 4.9. Hasil Pengujian ke-2 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Pertama

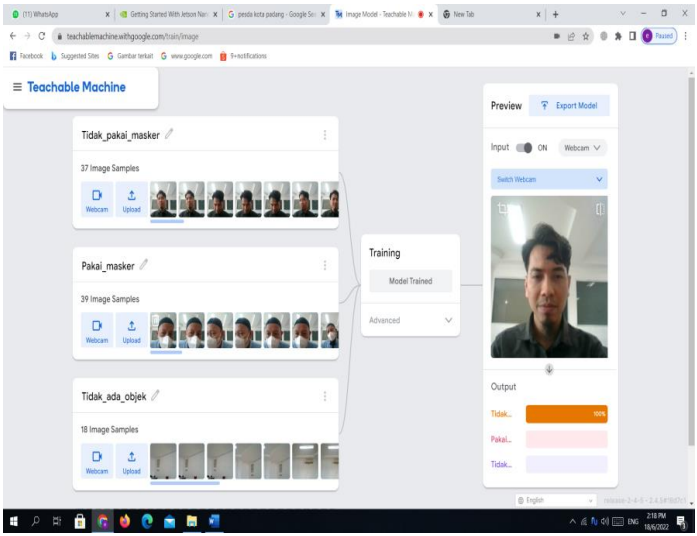


Gambar 4.10. Hasil Pengujian ke-3 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Pertama

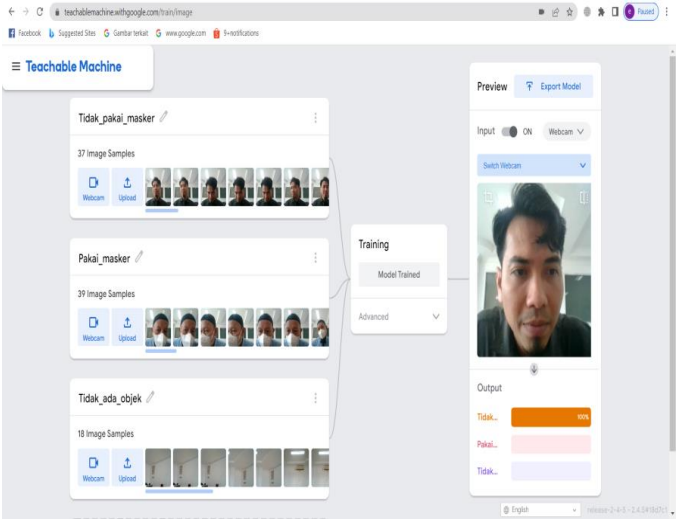


Gambar 4.11. Hasil Pengujian ke-4 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Pertama

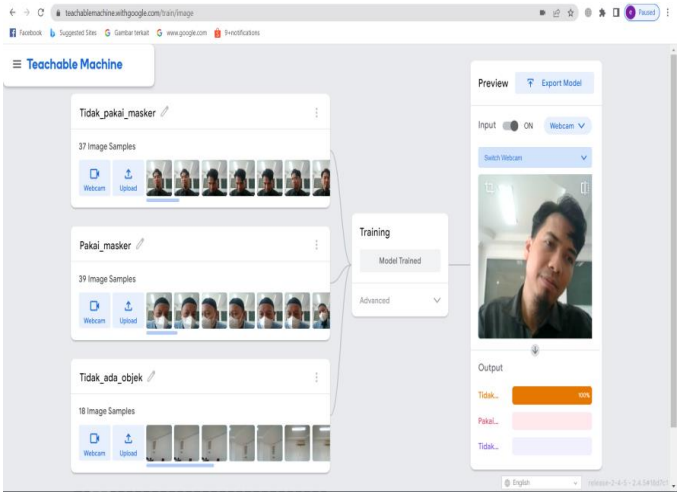
Pada pengujian berikutnya dengan sampel orang ke-2 (kedua) juga dilakukan dengan menguji beberapa posisi wajah di depan kamera. Hasil pengujian dapat dilihat pada gambar 4.12 sampai dengan gambar 4.15. Pada pengujian ini menunjukkan nilai 100% pada *Class* 'Tidak_pakai_masker'. Hal ini menunjukkan bahwa sistem deteksi yang dibangun menggunakan *Teachable Machine* mampu mengenali objek dengan baik dan akurat.



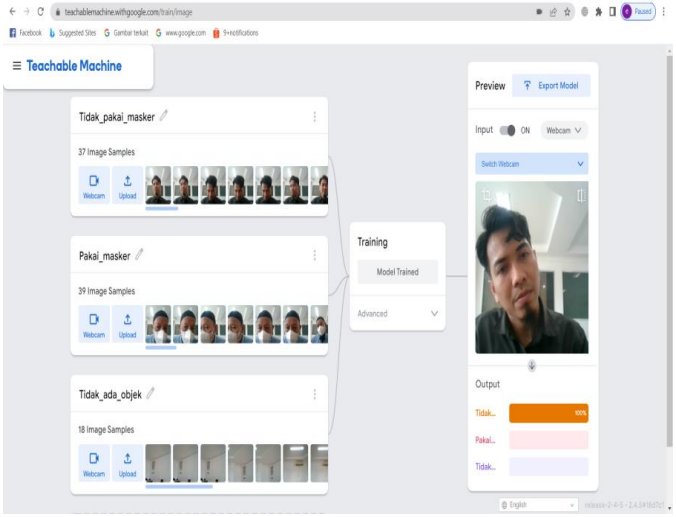
Gambar 4.12. Hasil Pengujian ke-1 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Kedua



Gambar 4.13. Hasil Pengujian ke-2 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Kedua

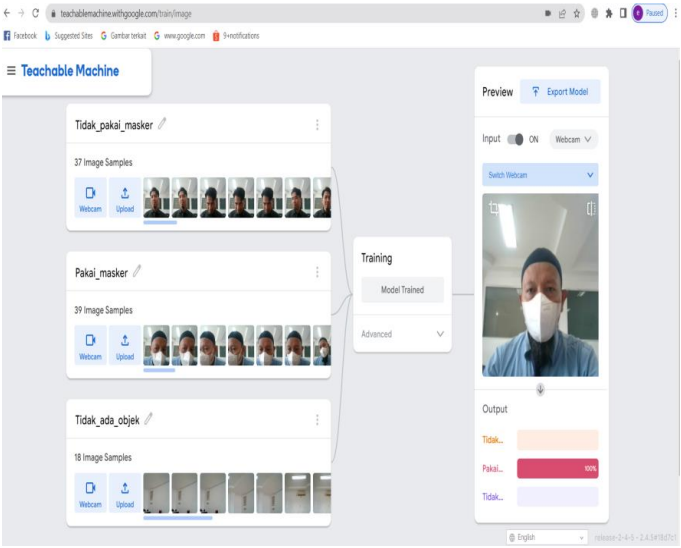


Gambar 4.14. Hasil Pengujian ke-3 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Kedua

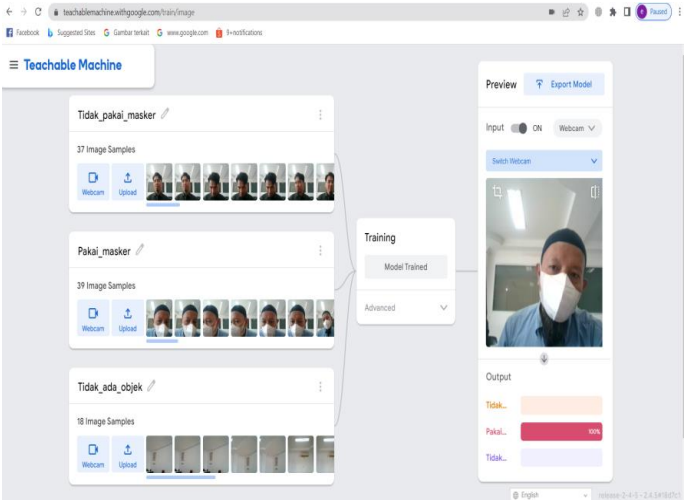


Gambar 4.15 Hasil Pengujian ke-4 Pada Pendeteksian 'Tidak_pakai_masker' Pada Sampel Orang Kedua

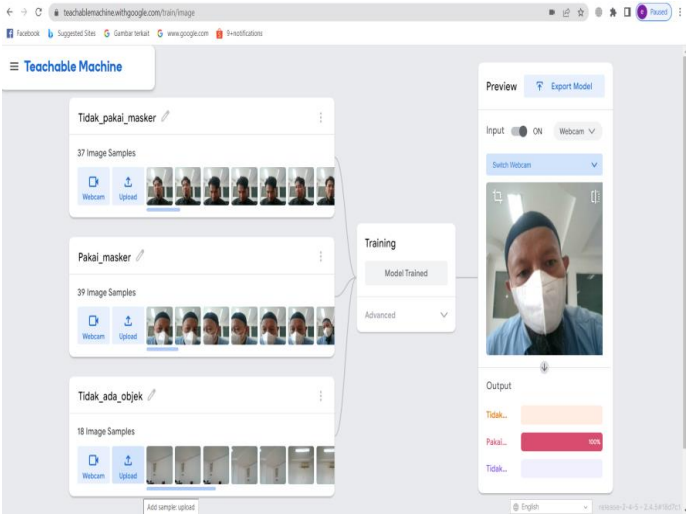
Selanjutnya dilakukan pengujian menggunakan masker dengan 2 (dua) buah sampel orang, yaitu sampel orang ke-1 pada gambar 4.16 sampai dengan gambar 4.19. Untuk sampel orang ke-2 pada gambar 4.20 sampai dengan gambar 4.23. Pada pengujian tersebut juga menghasilkan nilai 100 %. Hal ini menunjukkan bahwa sistem dapat mengenali orang yang tidak menggunakan masker dan orang yang menggunakan masker dengan baik dan akurat.



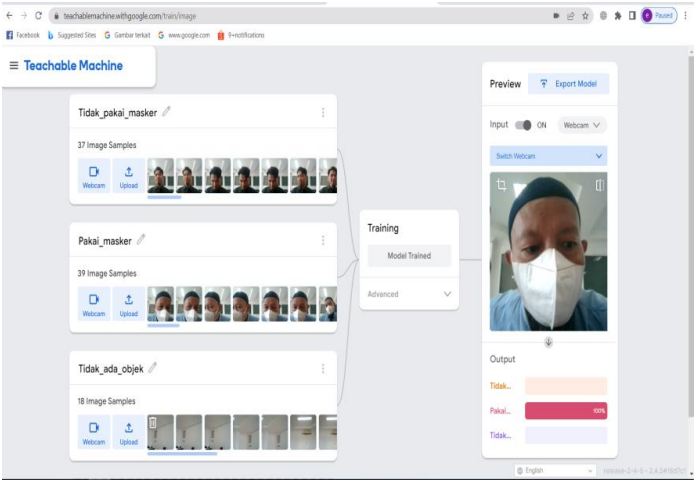
Gambar 4.16 Hasil Pengujian ke-1 Pendeteksian Masker Pada Sampel Orang Pertama



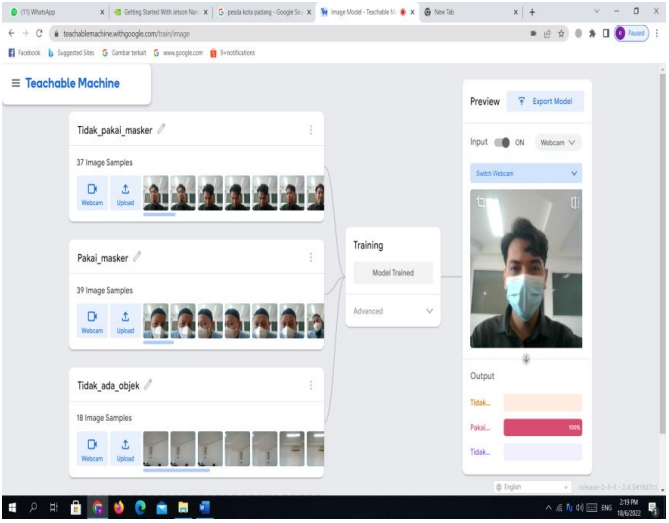
Gambar 4.17. Hasil Pengujian ke-2 Pendeteksian Masker Pada Sampel Orang Pertama



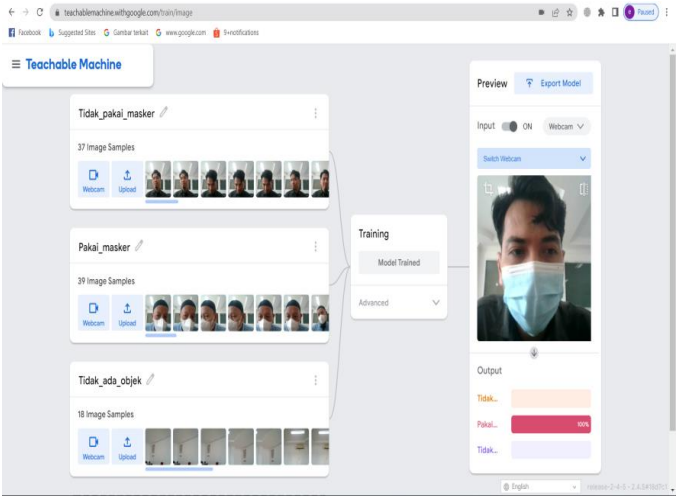
Gambar 4.18. Hasil Pengujian ke-3 Pendeteksian Masker Pada Sampel Orang Pertama



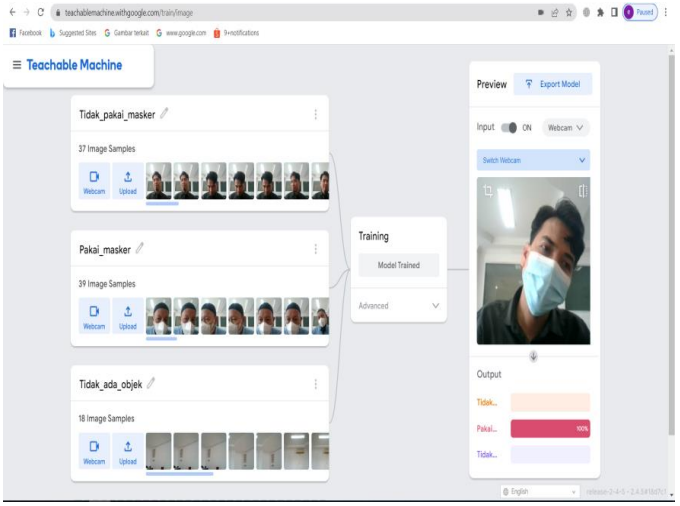
Gambar 4.19. Hasil Pengujian ke-4 Pendeteksian Masker Pada Sampel Orang Pertama



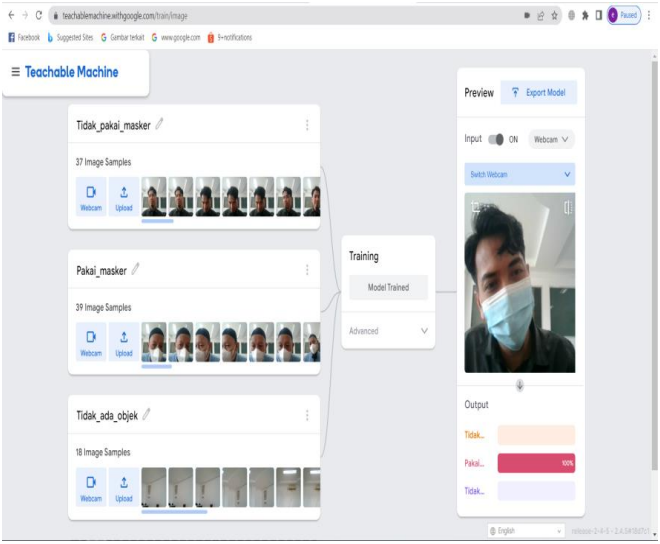
Gambar 4.20. Hasil Pengujian ke-1 Pendeteksian Masker Pada Sampel Kedua



Gambar 4.21. Hasil Pengujian ke-2 Pendeteksian Masker Pada Sampel Kedua

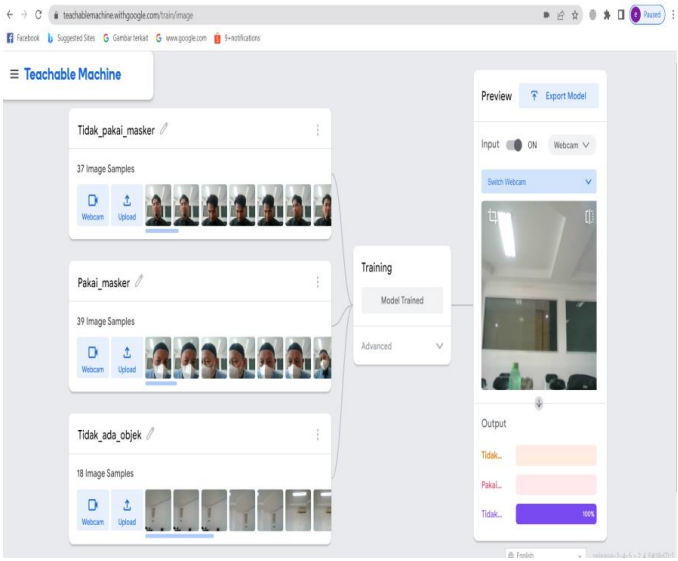


Gambar 4.22. Hasil Pengujian ke-3 Pendeteksian Masker Pada Sampel Kedua

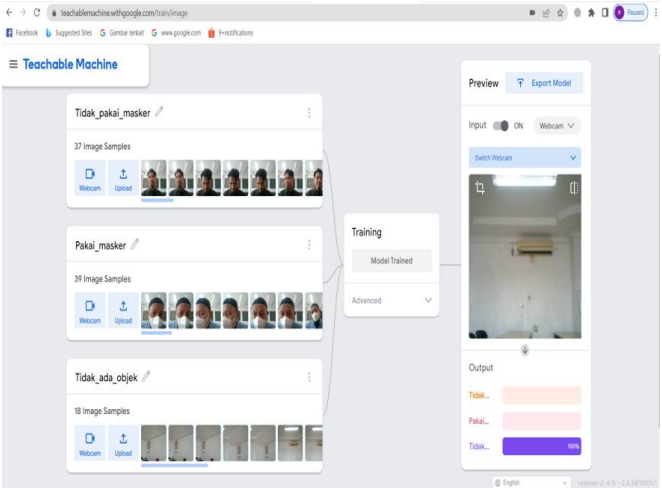


Gambar 4.23. Hasil Pengujian ke-4 Pendeteksian Masker Pada Sampel Kedua

Proses selanjutnya dilakukan pengujian pendeteksian dengan kondisi tidak ada objek, yaitu tidak ada orang di depan kamera. latar yang diambil dalam pengujian ada 2 (dua) buah. Hasil yang ditunjukkan pada *Class* 'Tidak_ada_objek' pada gambar 4.24 dan gambar 4.25 adalah 100%. Ini menunjukkan bahwa sistem dapat mengenali 'tidak ada objek (orang)' dengan baik dan akurat.

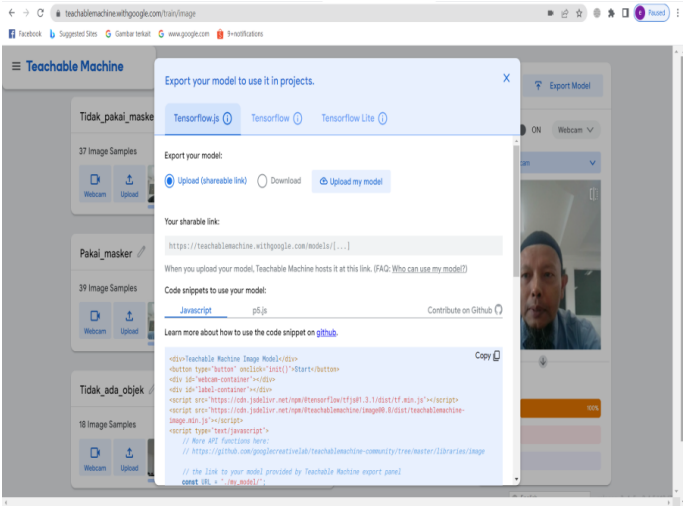


Gambar 4.24 Hasil Pengujian Pendeteksian 'Tidak_ada_objek" Pada Sampel pertama



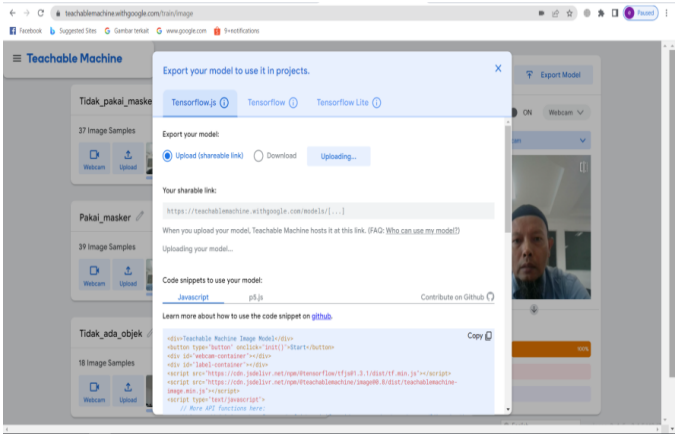
Gambar 4.25. Hasil Pengujian Pendeteksian 'Tidak_ada_objek" Pada Sampel Kedua

Tahap selanjutnya adalah mengupload model yang sudah ditraining dan diuji coba. Klik tombol '*upload my model*' seperti pada tampilan gambar 4.26.



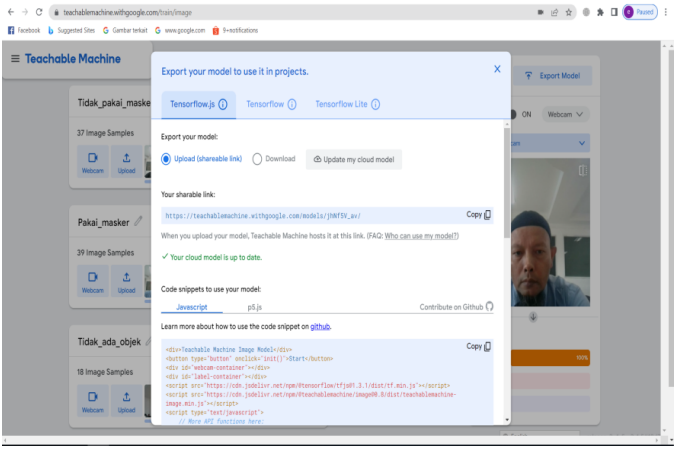
Gambar 4.26. Memilih '*Upload my model*'

Kemudian proses upload berlangsung seperti pada gambar 4.27. Lama proses upload tergantung pada jumlah gambar (*image*) yang direkam. Disamping itu juga tergantung kepada spesifikasi dari laptop yang digunakan.



Gambar 4.27. Proses '*Upload my model*' Sedang Berlangsung

Setelah proses 'uploading' selesai, maka pada '*Your sharable link*' terdapat link tempat penyimpanan data gambar (image) yang sudah ditraining tersebut seperti terlihat pada gambar 4.28. Alamat link url ini dapat kita gunakan selama masih tersimpan di database *cloud* google-nya.



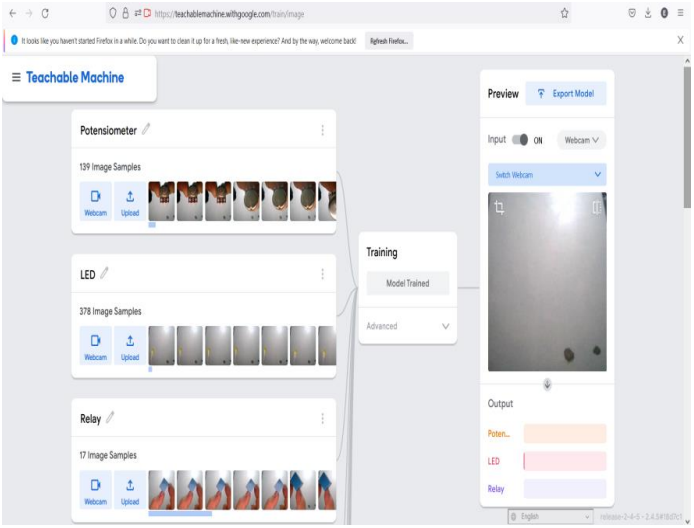
Gambar 4.28. Proses menyalin Alamat *Link Url* Deteksi Masker Dengan *Teachable Machine*

Untuk menyalin alamat ini, klik tombol '*Copy*' seperti terlihat pada gambar 4.28. Alamat ini nanti juga akan digunakan pada contoh aplikasi selanjutnya, yaitu deteksi masker yang dilengkapi dengan LED Indikator. Alamat hasil penekanan tombol tersebut : https://teachablemachine.withgoogle.com/models/jhNf5V_av/

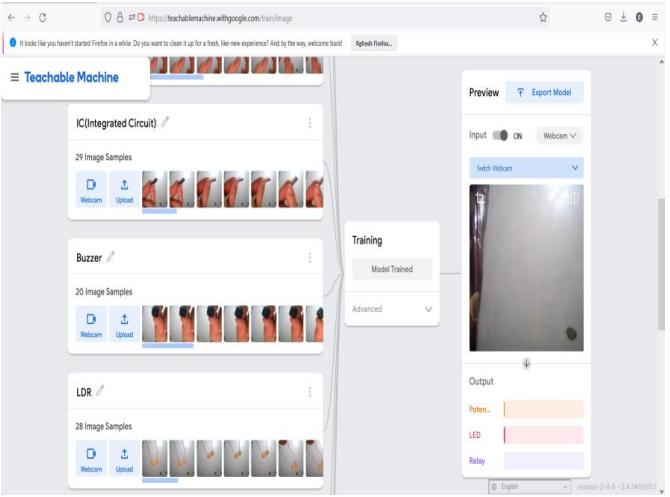
B. Deteksi Komponen Elektronika

Pada contoh kali ini *software* aplikasi *Teachable Machine* digunakan untuk mendeteksi komponen elektronika. Komponen yang akan dideteksi ada sebanyak 9 buah, antara lain *: Potentiometer, LED, Relay, Integrated Circuit (IC), Buzzer, LDR, Resistor, Dioda, Transistor*. Untuk itu jumlah *class* yang digunakan juga menjadi sebanyak 9 buah. Untuk merekam data (*collect data*) gambar (*image*) caranya sama dengan langkah pada contoh sebelumnya (deteksi mas-

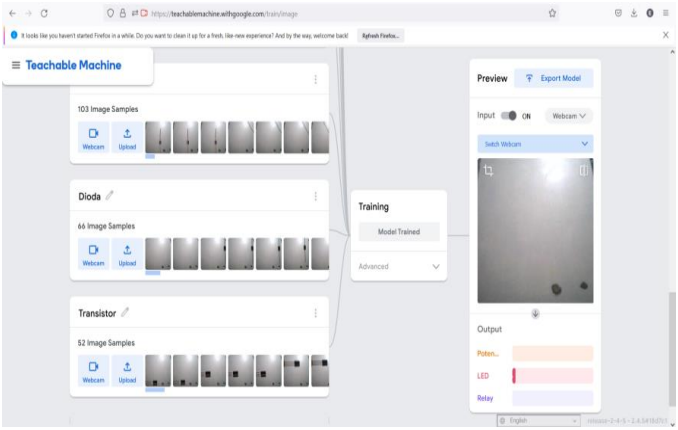
ker). Gambar 4.29 sampai dengan gambar 4.31 menunjukkan class dari masing-masing objek yang sudah direkam datanya.



Gambar 4.29. Hasil Perekaman Data Gambar (*Image*) *Class* : *Potensiometer*, *LED*, *Relay*

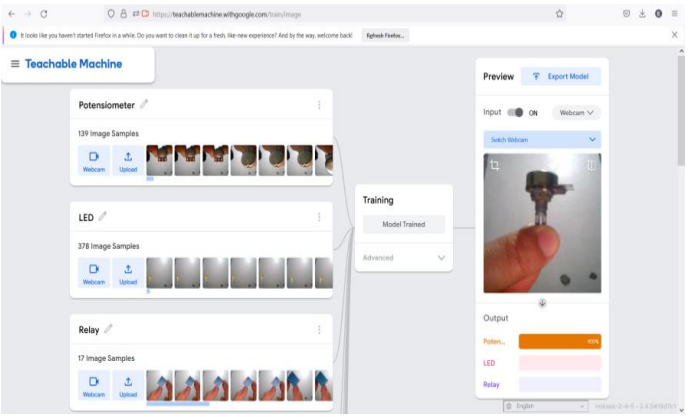


Gambar 4.30. Hasil Perekaman Data Gambar (*Image*) *Class* : IC (*Integrated Circuit*), *Buzzer*, LDR (*Light Dependent Resistor*)



Gambar 4.31. Hasil Perekaman Data Gambar (*Image*) *Class* : *Resistor*, *Dioda*, *Transistor*

Setelah proses *training* model dilakukan, maka selanjutnya dilakukan pengujian proses pendeteksian komponen. Pada pengujian pertama ini dilakukan proses pendeteksian *potensiometer*. Fisik dari *potensiometer* tersebut didekatkan ke kamera (*webcam*). Hasil pengujian dapat dilihat pada gambar 4.32 berikut ini.

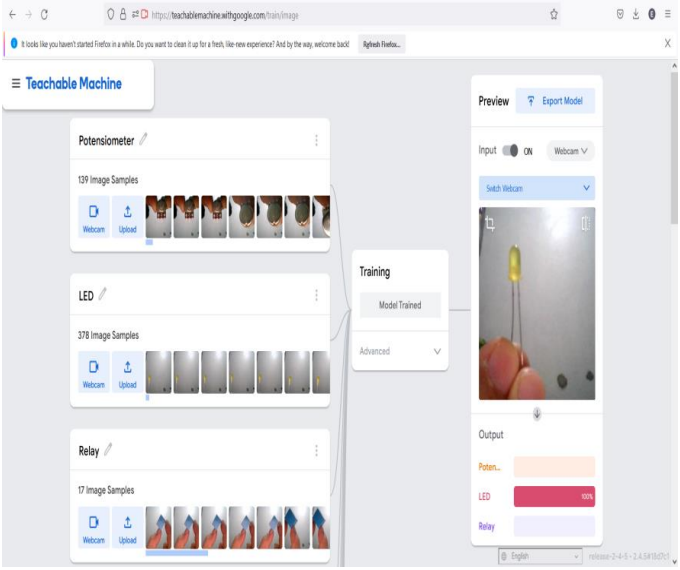


Gambar 4.32. Hasil Pengujian Komponen Pertama : *Potensiometer*

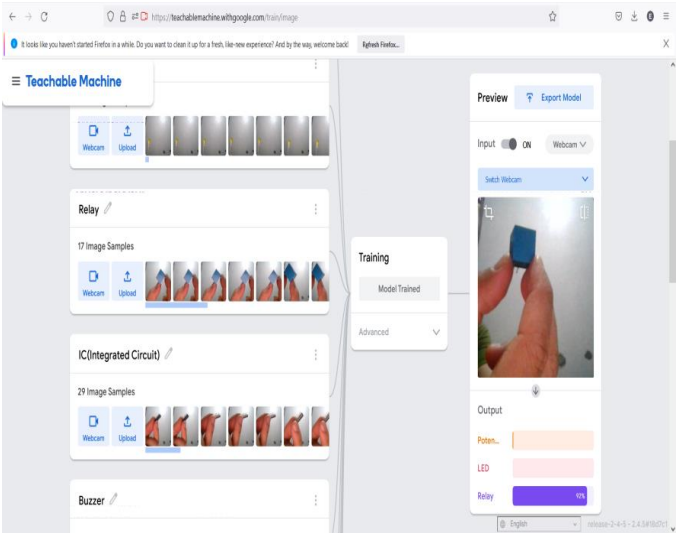
Dari hasil pengujian pada gambar 4.32 tersebut dapat dilihat bahwa *software* aplikasi *Teachable Machine* dapat mengenali komponen potensiometer dengan baik. Ini terbukti dari hasil pengujiannya 100%. Berdasarkan hasil pengujian yang telah dilakukan, jika pengujian tersebut bernilai lebih dari 90% maka besar kemungkinan komponen yang diuji memang betul-betul fisik dari komponen tersebut.

Pada pengujian selanjutnya dapat dilihat pada gambar 4.33 sampai dengan 4.40, secara berturut-turut komponen yang diuji adalah : *LED*, *Relay*, *Integrated Circuit (IC), Buzzer, LDR, Resistor, Dioda, Transistor*. Dari

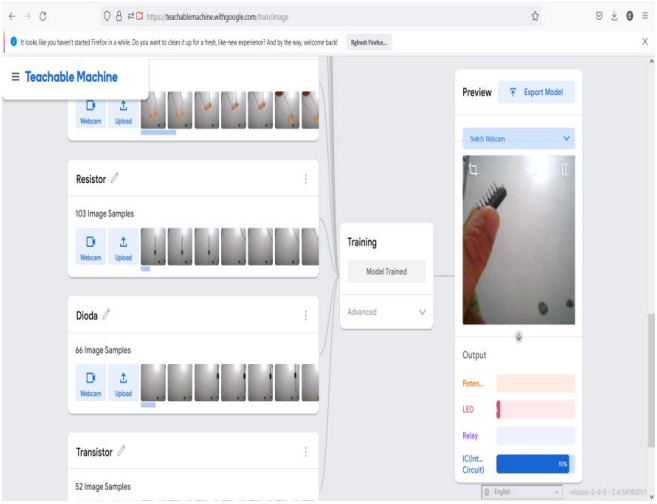
hasil pengujian dapat dilihat bahwa persentasenya lebih dari 90%, yang menandakan bahwa *sofware* aplikasi *Teachable Machine* dapat mengenali fisik dari komponen elektronika tersebut dengan baik dan akurat.



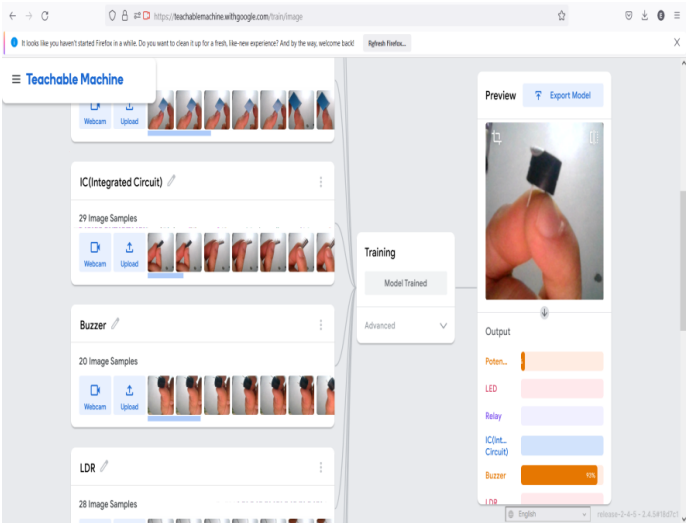
Gambar 4.33. Hasil Pengujian Komponen Kedua : LED (*Light Emitting Diode*)



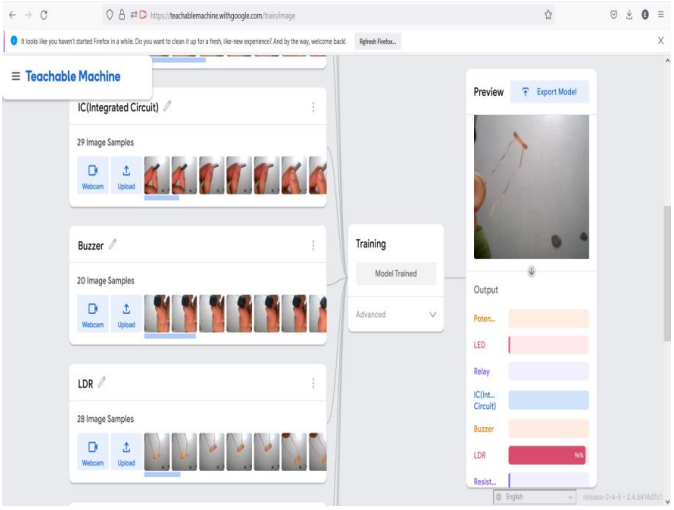
Gambar 4.34. Hasil Pengujian Komponen Ketiga : Relay



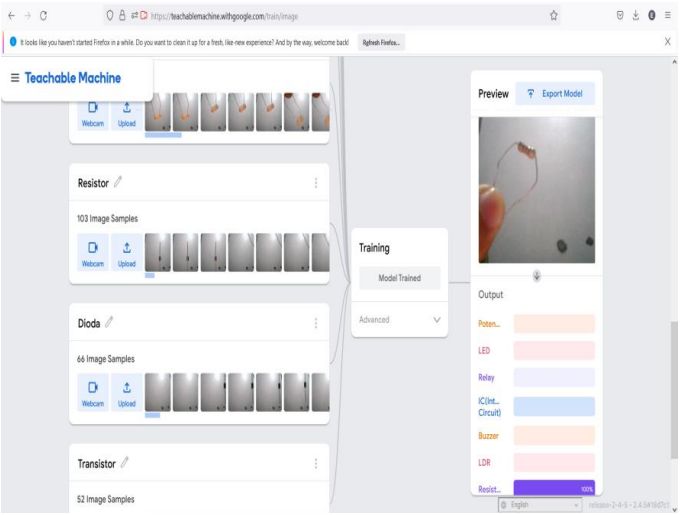
Gambar 4.35. Hasil Pengujian Komponen Keempat : IC (*Integrated Circuit*)



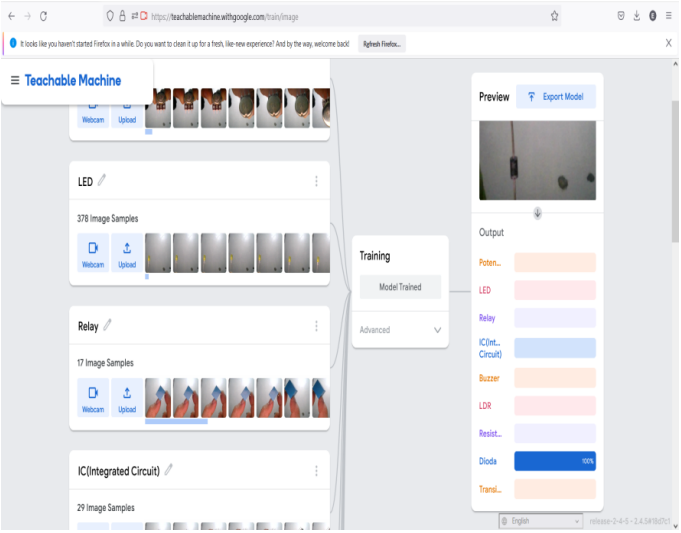
Gambar 4.36. Hasil Pengujian Komponen Kelima : Buzzer



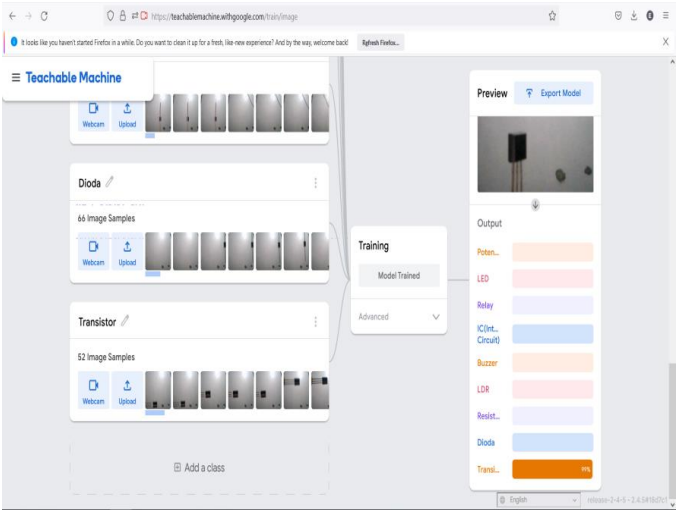
Gambar 4.37. Hasil Pengujian Komponen Keenam : LDR (Light Dependent Resistor)



Gambar 4.38. Hasil Pengujian Komponen Ketujuh : Resistor



Gambar 4.39. Hasil Pengujian Komponen Kedelapan : Dioda



Gambar 4.40. Hasil Pengujian Komponen Kesembilan : Transistor

C. Deteksi Penggunaan Masker Pada Wajah Dilengkapi Dengan Indikator LED (Light Emitting Diode)

Pada pengujian kedua ini, pendeteksian penggunaan masker dilengkapi dengan indikator LED. Dalam hal ini digunakan 3 buah warna LED yang mewakili kondisi objek yang dideteksi.

LED Merah : sebagai indikator Tidak_pakai_masker

LED Kuning : sebagai indikator Pakai_masker

LED Hijau : sebagai indikator Tidak_ada_objek

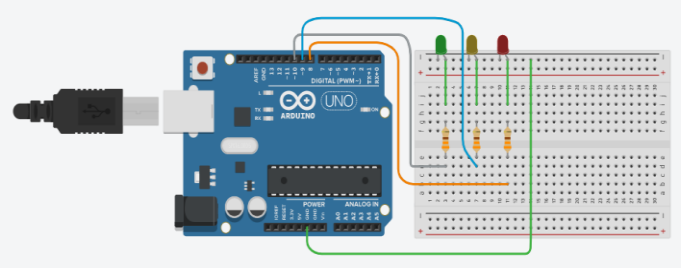
Pada dasarnya sistem ini juga bisa dilengkapi dengan *hardware* lain, seperti LCD (*Liquid Crystal Display*) sebagai penampil pesan, sehingga pengguna bisa membaca pesan yang tampil pada LCD tersebut. Sistem ini juga bisa dikembangkan dengan dilengkapi dengan pesan berupa

suara pada speaker, sehingga sistem menjadi lebih banyak kemampuan dan fasilitasnya.

Adapun komponen yang dibutuhkan untuk antara lain :

- Modul Arduino Uno R3 1 buah
- Kabel USB 1 buah
- Project Board 1 buah
- Resistor 330 ohm 3 buah
- LED 3 buah
- Kabel male to male 5 buah

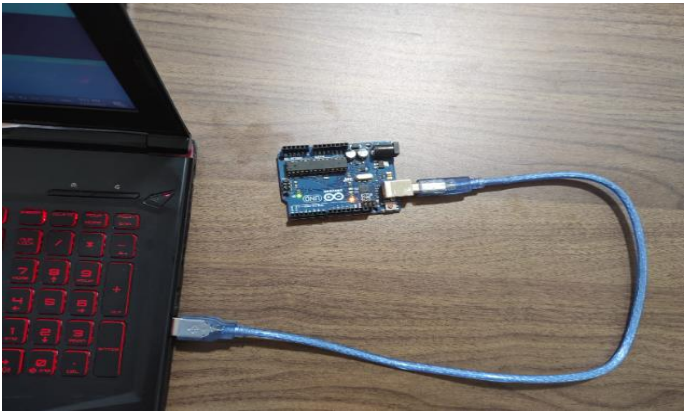
Komponen tersebut dirakit di atas project board, seperti terlihat pada gambar skema rangkaian 4.38 berikut ini.



Gambar 4.41. Skema Rangkaian Arduino Uno R3 dengan 3 buah LED Indikator

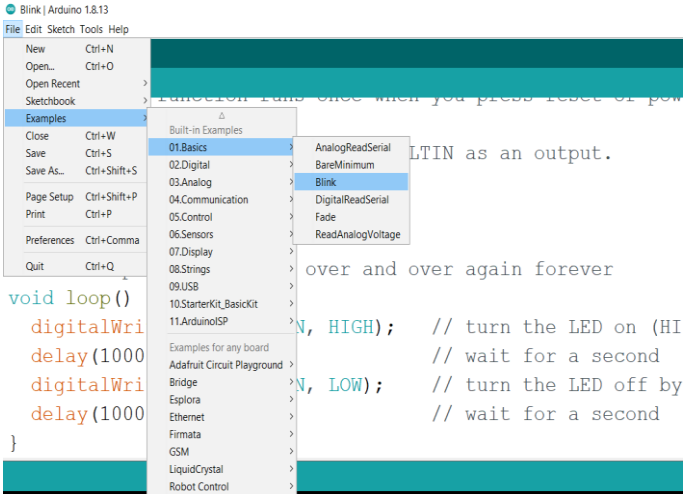
Sebelum rangkaian dirakit sebaiknya dilakukan pengujian terhadap Board Arduino Uno R3 tersebut. Hal ini untuk memastikan board tersebut berfungsi atau tidak. Caranya dengan membuat program sederhana kemudian meng-*upload* program tersebut ke board Arduino Uno R3. Untuk lebih jelasnya ikuti langkah-langkah berikut ini.

Terlebih dahulu hubungkan kabel USB dari Arduino Uno R3 ke laptop seperti terlihat pada gambar 4.42.



Gambar 4.42. Menghubungkan Arduino Uno R3 dengan Laptop Menggunakan Kabel USB

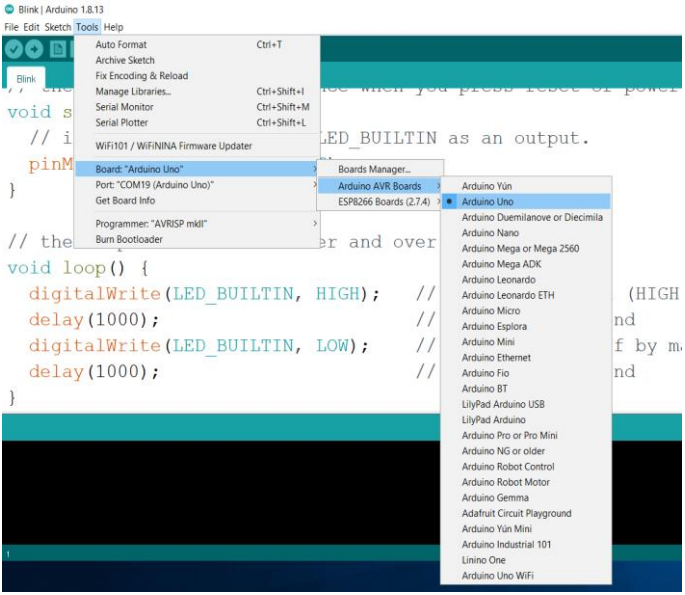
Buka aplikasi Arduino IDE. Dalam hal ini penulis menggunakan Arduino IDE 1.8.13. Kemudian pilih *Examples-Basics-Blink*, seperti tampilan pada gambar 4.43.



Gambar 4.43. Program (sketch) Blink Pada Arduino IDE

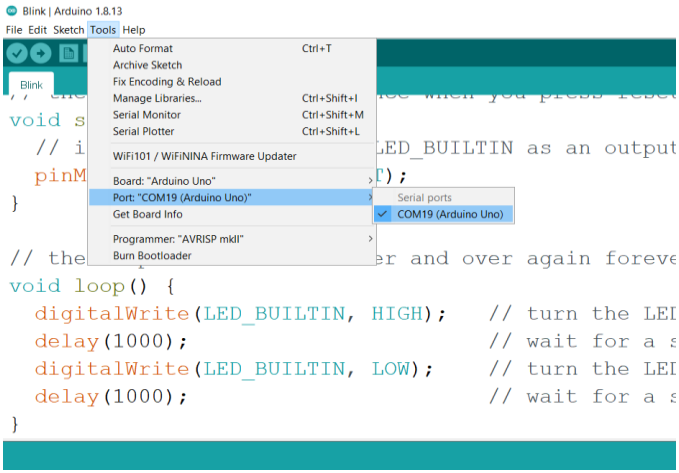
Program ini merupakan program sederhana untuk menguji apakah board Arduino Uno R3 nya bekerja dengan baik. Jika program ini berjalan dengan baik, maka LED internal (pin 13) pada Arduino akan menyala dan padam secara bergantian dalam durasi 1 detik.

Langkah selanjutnya pilih board Arduino Uno R3 seperti tampilan pada gambar 4.44 berikut ini.



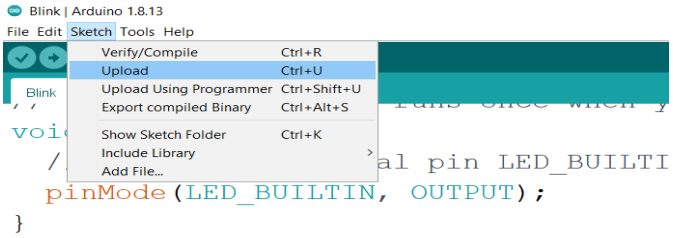
Gambar 4.44. Memilih Board Arduino Uno R3 pada Arduino IDE 1.8.13

Kemudian centang pada Port serial Arduino sesuai dengan nilai serial communication yang digunakan. Pada gambar 4.45 port serial yang digunakan adalah port nomor 19.



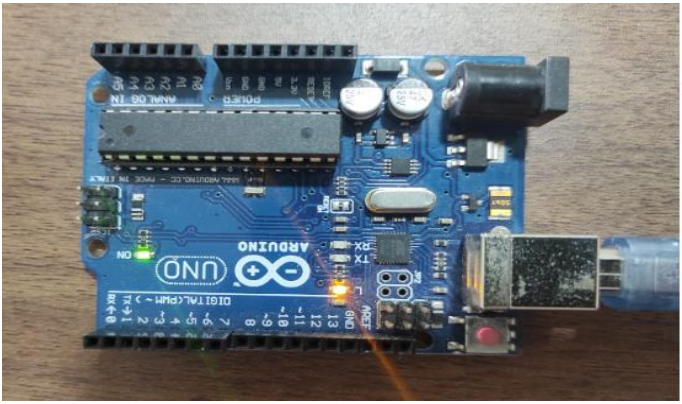
Gambar 4.45. Memilih Serial Port Pada Arduino IDE 1.18.3

Selanjutnya lakukan *upload sketch* program untuk pengaktifan LED berkedip (*blink*) dengan cara menekan tombol tanda pada posisi kiri atas. Bisa juga dengan menekan tombol Ctrl+U atau dengan memilih *Sketch-Upload* seperti terlihat pada gambar 4.46.



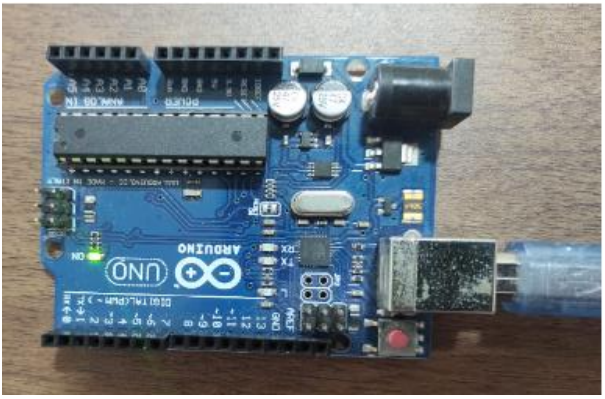
Gambar 4.46. Menguploadl Program ke Arduino menggunakan Arduino IDE 1.18.3

Setelah proses upload selesai, maka jika program berhasil LED internal (pin 13) menyala seperti tampilan pada gambar 4.47 berikut ini.



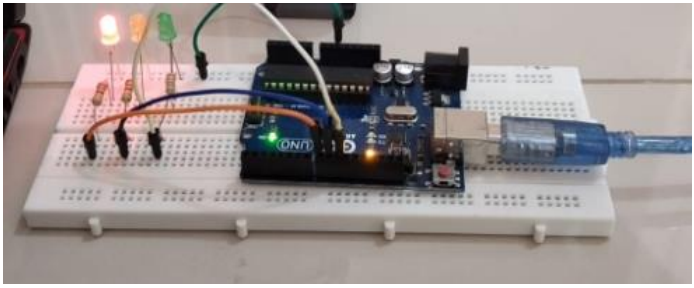
Gambar 4.47. LED Internal (pin 13) Arduino Menyala

Setelah 1 detik, LED *internal* (pin 13) padam seperti tampilan pada gambar 4.44 berikut ini.



Gambar 4.48. LED *Internal* (pin 13) Arduino Padam

Langkah selanjutnya adalah merangkai rangkaian LED seperti pada skema rangkaian pada gambar 4.48. Hasil perakitan rangkaian tersebut dapat dilihat pada gambar 4.48 berikut ini.



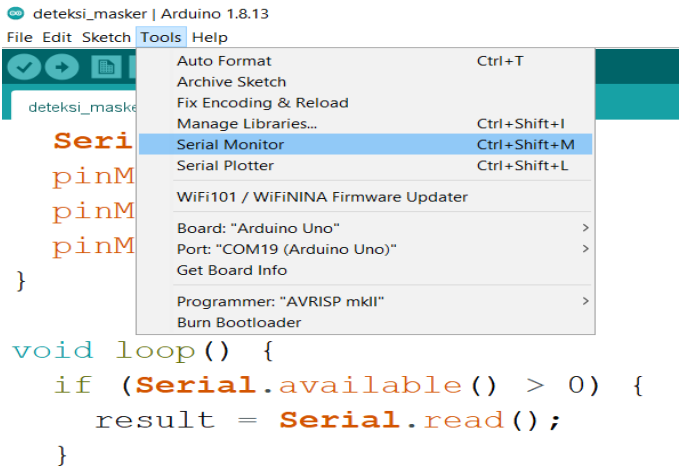
Gambar 4.49. Hasil Perakitan Rangkaian LED Indikator

Setelah gambar rangkaian upload program (*sketch*) berikut ini.

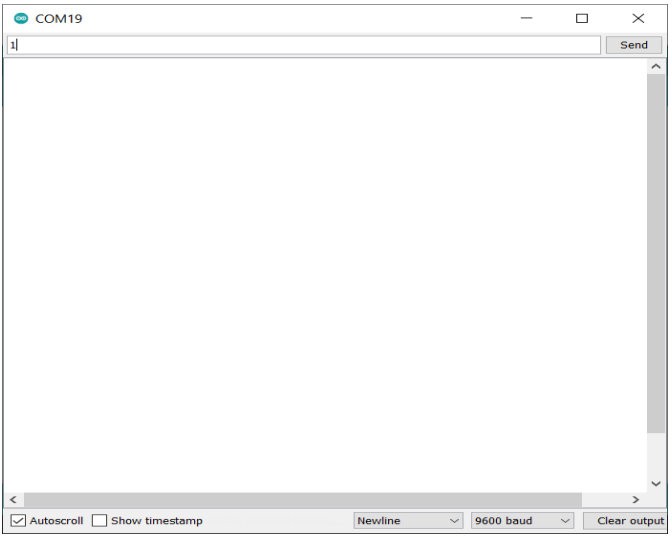
```
char result = '0';
void setup() {
Serial.begin(9600);
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);
}

void loop() {
 if (Serial.available() > 0) {
 result = Serial.read();
}
 switch (result) {
case '1':
 digitalWrite(8, HIGH);
```

```
 digitalWrite(9, LOW);
digitalWrite(10, LOW);
 break;
 case '2':

digitalWrite(8, LOW);
digitalWrite(9, HIGH);
digitalWrite(10, LOW);
break;
case '3':
digitalWrite(8, LOW);

digitalWrite(9, LOW);

digitalWrite(10, HIGH);

 break;

default:

digitalWrite(8, LOW);

 digitalWrite(9, LOW);

digitalWrite(10, LOW);

}

}
```

Setelah program (*sketch*) diupload, dilakukan pengujian pengiriman angka '1', '2' dan '3' melalui serial monitor. Caranya dengan menekan tombol Ctrl+Shift+M pada keyboard atau bisa juga dengan memilih '*Tools*'-'*Serial Monitor*' seperti terlihat pada gambar 4.50 berikut ini.



Gambar 4.50. Mengaktifkan Serial Monitor Arduino IDE

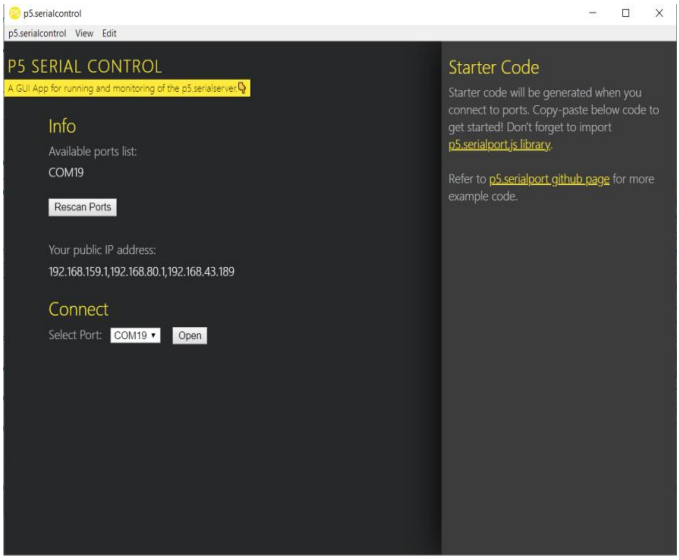


Gambar 4.51. Tampilan Serial Monitor Arduino IDE

Pastikan pada bagian kanan bawah serial monitor ini nilai *baud rate*-nya sama dengan yang ada di program (*sketch*) yaitu '9600 baud'. Jika sudah sama, langkah selanjutnya adalah mengirim angka '1' secara serial ke Arduino Uno R3. Caranya dengan mengetik angka '1' kemudian menekan tombol 'send' atau tombol enter. Jika pada saat angka '1' ditekan, LED indikator Merah menyala, maka program (*sketch*) yang dibuat serta komunikasi serial sudah bekerja dengan baik. Tampilan LED Merah yang aktif dapat dilihat pada gambar 4.51.

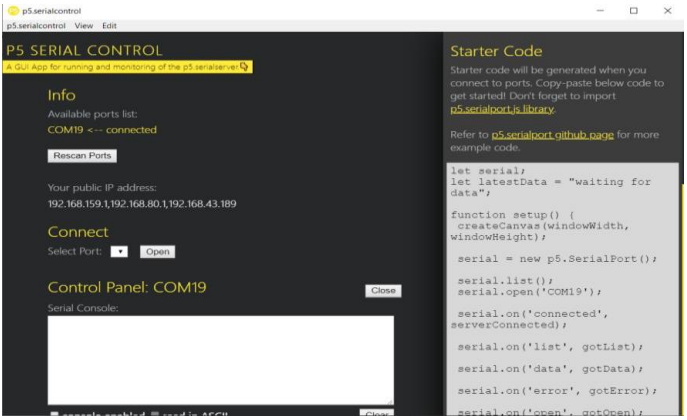
Pengujian selanjutnya yaitu dengan mengirimkan angka '2' dan angka '3'. Jika pada saat angka '2' dikirim, LED indikator Kuning yang menyala. Kemudian jika angka '3' yang dikirim maka LED indikator Hijau yang menyala, maka program (*sketch)* dan komunikasi serial secara keseluruhan sudah bekerja dengan baik.

Setelah semua proses pengujian tersebut selesai, tahapan selanjutnya adalah mengaktifkan *software* aplikasi P5 Serial Control. *Software* ini berfungsi menghubungkan laptop dengan alat melalui komunikasi serial. *Software* ini bersifat gratis dapat diunduh pada laman Github. Tampilan *software* tersebut dapat dilihat pada gambar 4.52 berikut ini.



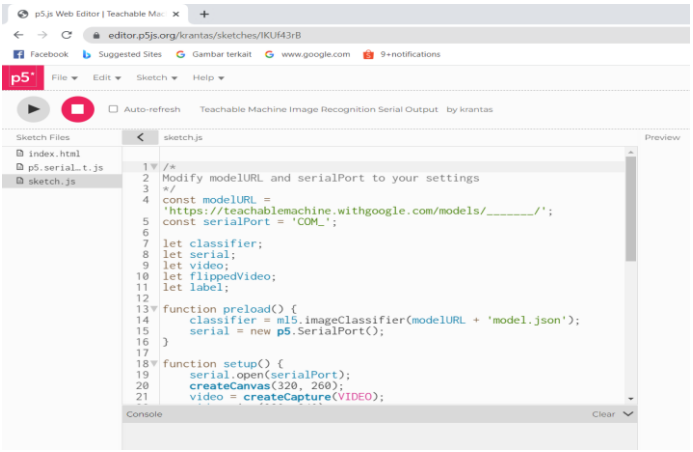
Gambar 4.52. Tampilan Software Aplikasi P5 Serial Control

Pada gambar 4.52 tersebut dapat dilihat bahwa serial communication yang tersedia adalah 'COM19'. Pilih 'COM19' pada '*Select Port*', kemudian klik tombol '*Open*'. Jika proses ini berhasil, maka pada 'Info' akan tampil pesan bahwa 'COM19 ← *connected*', seperti terlihat pada gambar 4.53 berikut ini.



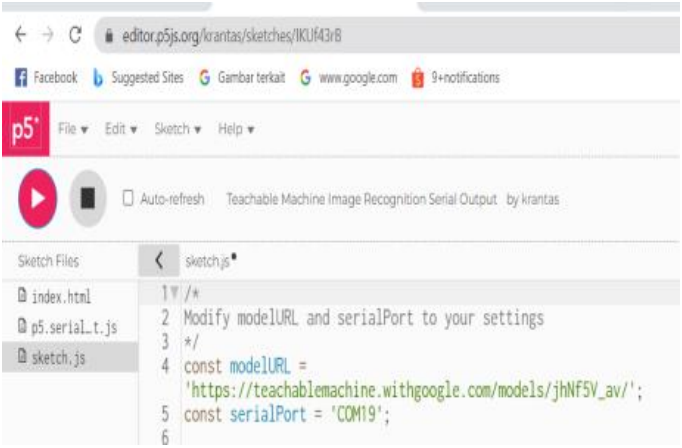
Gambar 4.53. Info Status 'COM19 <-- *connected*'

Langkah selanjutnya adalah masuk ke link URL : https://editor.p5js.org/krantas/sketches/IKUf43rB seperti terlihat pada gambar 4.53. Link ini berisi program yang menggunakan editor.p5js.org dan digunakan untuk menjalankan proses klasifikasi pada *Teachable Machine* serta dibagikan secara gratis pada laman Github.



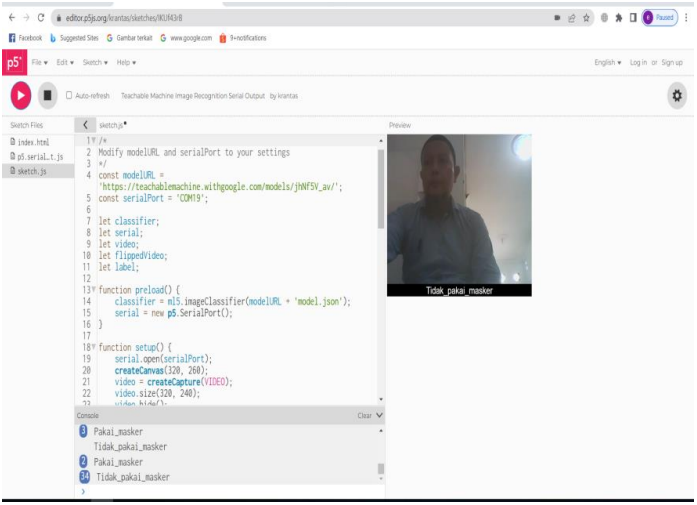
Gambar 4.54. *Link* Editor.p5js.org

Pada gambar 4.54 tersebut terdapat 3 (tiga) buah program (*sketch*), yaitu ; '*index.html*', '*p5.serialport.js*', '*sketch.js'*. untuk keperluan contoh aplikasi yang ketiga ini, program yang perlu dimodifikasi hanyalah '*sketch.js*'. Pada '*sketch*.js' tersebut dapat dilihat perintah setting konstanta untuk '*modelURL*' dan '*serialPort'*. Masukkan alamat URL dari '*model'* yang sudah disimpan di *cloud* google pada contoh penggunaan *Teachable Machine* pertama. Kemudian isikan '*serialPort*' sesuai dengan nilai port yang ada pada P5 Serial Control, yaitu COM19, sehingga program pada '*sketch.js*' menjadi sebagai berikut.



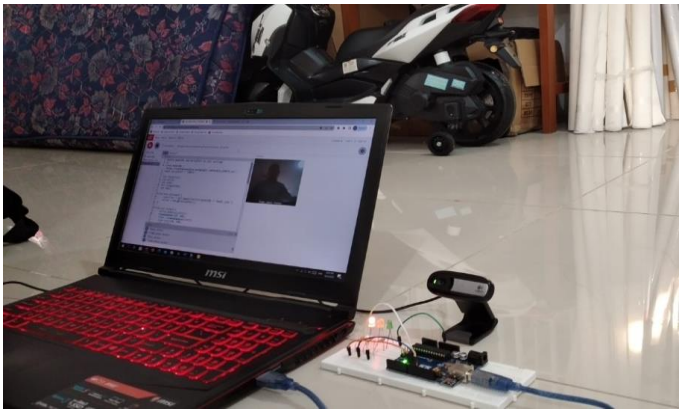
Gambar 4.55. Modifikasi alamat '*modelURL*' dan '*serialPort*' pada '*sketch.js*'

Selanjutnya jalankan program dengan menekan tombol (tombol '*play*').



Gambar 4.56. Hasil Pengujian Pertama (Tidak_pakai_masker)

Pada gambar 4.56 dapat dilihat pada saat orang yang berada di depan kamera tidak menggunakan masker, maka keterangan yang ada di bawah tampilan kamera tersebut adalah ; 'Tidak_pakai_masker' dan pada bagian kiri bawah nampak bahwa hasil pendeteksian 'Tidak_ pakai_masker' sudah dilakukan sebanyak 34 kali tanpa ada perpindahan ke *class* yang lain. Ini membuktikan bahwa sistem bekerja dengan baik pada pendeteksian orang yang tidak menggunakan masker.

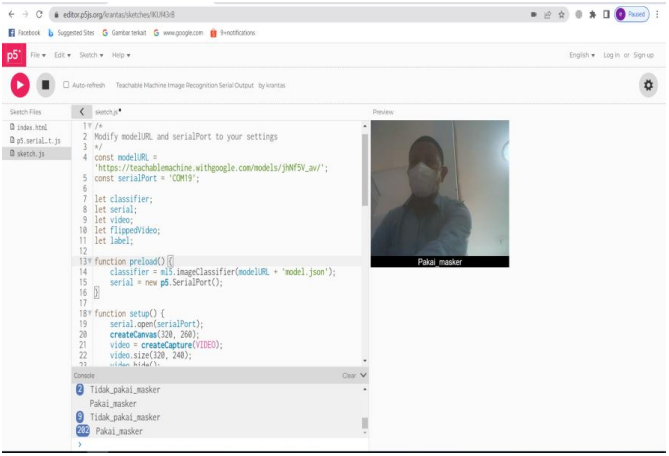


Gambar 4.57. LED Indikator Merah Menyala Saat Ada Orang Yang Tidak Memakai Masker

Kemudian kalau kita lihat pada gambar 4.57 pada saat ada orang yang tidak menggunakan masker, maka LED Indikator Merah menyala. Seperti yang telah dijelaskan sebelumnya, LED Indikator ini bisa dikembangkan dengan penambahan komponen yang lain, seperti LCD (*Liquid Crystal Display)*. Dengan menggunakan LCD ini, maka pada saat LED Indikator Merah menyala, pada LCD akan tampil pesan 'Anda Tidak Menggunakan Masker !'. Bisa juga dengan penambahan komponen penghasil suara pada *speaker*, sehingga pada saat pengguna tidak menggunakan masker, akan keluar pesan pada speaker 'Anda Tidak Menggunakan Masker!'.

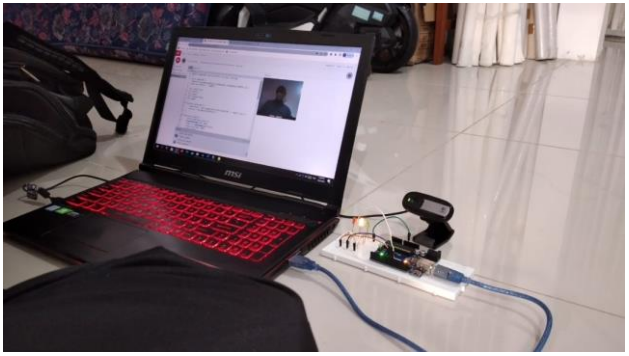
Selanjutnya pada gambar 4.58 dapat dilihat pada saat orang yang berada di depan kamera sedang menggunakan masker, maka keterangan yang ada di bawah tampilan kamera tersebut adalah ; 'Pakai_masker' dan pada bagian kiri bawah nampak bahwa hasil pendeteksian 'Pakai_masker' sudah dilakukan sebanyak 202 kali. Ini

membuktikan bahwa sistem bekerja dengan baik pada pendeteksian orang yang menggunakan masker.



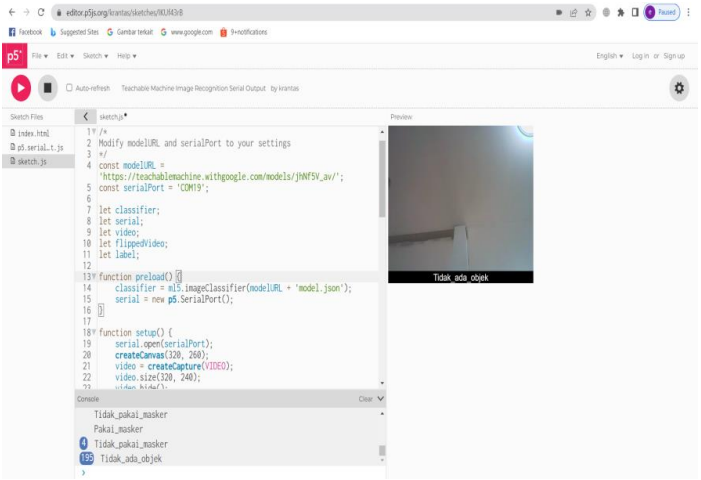
Gambar 4.58. Hasil Pengujian Kedua (Pakai_masker)

Kemudian kalau kita lihat pada gambar 4.59 pada saat ada orang yang menggunakan masker, maka LED Indikator Kuning menyala.



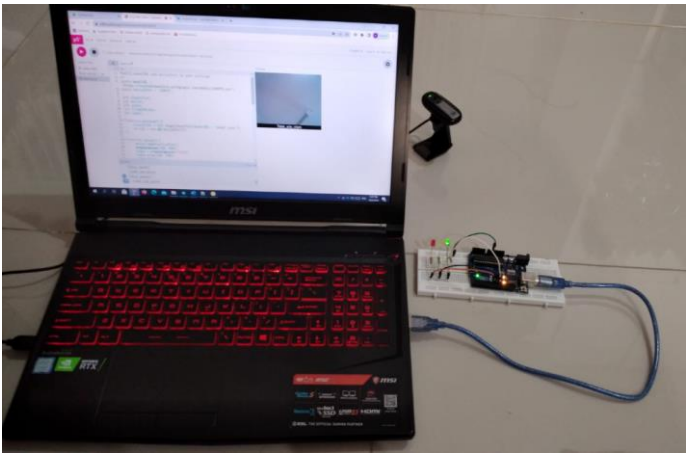
Gambar 4.59. LED Indikator Kuning Menyala Saat Ada Orang Yang Memakai Masker

Pada pengujian selanjutnya pada gambar 4.60 dapat dilihat pada saat tidak ada orang yang berada di depan kamera, maka keterangan yang ada di bawah tampilan kamera tersebut adalah ; 'Tidak_ada_objek'.



Gambar 4.60. Hasil Pengujian Ketiga (Tidak_ada_objek)

Pada bagian kiri bawah gamar 4.60 tersebut nampak bahwa hasil pendeteksian 'Tidak_ada_objek' sudah dilakukan sebanyak 195 kali. Ini membuktikan bahwa sistem bekerja dengan baik pada pendeteksian saat tidak ada orang (objek) di depan kamera. Kemudian kalau kita lihat pada gambar 4.61 pada saat tidak ada objek, maka LED Indikator Hijau menyala.



Gambar 4.61. LED Indikator Hijau Menyala Saat Tidak Ada Orang (Objek) Di Depan Kamera

Pada gambar 4.61 dapat dilihat bahwa kamera dihadapkan ke atas plafon untuk mendapatkan kondisi tidak ada orang (objek) di depan kamera. Hal ini karena saat pengujian, penulis berada di depan komputer (laptop).

BAB V

Contoh Aplikasi Deep Learning dengan NVIDIA Jetson Nano

A. Contoh Program *Python* Dengan *openCV Library* Pada NVIDIA Jetson Nano

Sebelum memulai dengan contoh aplikasi deep learning, disini dilakukan percobaan program *python* sederhana antara lain ; program menampilkan file gambar, program untuk mengaktifkan kamera USB dan program untuk mengambil dan menyimpan gambar dari kamera. Sebelum membuat program tersebut, *install openCV library* dengan mengetikkan perintah :

$ sudo apt-get install python3-opencv

- **Menampilkan file gambar**

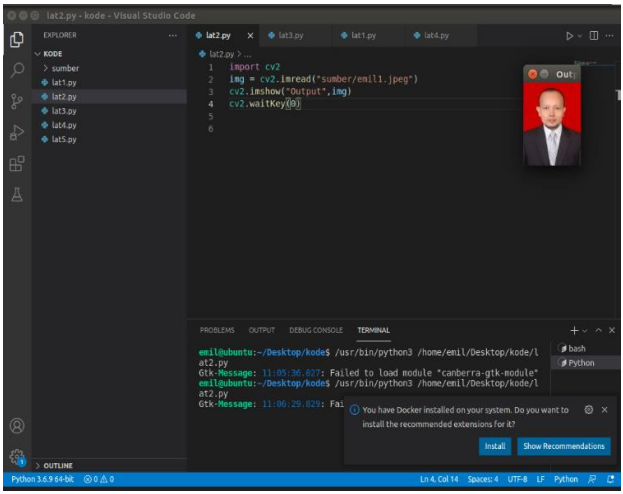
  Untuk menampilkan file gambar program pythonnya sebagai berikut.

```
import cv2
img = cv2.imread("sumber/emil1.jpeg")
cv2.imshow("Output",img)
cv2.waitkey(0)
```

Program di atas memanfaatkan *library openCV*. Dengan perintah *import cv2* pada baris pertama maka *library openCV* dipanggil. Selanjutnya pada baris kedua dilakukan pembacaan file '*emil1.jpeg*'. file ini terletak pada folder 'sumber' sehingga pembacaannya harus menyertakan folder 'sumber' tersebut. Baris ketiga

berfungsi menampilkan file gambar yang telah dibaca. Selanjutnya pada baris keempat program akan me-nunggu penekanan sembarang tombol pada *keyboard* untuk mengakhiri program ini.

Setelah program dibuat klik tombol *running* (▶). Pada gambar 5.1 dapat dilihat tampilan program yang telah berhasil menampilkan foto dengan nama file'emil1.jpeg'.



Gambar 5.1. Tampilan Program Menampilkan File Gambar

- **Mengaktifkan Kamera USB Secara *Realtime***

  Untuk mengaktifkan kamera USB secara realtime, langkah pertama adalah menghubungkan konektor USB kamera pada port USB NVIDIA Jetson Nano. Selanjutnya buat program *python* berikut ini.

  import numpy as np
  import cv2
  #change camera no

```
cap = cv2.VideoCapture(1)
while(True):
        ret, frame = cap.read()
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) &0xff == ord('q'):
         break
  cap.release()
  cv2.destroyAllWindows()
```

Program tersebut akan berjalan terus menerus sampai kita menekan tombol 'q'. Untuk menjalankan program tersebut ketikkan perintah ini di Terminal :

$ sudo python3 camera-usb-demo.py

Adapun tampilan program dapat dilihat pada gambar 5.2. Untuk menutup program, tekan tombol 'q'.

Gambar 5.2. Tampilan Program Mengaktifkan Kamera USB Secara *Realtime*

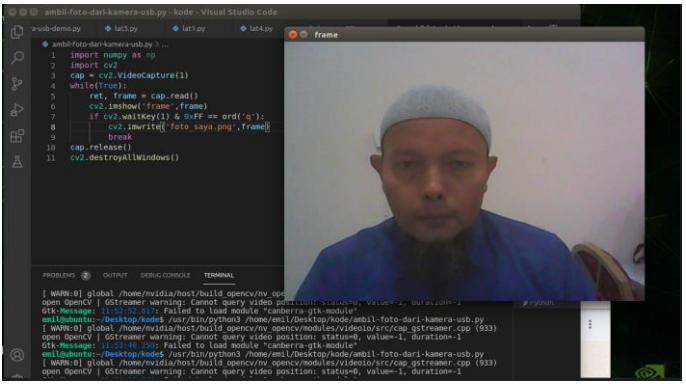
- **Mengambil dan Menyimpan Gambar dari Kamera USB**

  Untuk mengambil gambar dari Kamera USB, kita dapat menggunakan fungsi *cv2.imwrite().* Fungsi ini akan menyimpan frame video ke dalam sebuah file. Setelah penekanan tombol 'q' program akan berhenti setelah sebelumnya menyimpan frame tersebut dalam bentuk file. Pada kasus ini, file gambar akan disimpan dengan nama 'foto_saya.png'.

  Program tersebut dimodifikasi dari program '*kamera-usb-demo.py*'. *Listing* program selengkapnya sebagai berikut.

```
import numpy as np
import cv2
cap = cv2.VideoCapture(1)
while(True):
ret, frame = cap.read()
cv2.imshow('frame',frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
cv2.imwrite('mypicture.png',frame)
break
cap.release()
cv2.destroyAllWindows()
```

Adapun tampilan program setelah dijalankan dapat dilihat pada gambar 5.3.



Gambar 5.3. Tampilan Program Mengambil dan Menyimpan Gambar dari Kamera USB

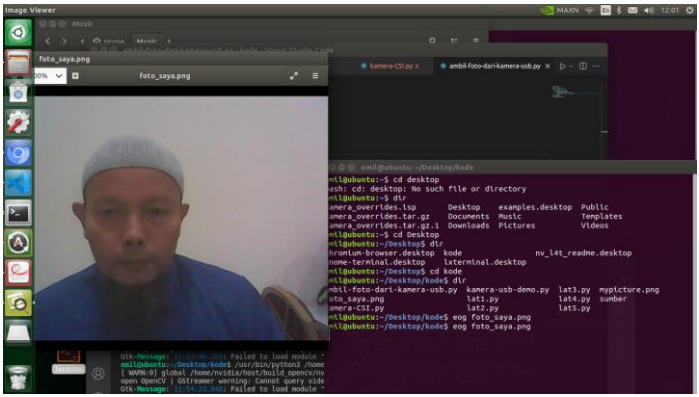
File gambar yang sudah tersimpan dapat dibuka dan dilihat dengan perintah :

$ eog foto_saya.png

Tampilan file yang telah tersimpan dapat dilihat pada gambar 5.4. Pada bagian kanan bawah terlihat bahwa untuk masuk ke folder tersebut harus menggunakan perintah :

$ cd desktop
$ cd kode

Hal ini karena file gambar tersebut terletak pada folder 'Desktop/kode'.



Gambar 5.4. Menampilkan File Gambar

B. Contoh Aplikasi *Deep Learning* Pada NVIDIA Jetson Nano Menggunakan Bahasa Pemrograman *Python*

NVIDIA Jetson Nano dirancang untuk membangun berbagai aplikasi cerdas. Dengan GPU 128 core (*Graphic Processing Unit*), NVIDIA Jetson Nano dapat digunakan untuk melakukan komputasi berbasis *machine learning*.

1. Mengatur *Jetson Inference Library*

Untuk contoh aplikasi deep learning ini kita ambil dari Hello API dengan *Jetson Library Inference* untuk membangun aplikasi machine learning. Lebih lanjut tentang *library Jetson Inference* dapat diakses pada https://github.com/dusty-nv/jetson-inference. Sebelum menyiapkan library ini, kita perlu menginstal semua *library* yang diperlukan. Caranya adalah dengan mengetik beberapa perintah berikut ini.

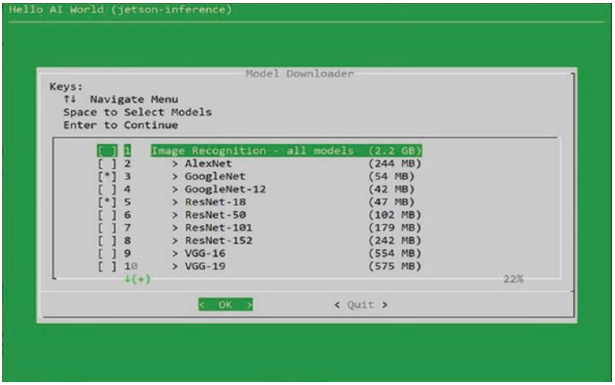
$ sudo apt-get update

$ sudo apt-get install git cmake
$ sudo apt-get install libpython3-dev python3-numpy

Selanjutnya, buat *library Jetson Inference* dari kode sumber. Lakukan *kloning Jetson Inference* dari *Github*. Setelah itu, buat *library* dengan mengetikkan beberapa perintah berikut ini.

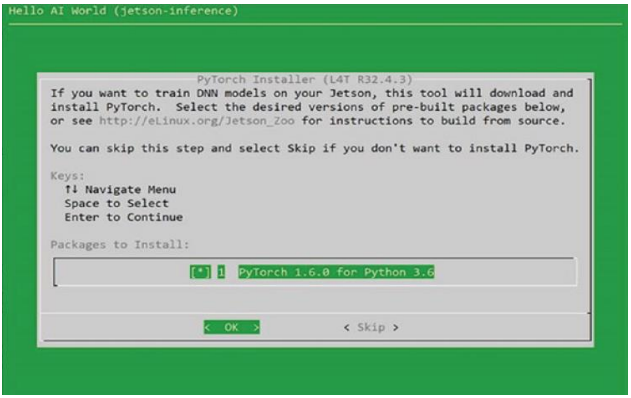
$ git clone https://github.com/dusty-nv/jetson-inference
$ cd jetson-inference
$ git submodule update --init
$ mkdir build
$ cd build
$ cmake ../

Selama proses instalasi ini, akan muncul dialog seperti yang ditunjukkan pada gambar 5.5. Pilih beberapa model yang ingin digunakan dalam aplikasi. Pastikan data penyimpanan pada microSD NVIDIA Jetson Nano masih tersedia.



Gambar 5.5. Tampilan Pilihan Model Untuk *Traning* Data

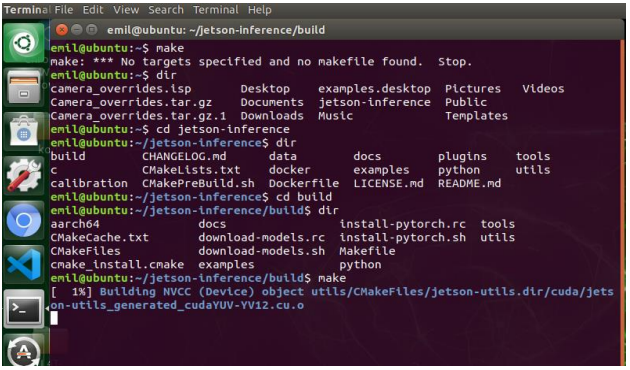
Kita juga akan ditanya untuk menginstal *PyTorch*, seperti yang ditunjukkan pada Gambar 5.6. Pilih *PyTorch* untuk versi *Python* yang kita gunakan.



Gambar 5.6. Menginstall *Pytorch* Untuk *Python*

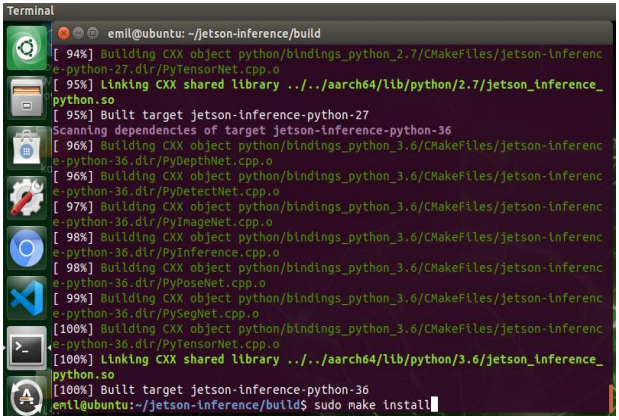
Proses ini membutuhkan waktu beberapa menit untuk diselesaikan. Setelah selesai masuk ke folder '*jetson-inference/build*'. Kemudian, *instal library* tersebut menggunakan beberapa perintah berikut ini.

$ make



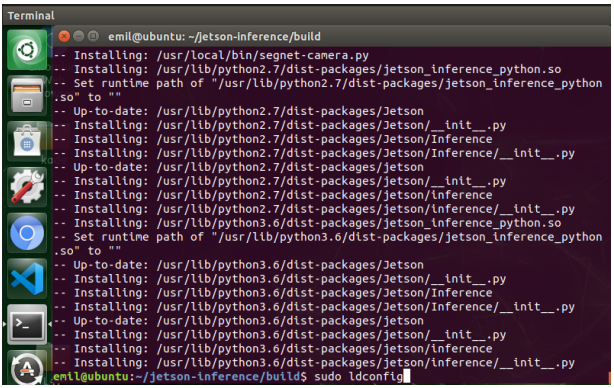
Gambar 5.6. Tampilan Hasil Eksekusi *$ make*

$ sudo make install



Gambar 5.7. Tampilan Hasil Eksekusi
$ sudo make install

$ sudo ldconfig



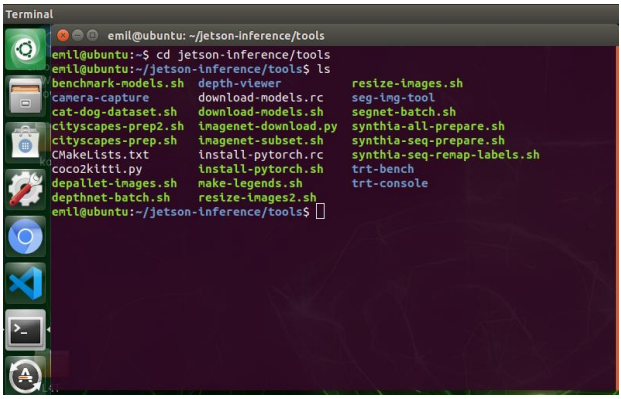
Gambar 5.8. Tampilan Hasil Eksekusi
$ sudo ldconfig

Sekarang *Jetson-inference* sudah siap untuk digunakan. Terkadang kita ingin memodifikasi atau menambahkan beberapa data model. Untuk itu kita dapat memanggil model program yang dapat *didownlo-*

ad langsung. Untuk melakukan pemanggilan ini tersedia *tools* yang berada pada folder '*jetson-inference/tools*' seperti yang ditunjukkan pada gambar 5.9. Ketikkan perintah berikut ini.

$ cd jetson-inference/tools
$ ./download-models.sh

Selanjutnya perintah akan dieksekusi dan tampilan eksekusi tersebut dapat kita lihat pada gambar 5.9. Pada gambar 5.9 dapat dilihat bahwa tools ini akan mengunduh model-program secara langsung.



Gambar 5.9. Tools Program pada Jetson Inference

2. *Image Recognition* Menggunakan NVIDIA *Jetson Nano*

Untuk mengingat (*recognize*) image (objek) pada NVIDIA Jetson Nano, kita dapat menggunakan contoh program yang sudah ada pada folder '*jetson-inference/python /examples*. Nama programnya adalah *my-recognition.py.* Listing programnya sebagai berikut.

import jetson.inference
import jetson.utils

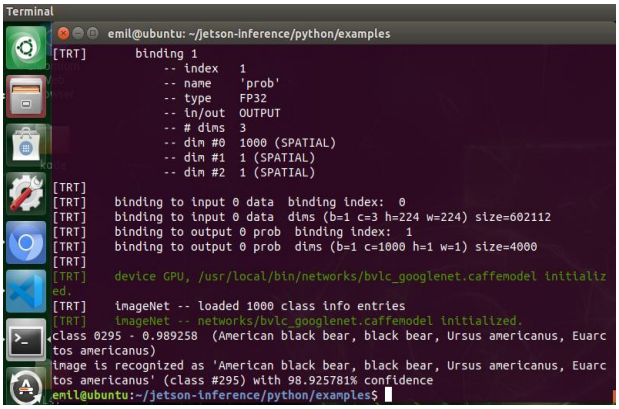
```
import argparse
# parse the command line
parser = argparse.ArgumentParser()
parser.add_argument("filename", type=str,
help="filename of the
image to process")
parser.add_argument("--network", type=str,
default="googlenet",
help="model to use, can be: googlenet, resnet-18, ect.")
opt = parser.parse_args()
# load an image (into shared CPU/GPU memory)
img = jetson.utils.loadImage(opt.filename)
# load the recognition network
net = jetson.inference.imageNet(opt.network)
# classify the image
class_idx, confidence = net.Classify(img)
# find the object description

class_desc = net.GetClassDesc(class_idx)
# print out the result
print("image is recognized as '{:s}' (class #{:d}) with
{:f}%
confidence".format(class_desc, class_idx, confidence *
100))
```

Untuk mencoba program tersebut, gunakan file gambar '*black_bear.jpg*' dari '*jetson-inference/examples /my-recognition*. Salin file '*black_bear.jpg*' ke dalam folder tersebut. Selanjutnya ketik program berikut ini.

```
$ sudo python3 my-recognition.py black_bear.jpg
```

Saat perintah tersebut dijalankan untuk pertama kali, ini akan memakan waktu beberapa menit

karena aplikasi ini akan melakukan proses pelatihan (training). Program tersebut juga melakukan deteksi objek. Hasilnya dapat dilihat pada gambar 5.10 :



Gambar 5.10. Hasil eksekusi program *my-recognition.py*

Pada gambar 5.10 di bagian bawah dapat dilihat bahwa hasil pendeteksiannya memiliki prediksi dengan keyakinan sebesar 98.925781% bahwa gambar yang dideteksi adalah '*American black bear, black bear, Ursus americanus, Euarctos americanus*'. Gambar yang dimuat dalam file '*black_bear.jpg*' ini berada pada *Class* nomor #295 dari 1000 class image (gambar) yang dimuat ke dalam proses *training*.

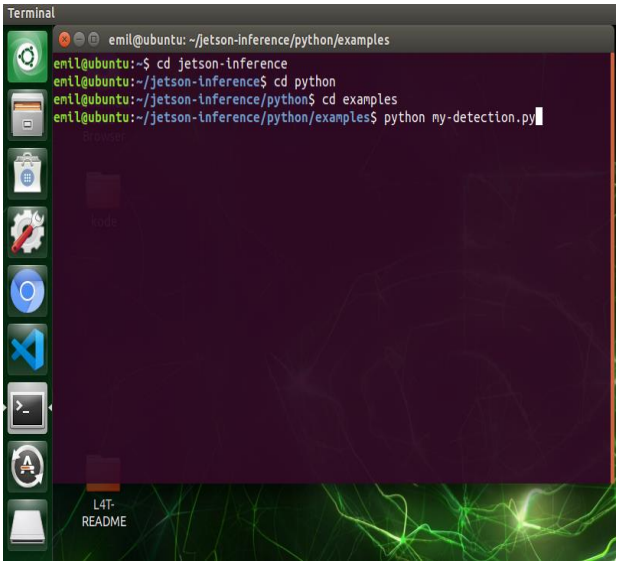
3. Deteksi Objek Secara *Realtime* Menggunakan NVIDIA *Jetson Nano*

Untuk mendeteksi objek secara *realtime* pada NVIDIA Jetson Nano, kita dapat menggunakan contoh program yang sudah ada pada folder '*jetson-inference/python /examples*. Ketikkan perintah berikut ini.

$ cd jetson-inference

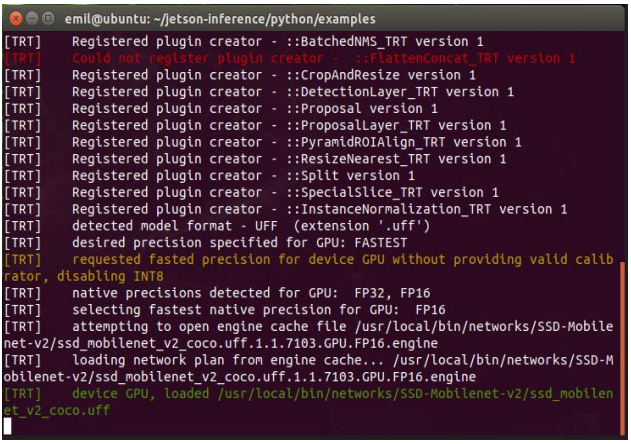
$ cd python
$ cd examples
$ cd python my-detection.py

Tampilan perintah-perintah tersebut dapat dilihat pada gambar 5.11.

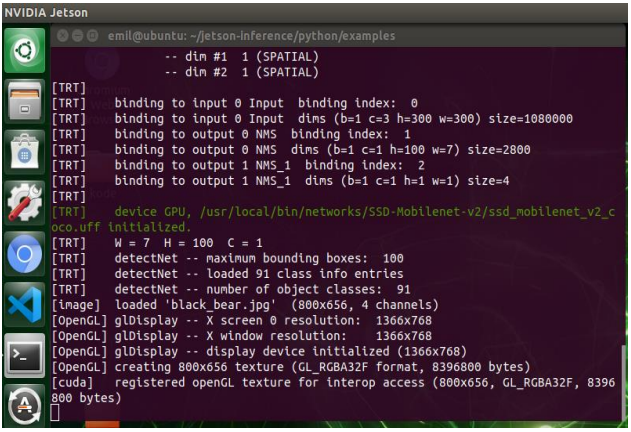


Gambar 5.11. Tampilan Perintah Untuk Menjalankan *Program my-detection.py*

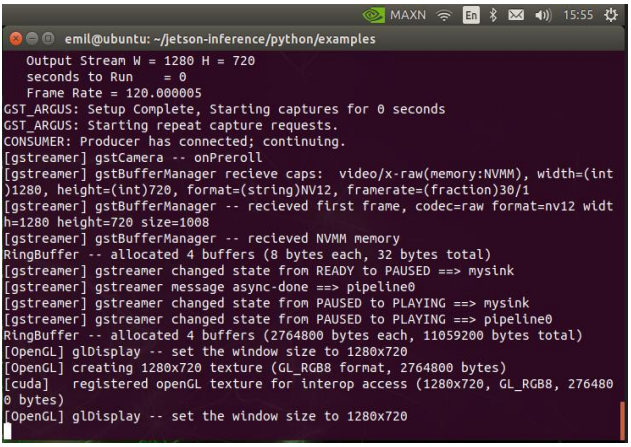
Setelah perintah pada baris terakhir dijalankan maka NVIDIA akan mengeksekusi perintah tersebut. Tampilan eksekusi dapat dilihat pada gambar 5.11 sampai dengan gambar 5.12.



Gambar 5.12. Tampilan Eksekusi Pertama Pada *Program my-detection.py*

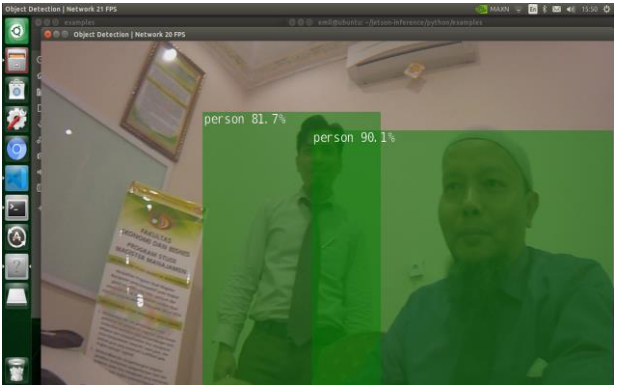


Gambar 5.13. Tampilan Eksekusi Kedua Pada *Program my-detection.py*



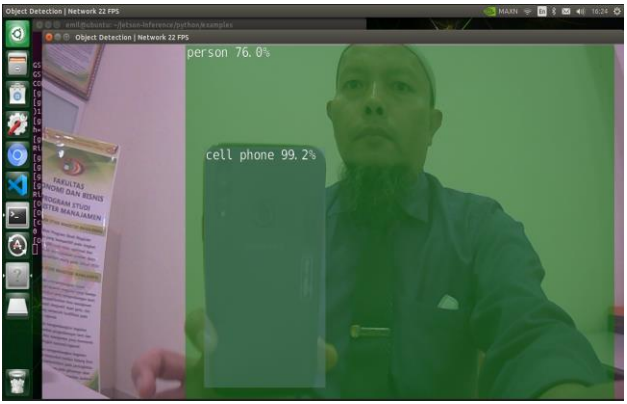
Gambar 5.14. Tampilan Eksekusi Ketiga
Pada *Program my-detection.py*

Selanjutnya program akan melakukan pendetek-sian objek yang berada di depan kamera secara *realtime*. Hasil pendeteksian dari program '*my-detection.py*' dapat dilihat pada gambar 5.15 sampai dengan gambar 5.17.



Gambar 5.15. Hasil Pengujian Pertama :
Deteksi Orang

Pada pengujian pertama, program mampu mendeteksi orang (*person*) dengan persentase pendeteksian 81.7% dan 90.1%.



Gambar 5.16. Hasil Pengujian Pertama : Deteksi Orang dan 1 Buah Benda (Handphone)

Pada pengujian kedua, program mampu mendeteksi orang (person) dan benda (1 buah handphone) dengan persentase pendeteksian 76.0% dan 99.2%.

Gambar 5.17. Hasil Pengujian Ketiga : Deteksi Orang dan 3 Buah Benda (Cangkir, *Mouse* dan *Keyboard*)

Pada pengujian ketiga, program mampu mendeteksi orang (person) dan 3 buah benda (Cangkir (cup), Mouse dan Keyboard) dengan persentase pendeteksian 95.3% dan 71.7%, 56.4% dan 88.2%.

Jika dilihat dari kesemua hasil pengujian tersebut dapat dikatakan sistem pendeteksian benda bekerja dengan baik. Namun persentase objek yang dideteksi berbeda-beda. Hal ini tergantung kepada banyak faktor, diantaranya jumlah sampel data pelatihan (*training*) objek tersebut dan kesamaan objek dengan sampel yang dilatih.

DAFTAR PUSTAKA

A. H. Ahamad, N. Zaini, and M. F. A. Latip, "Person Detection for Social Distancing and Safety Violation Alert based on Segmented ROI," Proc. - 10th IEEE Int. Conf. Control Syst. Comput. Eng. ICCSCE 2020, no. August, pp. 113–118, 2020.

Arduino LED Indicator Available at https://www.tinkercad.com/things/bL82jA7lif8-sizzling-amur-bruticus [Accessed 15 Juni 2022].
Arduino. (2015). "*ARDUINO UNO REV3*".

B. Abdulsalam Abdulrahman and A. Ali Mohammed, "Detection and classification of falling in elderly people using customized deep learning algorithm," J. Zankoy Sulaimani - Part A, vol. 23, no. 1, pp. 119–130, 2021.

Carney, M., Webster, B., Alvarado, I., Phillips, K., Howell, N., Griffith, J., ... & Chen, A. (2020, April). Teachable machine: Approachable Web-based tool for exploring machine learning classification. In Extended abstracts of the 2020 CHI conference on human factors in computing systems (pp. 1-8).

Cholissodin, Imam dkk. (2019). *Buku Ajar AI, Machine Learning & Deep Learning. Malang* : Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya (UB).

D. J. Borkovich, M. Ga, R. J. Skovira, and F. Kohun, "*Virtual Social Distancing: a Digital Ethnography of Online Learning,"* Issues Inf. Syst*.*, vol. 22, no. 4, pp. 244–257, 2021.

Dwivedi, U. (2021, June). *Introducing children to machine learning through machine teaching. In Interaction Design and Children* (pp. 641-643).

Editor P5js Available at :
https://editor.p5js.org/krantas/sketches/IKUf43rB [Accessed 15 Juni 2022].

G. V. Shalini, M. K. Margret, M. J. S. Niraimathi, and S. Subashree, "*Social Distancing Analyzer Using Computer Vision and Deep Learning,"* J. Phys. Conf. Ser., vol. 1916, no. 1, 2021.

I. Amerini, C. T. Li, and R. Caldelli, "*Social Network Identification Through Image Classification with CNN*," IEEE Access, vol. 7, pp. 35264–35273, 2019.

Illustrated: 10 CNN Architectures. [Online] Available at : https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d#e276 [Accessed 20 Juni 2022].

J. Kondragunta and G. Hirtz, "*Gait Parameter Estimation of Elderly People using 3D Human Pose Estimation in Early Detection of Dementia,*" Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS, vol. 2020-July, pp. 5798–5801, 2020.

Jetson Nano Developer Kit Available at
https://developer.nvidia.com/embedded/jetson-nano-developer-kit [Accessed 09 Mei 2022].

Jetson-inference example Available at
https://github.com/dusty-nv/ jetson-inference [Accessed 22 Juni 2022].

K. Bhambani, T. Jain, and K. A. Sultanpure, "*Real-Time Face Mask and Social Distancing Violation Detection System using YOLO,*" Proc. B-HTC 2020 - 1st IEEE Bangalore Humanit. Technol. Conf., 2020.

K. Zheng, F. Wu, and X. Chen, "*Laser-based people detection and obstacle avoidance for a hospital transport robot*," Sensors (Switzerland), vol. 21, no. 3, pp. 1–24, 2021.

Kurniawan, Agus. (2021). IoT Projects with NVIDIA Jetson Nano : Fakultas Ilmu Komputer (FILKOM), Universitas Indonesia (UI). Apress.

M. Aghaei, M. Bustreo, Y. Wang, G. Bailo, P. Morerio, and A. Del Bue, "*Single Image Human Proxemics Estimation for Visual Social Distancing*," pp. 2784–2794, 2021.

M. Aledhari, R. Razzak, R. M. Parizi, and A. Dehghantanha, "*A Deep Recurrent Neural Network to Support Guidelines and Decision Making of Social Distancing*," Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020, pp. 4233–4240, 2020.

M. K. Mudaraddi, "*Social Distancing Using AI and Deep Learning*," Int. J. Res. Appl. Sci. Eng. Technol., vol. 9, no. VII, pp. 1816–1822, 2021.

Membuat Model ML menggunakan Teachable Machine. [Online] Available at : https://yunusmuhammad007. medium.com/membuat-model-ml-menggunakan-teachable-machine-25671f8e4617 [Accessed 01 Juni 2022].

Mengenal Arsitektur LeNet. [Online] Available at : https://softscients.com/2022/02/24/mengenal-arsitektur-lenet/ [Accessed 15 Juni 2022].

Mengenal Deep Learning Lebih Jelas. [Online] Available at : https://www.dicoding.com/blog/mengenal-deep-learning/ [Accessed 09 Juni 2022].

P. Gharti, "*A study of fall detection monitoring system for elderly people through IOT and mobile based application devices in indoor environment*," CITISIA 2020 - IEEE Conf. Innov. Technol. Intell. Syst. Ind. Appl. Proc., 2020.

P. Somaldo, F. A. Ferdiansyah, G. Jati, and W. Jatmiko, "*Developing Smart COVID-19 Social Distancing Surveillance Drone using YOLO Implemented in Robot Operating System simulation environment*," IEEE Reg. 10

Humanit. Technol. Conf. R10-HTC, vol. 2020-Decem, 2020.

Putra, Jan Wira Gotama. (2020). Pengenalan Pembelajaran Mesin dan Deep Learning.

R. Jadhav, S. Phule, and P. Univaersity, "*Human Detection and Monitoring Social Distancing for Covid-19 Using OpenCV and CNN*," J. Xidian Univ., vol. 15, no. 1, pp. 335–339, 2021.

S. K. Jarraya, M. H. Alotibi, and M. S. Ali, "*A deep-CNN crowd counting model for enforcing social distancing during COVID19 Pandemic: Application to Saudi Arabia's public places*," Comput. Mater. Contin., vol. 66, no. 2, pp. 1315–1328, 2020.

S. Lakshmi, P. Kavipriya, M. R. Ebenezar Jebarani, and T. Vino, "*A Novel Approach of Human Hunger Detection especially for physically challenged people*," Proc. - Int. Conf. Artif. Intell. Smart Syst. ICAIS 2021, pp. 921–927, 2021.

S. Majhi, S. K. Nayak, S. S. Barik, K. Singh, and S. Biswal, "*Real-time Object Detection and Recognition Using Deep Learning with YOLO Algorithm for Visually Impaired People*," J. Xidian Univ., vol. 14, no. 4, pp. 2354–2362, 2020.

S. Saponara, A. Elhanashi, and A. Gagliardi, "*Implementing a real-time, AI-based, people detection and social distancing measuring system for Covid-19*," J. Real-Time Image Process., no. 0123456789, 2021.

S. Yadav, "*Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence*," Int. J. Res. Appl. Sci. Eng. Technol., vol. 8, no. 7, pp. 1368–1375, 2020.

Santoso, H. (2015). *Panduan praktis Arduino untuk pemula (Vol. 1). Elangsakti. com.*

T. Pardhu, D. V. S. C. Babu, and E. Amareshwar, "*Design of Obstacle Detection System for Visually Challenged People,*" Int. J. Recent Technol. Eng., vol. 8, no. 5, pp. 5–8, 2020.

The Last 5 Years In Deep Learning. [Online] Available at : https://adeshpande3.github.io/ [Accessed 03 Juni 2022].

Use Teachable Machine AI to Control Anything Available at https://create.arduino.cc/projecthub/alankrantas/use-teach able-machine-ai-to-control-anything-2ad1ee Jetson-inference example Available at https://github.com/dusty-nv/jetson-inference [Accessed 11 Juni 2022].

V. Zope, N. Joshi, S. Iyengar, and K. Mahadevan, "Pr ep rin pe er r Pr ep rin t n ot pe," no. Icicnis, pp. 140–148, 2020.

W. Choi and E. Shim, "*Optimal strategies for social distancing and testing to control COVID-19*," J. Theor. Biol., vol. 512, p. 110568, 2021.

W. Kurniawan, S. Ibrahim, and M. Sulistyo, "*People detection and tracking methods for intelligent surveillance system*," AIP Conf. Proc., vol. 2217, no. April, 2020.

W. Wang, Y. Wang, M. Zhou, and W. Nie, "*A Novel Vital Sign Sensing Algorithm for Multiple People Detection Based on FMCW Radar*," Asia-Pacific Microw. Conf. Proceedings, APMC, vol. 2020-Decem, pp. 1104–1106, 2020.

Y. C. Hou, M. Z. Baharuddin, S. Yussof, and S. Dzulkifly, "*Social Distancing Detection with Deep Learning Model*," 2020 8th Int. Conf. Inf. Technol. Multimedia, ICIMU 2020, no. April, pp. 334–338, 2020.

Y. Guo, W. Qin, Z. Wang, and F. Yang, "*Factors influencing social distancing to prevent the community spread of COVID-19 among Chinese adults*," Prev. Med. (Baltim)., vol. 143, no. December 2020, p. 106385, 2021.

Y. Sahraoui, C. A. Kerrache, A. Korichi, B. Nour, A. Adnane, and R. Hussain, "*Deep D ist : A D eep -L earning -B ased I o V F ramework for R eal -T ime O bjects and D istance V iolation D etection*," no. September, 2020.

Y. Verbelen, S. Kaluvan, U. Haller, M. Boardman, and T. B. Scott, "*Design and implementation of a social distancing and contact tracing wearable*," Colloq. Inf. Sci. Technol. Cist, vol. 2020-June, pp. 466–471, 2020.

Y. Zhou, "*Deep learning based people detection, tracking and re-identification in intelligent video surveillance system,*" Proc. - 2020 Int. Conf. Comput. Data Sci. CDS 2020, pp. 443–447, 2020.

TENTANG PENULIS

Emil Naf'an, S.Kom., M.Kom. Lulus S1 di Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Putra Indonesia YPTK Padang tahun 2000, lulus S2 di Program Magister Teknik Informatika Universitas Putra Indonesia YPTK Padang tahun 2013, sedang studi S3 di Universiti Kebangsaan Malaysia. Saat ini adalah dosen tetap Program Studi Sistem Komputer Fakultas Ilmu Komputer Universitas Putra Indonesia YPTK Padang.

Fajrul Islami, S.Kom., M.Kom. Lulus S1 di Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Putra Indonesia YPTK Padang tahun 2011, lulus S2 di Pogram Magister Teknik Informatika Univeritas Putra Indonesia YPTK Padangtahun 2013, sedang studi S3 di Program Studi Doktor Teknologi Informasi Universitas Putra Indonesia YPTK Padang. Saat ini adalah dosen tetap Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Putra Indonesia YPTK Padang.

Gushelmi, S.Kom., M.Kom. Lulus S1 di Program Studi Sistem Informasi tahun 2002, lulus S2 di Program Magister Teknik Informatika Univeristas Putra Indonesia YPTK Padang 2007, sedang studi S3 di Universiti Kebangsaan Malaysia . Saat ini adalah dosen Tetap Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Putra Indonesia YPTK Padang.

SINOPSIS

Perkembangan Ilmu pengetahuan dan teknologi dalam beberapa tahun terakhir sangat cepat, khususnya ilmu pada bidang AI (*Artificial Inteligence*). Salah satunya adalah metode yang ada pada AI yaitu *Deep Learning*. Deep learning merupakan subbidang *machine learning* yang algoritmanya terinspirasi dari struktur otak manusia. Struktur tersebut dinamakan *Artificial Neural Networks* atau disingkat ANN. Pada dasarnya, ia merupakan jaringan saraf yang memiliki tiga atau lebih lapisan ANN.

Dalam buku ini, akan dijelaskan dasar-dasar tentang metode Deep Learning serta contoh penerapan aplikasi Deep Learning dengan beberapa pemrograman. Buku diawali dengan pembahasan Konsep AI secara umum, metode *Deep Learning*, Algoritma *Deep Learning*, Arsitektur CNN, Pengenalan *Jetson Nano*, dan contoh aplikasi penerapan Deep Learning. Buku ini juga membahas penerapan Deep Learning menggunakan salah satu perangkat yaitu NVIDIA *Jetson Nano*, yang cocok digunakan untuk para pengembang aplikasi AI. Dengan NVIDIA *Jetson Nano* ini, tidak hanya perusahaan besar, pengembang perorangan, peneliti, bahkan pelajar di sekolah maupun perguruan tinggi bisa mengembangkan perangkat pintar dengan lebih mudah.