KLASIFIKASI BUAH SEGAR DAN BUSUK MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* DENGAN *TFLITE* SEBAGAI MEDIA PENERAPAN MODEL *MACHINE LEARNING*

Dino Rahman Sya'bani[1], Amir Hamzah[2], Erma Susanti[3]*
Institut Sains & Teknologi AKPRIND Yogyakarta
e-mail :[1]dinoorahman75@gmail.com, [2]amir@akprind.ac.id, [3]erma@akprind.ac.id
**Corresponding Author* : *erma@akprind.ac.id*

ABSTRACT

CNN (Convolutional Neural Network) is one of the type of Deep Learning algorithm that can accept input in the form of images, or objects such as fruit, animals, office equipment, and so on. This CNN works is inspired by the neuron system in humans based on visual perception. The CNN model can be implemented into the Android system using TFLite. TFLite is a Machine Learning library specially designed for Android devices to create Machine Learning models in small binary and low latency. This can make it easier for Machine Learning researchers to use smartphones to develop the research system. Current technological advances make the fruit industry no longer use the manual fruit sorting process carried out by humans. The process of sorting fruit that is fit for consumption (fresh) and unfit for consumption (rotten) can reach hundreds and thousands, so it needs the help of machines to increase work speed. This fruit sorting process can be circumvented by creating a Machine Learning model that can classify fresh and rotten fruit which will then be processed by the system based on the desired output requirements. If the model can perform the classification well, then the fruit industry can take steps to make an automatic sorter by implementing the model into its system. This study aims to create a model with the CNN (Convolutional Neural Network) Algorithm and design a prototype system that can classify fresh and rotten fruit (red apples, bananas, and mandarin oranges) and apply the resulting model using TFLite on an Android Smartphone. The results indicated that the resulting prototype model can be used by users in the Android system. The results of the training model obtained an accuracy value of 98.20%. Meanwhile, the results of testing showed that the model can classify images from the same or different sources by 91.65% from 1090 images. These results indicate that the model is included in the excellent classification category.

Keywords : *Android, Convolutional Neural Network, Classification, Machine Learning, TFLite*

INTISARI

CNN (Convolutional Neural Network) merupakan salah satu jenis algoritma Deep Learning yang dapat menerima input berupa gambar, ataupun objek seperti buah, hewan, alat kantor, dan sebagainya. CNN ini bekerja terinspirasi dari sistem neuron pada manusia berdasarkan persepsi visual. Model CNN dapat diterapkan ke dalam sistem Android menggunakan TFLite. TFLite merupakan library Machine Learning yang dirancang untuk perangkat Android untuk mengembangkan model Machine Learning dalam binari yang kecil dan latensi yang rendah. Hal ini tentunya dapat memudahkan para peneliti Machine Learning untuk menggunakan ponsel pintar untuk mengembangkan sistem yang ditelitinya. Kemajuan teknologi saat ini membuat industri buah tidak lagi menggunakan proses sortir buah secara manual yang dikerjakan oleh manusia. Proses sortir buah yang layak dikonsumsi (segar) dan tidak layak dikonsumsi (busuk) dapat berjumlah ratusan, bahkan ribuan, sehingga sangat membutuhkan bantuan mesin untuk dapat meningkatkan kecepatan kerja. Proses sortir buah ini dapat disiasati dengan membuat model Machine Learning yang mampu mengklasifikasikan buah segar dan busuk yang kemudian akan diproses oleh sistem berdasarkan kebutuhan keluaran yang dikehendaki. Jika model dapat melakukan klasifikasi dengan baik, maka selanjutnya industri buah dapat mengambil langkah untuk melakukan pembuatan alat sortir otomatis dengan menerapkan model tersebut ke dalam sistemnya. Penelitian ini bertujuan untuk membuat model dengan Algoritma CNN (Convolutional Neural Network), dan merancang sistem prototipe yang dapat mengklasifikasi buah segar dan busuk (apel merah, pisang, dan jeruk mandarin) serta menerapkan model yang dihasilkan menggunakan TFLite pada Smartphone Android. Hasil *training* model didapatkan nilai akurasi sebesar 98,20%. Sedangkan hasil *testing* didapatkan bahwa model dapat mengklasifikasi gambar yang bersumber sama maupun berbeda sebesar 91,65% dari 1090 citra. Hasil ini menunjukkan bahwa model masuk dalam kategori *excellent classification*.

Kata kunci : *Android, Convolutional Neural Network, Klasifikasi, Machine Learning, TFLite*

1. PENDAHULUAN

Adanya sebuah sistem yang menggunakan teknologi kamera telah banyak digunakan sebagai pengolahan citra pada berbagai macam penelitian, salah satunya pada klasifikasi gambar. Pada umumnya, klasifikasi gambar

menggunakan *(CNN) Convolutional Neural Network* sebagai algoritmanya. *CNN* ini merupakan salah satu jenis algoritma *Deep Learning* yang dapat menerima masukan berupa citra, ataupun objek apapun seperti buah, hewan, alat kantor, dan sebagainya. *CNN* ini bekerja terinspirasi dari sistem *neuron* pada manusia berdasarkan persepsi visual. Sebelum diterapkan ke dalam sistem, rancangan arsitektur model *CNN* akan dilatih terlebih dahulu, hingga mencapai akurasi tertentu.

Model *CNN* dapat diterapkan ke dalam sistem *Android* menggunakan *TFLite*. *TFLite* merupakan Pustaka *Machine Learning* yang memang dirancang untuk perangkat *Android* mengembangkan mesin belajar dalam binari yang kecil dan latensi yang rendah. Hal ini tentunya dapat memudahkan para peneliti *Machine Learning* untuk menggunakan *Smartphone Android* yang dimiliki untuk mengembangkan sistem yang ditelitinya dan menggunakannya sebagai media penerapan/prediksi gambar, sehingga tidak memerlukan biaya yang tinggi untuk melakukan percobaan penerapan model *Machine Learning* yang telah dibangun. Pada *TFLite*, untuk mendeteksi objek dan mengklasifikasikan gambar hanya perlu menggunakan kamera pada perangkat *Android* yang dapat dilakukan dengan cara *Uploading File* (unggah file), *Taking Picture From Camera* (ambil gambar dari kamera), dan *Real-Time Camera Detection* (deteksi kamera secara langsung).

Penggunaan arsitektur model *Convolutional Neural Network* (*CNN*) untuk klasifikasi buah-buahan juga pernah dilakukan oleh (Maulana & Rochmawati, 2020). Pada penelitian tersebut model *CNN* dirancang dengan komponen arsitektur model diantaranya yaitu 3 *convolution layer* dan 2 *hidden* l*ayer sigmoid* dengan jumlah *kernel* 700 dan 500, serta *activation rectified linear unit (relu)*. Model ini dapat mengklasifikasikan citra buah dengan akurasi yang tinggi yaitu 97,97%. Model CNN yang dibuat dapat mengklasifikasikan gambar buah yang diambil dengan kamera *smartphone*. Model ini akan mengklasifikasi 15 kelas citra buah yang tidak dikenali ke dalam kelas buah yang dianggap paling mirip di antara kelas buah yang di-*training*. Jumlah data citra yang digunakan untuk proses pelatihan adalah 3.375 citra, terdiri dari 225 citra dari 15 kelas buah. Proses pengujian / *testing* pada penelitian ini menggunakan 345 data citra uji yang berisi 23 citra / gambar dari masing-masing 15 kelas citra buah. Adapun 15 kelas citra buah terdiri dari apel merah, alpukat, pisang, kurma, anggur merah, anggur putih, kiwi, lemon, leci, mangga, jeruk, pepaya, pir, stroberi, dan tomat. Proses klasifikasi berlangsung di *Google Collaboratory* (Maulana & Rochmawati, 2020)

Penelitian klasifikasi citra buah pernah dilakukan oleh (Sabilla, 2020) dengan merancang arsitektur model *Convolutional Neural Network* agar dapat mengklasifikasikan jenis buah yang segar agar dapat ditimbang menggunakan neraca buah kemudian dijual ke konsumen. Penelitian yang dilakukan oleh (Sabilla, 2020) ini menggunakan 2 *layer* 2 *maxpooling* dan *fully connected* dengan modifikasi *K-Nearest Neighbor* untuk hasil *training* akurasi yang didapat yaitu 0,990783 dengan waktu 0,0574331 detik. Kemudian untuk hasil akurasi *testing* 0,882979 dengan waktu komputasi 0,0266471 detik dengan *datasets* yang terkumpul sebanyak 312 data yang mana dibagi ke dalam 16 kelas. Kelas tomat dibagi menjadi 2 yakni tomat *cherry* dan tomat buah. Kelas cabai dibagi menjadi 4 kelompok yakni cabai rawit, cabai hijau, cabai merah dan cabai keriting. Tiap kelas dibagi lagi menyesuaikan kondisinya yaitu baik dan buruk. Sehingga menghasilkan kategori cabai rawit segar, cabai rawit tidak segar, cabai hijau segar, cabai hijau tidak segar, cabai merah segar, cabai merah tidak segar, cabai keriting segar, cabai keriting tidak segar, tomat cherry segar, tomat cherry tidak segar, tomat hijau segar, tomat hijau tidak segar, tomat buah segar, tomat buah tidak segar, tomat busuk dan cabai busuk. Untuk rancangan alatnya dibangun menggunakan *motherboard* raspberry-pi b+ dan kamera (*webcam*) yang digunakan untuk menangkap citra melakukan proses klasifikasi (Sabilla, 2020).

Kemudian penelitian yang dilakukan oleh (Kurniadi, Prasetyo, Ahmad, Aditya Wibisono, & Sandya Prasvita, 2021) yakni membandingkan kualitas maupun akurasi tertinggi antara algoritma *Support Vector Machine* (*SVM*) dan *Convolutional Neural Network* (*CNN*) sebagai model yang nantinya akan digunakan untuk mengklasifikasi buah – buahan. Rancangan arsitektur model *CNN* yang digunakan diantaranya yaitu *input* 100 x 100 *pixel*, 3 *convolution layer*, 2 *max pooling*, *activation rectified linear unit (relu)* dan 1 *fully connected layer* dengan *activation softmax* karena memiliki lebih dari 2 *class*. Sedangkan untuk Metode *SVM* tidak sama dengan *CNN* yang harus merancang arsitektur modelnya. Pada metode *SVM* hanya perlu memaksimalkan jarak antar *class* hingga mendapatkan *support vector* kemudian mendapatkan *hyperplane* yang paling optimal. *Datasets* yang digunakan berasal dari *Kaggle Datasets* yakni berjumlah 11.219 citra yang memiliki 17 *class* diantaranya yaitu apel, alpukat, pisang, *blueberry*, ceri, jagung, jambu, kiwi, lemon, mangga, jeruk, pepaya, nanas, rambutan, salak, strawberry, dan semangka. Tahap *training* dilakukan sebanyak 25 *epochs* dengan data *training* berjumlah 70% atau 8.407 dan data *testing* berjumlah 30% atau 2.812 dari total *datasets* yang dimiliki yaitu 11.219. Akurasi yang diperoleh dengan menggunakan metode *CNN* sejumlah 96,87% sedangkan metode *SVM* 93,09%, alat yang dibangun pada penelitian ini tidak ada, karena hanya melakukan analisis perbandingan arsitektur model manakah yang lebih baik antara *SVM* dan *CNN* dengan menggunakan *Library SKLearn – Confussion Matrix* (Kurniadi et al., 2021).

Pengklasifikasian buah zaitun yang dilakukan oleh (Dwi Antoko, Azhar Ridani, & Eko Minarno, 2021) ini menggunakan algoritma *Convolutional Neural Network* (*CNN*) sebagai arsitektur modelnya. Sebanyak 267 citra buah zaitun dengan 6 *class*/kategori diantaranya yaitu Arbequina, Arbosana, Changlot, Lechin, Picual, dan Verdial dijadikan *datasets* pada penelitian ini. Rancangan Model ini menggunakan *input* 210 x 210 *pixel*, 4 *convolution*

layer dengan *kernel* berukuran 3x3, 2 *max pooling*, *activation rectified linear unit (relu)* dengan *kernel* berukuran *2x2 stride 2, pooling layer* dengan kernel 2x2 stride 1, *image data generator* dengan parameter *rotation range* 360 dan 1 *fully connected layer* dengan *activation softmax.* Model ini menghasilkan 0,97 atau 97% *training accuration* dan 0,92 atau 92% *validation accuration*. Sistem yang dibangun sebagai pengujian model arsitektur pada penelitian ini menggunakan *Open Computer Vision*.

Adapun perbedaan penelitian ini dengan penelitian-penelitian sebelumnya yaitu terletak pada metode penerapan model *Machine Learning* yang digunakan. Metode penerapan model *Machine Learning* pada penelitian ini menggunakan TensorFlowLite/TFLite untuk mengimplementasikan model yang telah dibangun dan menggunakan sistem operasi Android sebagai *user interface/user experienced*. Penelitian ini bertujuan untuk mengetahui cara membangun model dengan Algoritma *Convolutional Neural Network* (*CNN*), dan merancang sistem prototipe yang dapat mengklasifikasi buah segar dan busuk (apel merah, pisang, dan jeruk mandarin) menggunakan algoritma *CNN* yang penerapan modelnya menggunakan *TFLite* pada *Smartphone Android*.

2. METODE PENELITIAN

2.1. Konsep Dasar

Deep Learning* dan *Machine Learning

Metode *Deep Learning* merupakan metode pembelajaran representasi berlapis, dan representasi dapat membentuk arsitektur jaringan saraf yang memiliki banyak *layer*. Lapisan pada *Deep Learning* dibagi menjadi tiga bagian yakni *Input Layer, Hidden Layer* dan *Output Layer*. *Hidden Layer* dapat dibuat banyak untuk menemukan konfigurasi algoritma yang tepat yang meminimalkan kesalahan dalam keluaran / *output*. Semakin banyak *layer* yang dihasilkan maka semakin sedikit pula kesalahan yang dihasilkan, sehingga meningkatkan akurasi yang lebih baik (LeCun, Yoshua, & Hinton, 2015).

Istilah *Machine Learning* (pembelajaran mesin) pertama kali didefinisikan oleh Arthur Samuel pada tahun 1959. Menurut Arthur Samuel, *Machine Learning* adalah cabang ilmu komputer yang memungkinkan komputer belajar memahami sesuatu tanpa program eksplisit. *Machine Learning* dapat didefinisikan sebagai metode komputasi eksperimental untuk meningkatkan kinerja atau membuat prediksi yang akurat. Adapun yang dimaksud dengan pengalaman adalah informasi yang ada yang dapat digunakan sebagai data pelatihan (Pujoseno, 2018).

Machine Learning adalah cabang dari kecerdasan buatan yang bertujuan membuat komputer belajar seperti manusia. Pembelajaran mesin dirancang untuk membantu manusia menganalisis data dan membuat prediksi. *Machine Learning* melakukan pelatihan data untuk mempelajari, menghasilkan pola dan fitur, lalu melakukan klasifikasi dan pengujian.

CNN (Convolutional Neural Network)

Convolution adalah *layer* yang menjadi dasar dari arsitektur *CNN* untuk melakukan operasi *convolution* pada *output* dari *layer* sebelumnya (Asrafil, Paliwang, Septian, Cahyanti, & Swedia, 2020).

Gambar / citra yang di-*input* ke model dan melewati lapisan ini akan dilakukan proses *Feature Extraction* (ekstraksi ciri). Gambar / citra dibagi menjadi beberapa bagian untuk setiap piksel sesuai dengan parameter yang ditentukan. Proses ini dapat membuat gambar menjadi lebih kecil atau berukuran sama, tetapi mengubah kedalaman gambar. *Convolutional layer* terdiri dari *neuron - neuron* yang tersusun membentuk filter dengan panjang dan tinggi dalam piksel. Misalkan terdapat contoh *layer* dengan dimensi 6x6x3, ini berarti *layer* tersebut memiliki panjang 6 piksel, tinggi 6 piksel dan ketebalan atau jumlah 3 *channel* / saluran pada model gambar yaitu *Red Green Blue*. Tiga filter tersebut akan digeser dan dilakukan operasi antara *input* dari nilai filter untuk menghasilkan *Feature Map* (peta fungsi) atau *Activation Map* (peta aktivasi) (Azis, Herwanto, & Ramadhani, 2021).

Secara umum tipe *Convolutional Neural Network Layer* secara umum dibagi menjadi dua tipe yaitu antara lain:

1. *Classification Layer* (Lapisan Klasifikasi)
   Classification Layer tersusun dari beberapa lapisan. Setiap *layer* tersusun dari *neuron* yang terhubung secara penuh atau *Fully Connected* terhadap *layer* lainnya. *Layer* ini menerima masukan dari hasil keluaran *Feature Extraction Layer* citra berupa vektor yang kemudian ditransformasikan layaknya *Multi Neural22 Networks* dengan tambahan berupa *Hidden Layer*. Hasil keluaran merupakan akurasi kelas yang akan diklasifikasi. Beberapa tingkatan hasil akurasi yakni diantaranya:
   a. *Excellent Classification* = 0,90-1,00 (90% - 100%
   b. *Good Classification* = 0,80-0,89 (80% - 89%)
   c. *Fair Classification* = 0,70-0,79 (70% - 79%)
   d. *Poor Classification* = 0,60-0,69 (60% - 69%)
   e. *Failure* = 0,50-0,59 (50% - 59%)
2. *Feature Extraction Layer* (Lapisan Ekstraksi Fitur)
   Layer ini berada di posisi awal arsitektur. *Neuron* yang terdapat di dalam *layer* ini terkoneksi pada daerah lokal dari *layer* sebelumnya. *Convolutional Layer* berada di *layer* pertama, sedangkan *Pooling Layer*

berada di *layer* kedua. *Layer* ini menerima masukan citra secara langsung, kemudian memprosesnya hingga menghasilkan keluaran berupa vektor untuk diolah pada *layer* berikutnya (Kulshrestha, 2019).

Object Detection

Object Detection (Deteksi Objek) merupakan teknik *Computer Vision* untuk menemukan contoh objek dalam suatu video ataupun gambar. Algoritma *Object Detection* dapat memanfaatkan *Machine Learning* ataupun *Deep Learning* untuk mendapatkan hasil yang berguna. Saat manusia melihat video ataupun gambar, maka manusia dapat memahami, mengenali bahkan menemukan objek dalam beberapa saat. Berbeda dengan komputer, dimana memerlukan komputasi kompleks agar dapat menghasilkan keluaran yang diharapkan. *Object Detection* bertujuan untuk mereplikasikan kecerdasan yang dimiliki manusia dalam melihat benda dengan menggunakan komputer. Cara kerja *Object Detection* yakni dengan menempatkan keberadaan objek dalam citra dan menggambar *border box* di sekitar objek tersebut. Proses ini melewati dua proses, yaitu mengklasifikasikan jenis objek, dan menggambar *border box* di sekitar objek tersebut. Skenario deteksi objek dan klasifikasi gambar terlihat serupa. Secara umum, *Object Detection* (Deteksi Objek) adalah mengidentifikasi lokasi objek pada citra / gambar, dan kemudian menghitung jumlah *instance* suatu objek, sedangkan *Classification* (Klasifikasi) adalah mengklasifikasikan citra / gambar pada suatu kategori tertentu (Ouaknine, 2018).

Pooling Layer* dan *Transfer Learning

Pooling Layer adalah pengurangan ukuran matriks dengan menggunakan operasi *Pooling* (Rizki, Medikawati Taufiq, Mukhtar, & Putri, 2021). *Pooling Layer* biasanya digunakan setelah *Convolution Layer*. Proses konvolusi dan *pooling* dilakukan beberapa kali untuk mendapatkan *Feature Maps* dengan ukuran yang dikehendaki.

Transfer Learning adalah suatu teknik atau metode yang memanfaatkan model yang sudah dilatih terhadap suatu *Datasets* untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *Starting Point*, memodifikasi dan memperbaharui parameternya, sehingga sesuai dengan *Datasets* yang baru (Sena, 2018).

Tensorflow* dan *Python

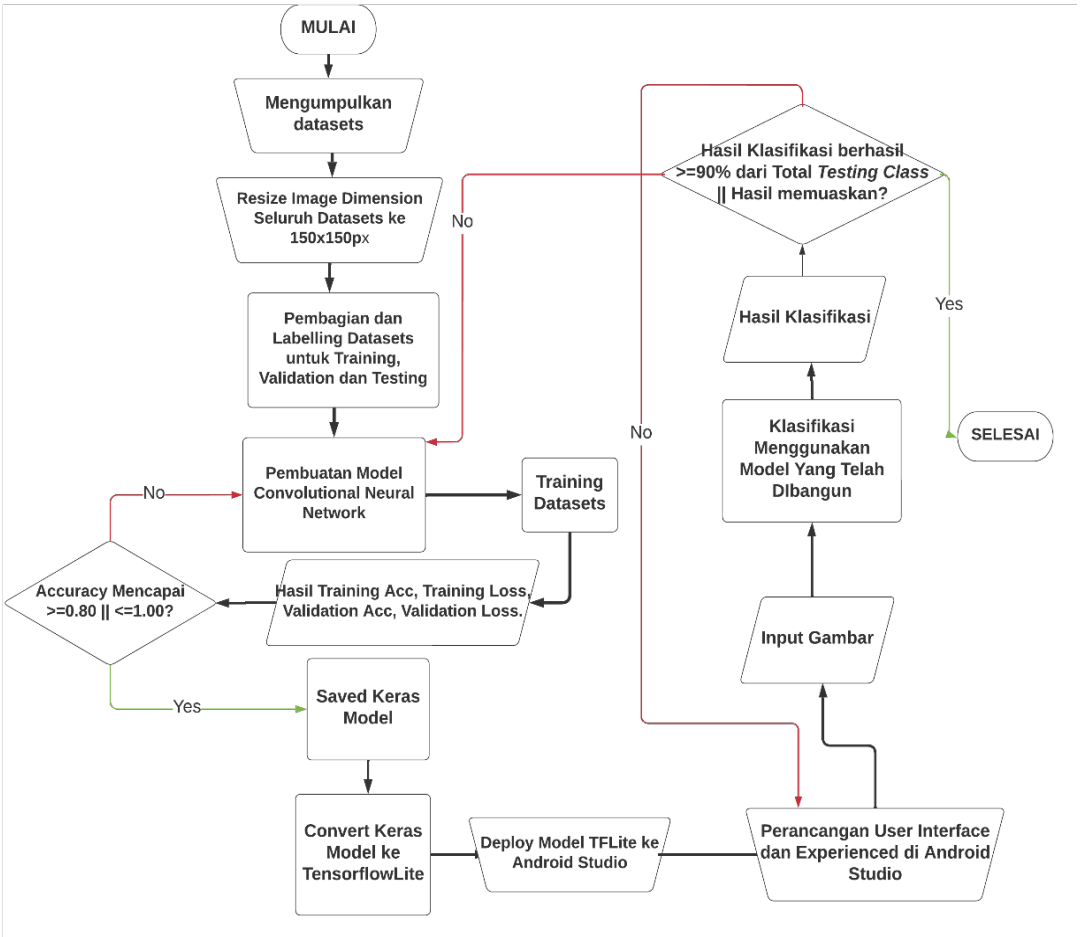
Tensorflow adalah pembelajaran mesin sumber terbuka dan perpustakaan jaringan saraf yang dikembangkan oleh *Google Brain*. *Tensorflow* dapat menjalankan algoritma *machine learning*. Tujuan *Tensorflow* adalah penelitian, pemrograman, dan pengembangan pembelajaran mesin. *Library* ini dapat digunakan dengan berbagai bahasa pemrograman seperti *Python, Java, C/C++* (Rucci & Casile, 2005).

Python dapat digunakan sebagai bahasa pemrograman dinamis dengan manajemen memori otomatis. *Python* dapat digunakan untuk berbagai tujuan pengembangan perangkat lunak dan dapat berjalan di berbagai *platform* sistem operasi. (Syahrudin & Kurniawan, 2018).

2.2. Diagram Alir Penelitian

Diagram Alir Penelitian terdapat pada Gambar 1. Penelitian dimulai dengan mengumpulkan *Datasets* melalui *Kaggle Datasets* sesuai dengan keperluan sistem, kemudian dilanjutkan dengan pembagian *datasets*. Pembagian *dataset* menurut (Géron, 2017) bergantung pada ukuran *dataset*, jika berisi 10 juta *instance*, maka menahan 1% berarti set pengujian akan berisi 100.000 *instance*. Jadi umum untuk menggunakan 70% data untuk *training* (pelatihan) dan 30% untuk *testing* (pengujian), karena sudah lebih dari cukup untuk mendapatkan perkiraan yang baik tentang kesalahan generalisasi. Oleh karena itu pada penelitian ini pembagian dataset untuk data *training* berjumlah 70%, data *validation* berjumlah 20%, dan data *testing* berjumlah 10% (5% dari sumber *datasets* yang sama dan 5% dari sumber yang berbeda). Pembagian tersebut dilakukan untuk menguji coba model apakah benar - benar dapat mengklasifikasikan citra / gambar yang berbeda dengan *datasets training* dan *validation* atau apakah hanya dapat mengklasifikasikan citra / gambar yang sama dengan *datasets training* dan *validation*.

Gambar di-*resize* dimensinya menjadi 150*px* (panjang dan lebarnya). Proses *resize* selesai, maka dilanjutkan dengan pembagian dan *labelling datasets* untuk *training*, *validation* dan *testing*. Perancangan arsitektur model dilakukan sebelum melakukan *training* model menggunakan *datasets training* dan *validation* yang sebelumnya telah disusun. *Training* model dilakukan hingga mendapatkan hasil akurasi lebih dari sama dengan 80 atau kurang dari sama dengan 1,00. Jika belum mendapatkan hasil yang diharapkan, maka dilakukan perancangan arsitektur model secara terus menerus dengan menambah / mengurangi *layer* konvolusi maupun jumlah *filter*. Apabila telah mendapatkan hasil yang diharapkan, maka lakukan *save* pada model yang telah di-*training*, lalu konversi ke ekstensi *TFLite*. Melakukan *deploy* model *TFLite* ke dalam *Android Studio*, kemudian merancang *interface* untuk menggunakan model sebagai media penerapan klasifikasinya. Saat melakukan klasifikasi sebaiknya menggunakan *datasets* yang bersumber sama dan *datasets* yang bersumber berbeda dengan *training* maupun *validation datasets*, agar dapat terlihat kemampuan model dalam mengklasifikasi apakah sudah masuk ke dalam kategori *good classification* (0,80 – 0,89) atau *excellent classification* (0,90 – 1,00). Jika belum memenuhi kategori tersebut, maka sebaiknya dilakukan perancangan model kembali. Cukup menggunakan kamera dan penyimpanan pada *Smartphone Android* maka proses klasifikasi buah segar dan busuk dapat dijalankan.



Gambar 1. Diagram Alir Penelitian

2.3. Mengetahui Akurasi Model

Cara untuk mengetahui hasil akurasi dapat dilakukan dengan dua cara, yaitu langsung mencetaknya melalui variabel pada model yang bernama *Accuracy*, *Loss*, *Val_Accuracy*, dan *Val_Loss* setelah proses *training* berakhir, dan dengan cara menghitung manual melalui *Confussion Matrix*. Rumus perhitungan akurasi model dengan *Confussion Matrix* diuraikan pada rumus perhitungan (1).

$$Akurasi = \frac{Total\ prediksi\ benar}{Total\ prediksi} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (1)$$

Adapun beberapa parameter yang akan digunakan untuk *Confussion Matrix* sebagai berikut:

a. *True Positive* (TP): Hasil dimana model benar memprediksi kelas positif (contoh: *input* buah apel berada di kelas positif, dan *output*-nya buah apel yang berada di kelas positif).
b. *True Negative* (TN): Hasil dimana model benar memprediksi kelas negatif (contoh: *input* buah jeruk berada di kelas negatif, dan *output*-nya benar buah jeruk berada di kelas negatif).
c. *False Positive* (FP): Hasil dimana model salah memprediksi kelas positif padahal sebenarnya kelas negatif (contoh: *input* buah jeruk yang berada di kelas negatif, namun *output*-nya buah apel yang berada di kelas positif).
d. *False Negative* (FN): Hasil dimana model salah memprediksi kelas negatif padahal sebenarnya kelas positif (contoh: *input* buah apel yang berada di kelas positif, namun *output*-nya buah jeruk yang berada di kelas negatif).

3. HASIL DAN PEMBAHASAN

3.1. Pengumpulan *Dataset* dan *Preprocessing*

Bahan dan objek penelitian yaitu citra buah apel, jeruk mandarin dan pisang yang segar maupun busuk, yang dikumpulkan melalui *Kaggle Datasets* sebanyak 10.900 citra yang mana 8.720 untuk *training class* dan 2.180 *validation class*. Kemudian sebanyak 1.090 citra untuk *testing class* (545 dari sumber sama dengan *datasets* dan 545 dari sumber berbeda dengan *datasets*). Rincian pembagian *datasets* terdapat pada Tabel 1. Setelah

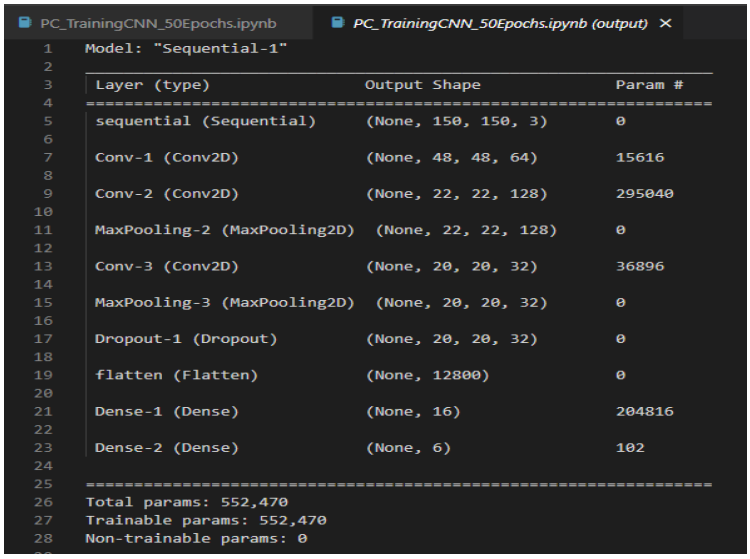
pengumpulan *dataset* dilakukan *preprocessing* dengan melakukan *resize* (pengubahan ukuran *dataset*), pembagian *dataset* untuk *training*, *validasi*, dan *testing* serta pelabelan pada *dataset*.

Tabel 1. Rincian pembagian *Datasets Training*, *Validation* dan *Testing*

No	*Class*	Data *Training*	Data *Validation*	Total Data *Class Training*	Data *Testing* (Sumber *Datasets* Sama Dengan *Training*)	Data *Testing* (Sumber *Datasets* Berbeda Dengan *Training*)	Total Data *Class Testing*
1	Apel Busuk	1.873	468	2.341	90	90	180
2	Apel Segar	1.355	338	1.693	91	91	182
3	Jeruk Mandarin Busuk	1.780	444	2.224	91	91	182
4	Jeruk Mandarin Segar	1.262	317	1.579	91	91	182
5	Pisang Busuk	1.276	320	1.596	91	91	182
6	Pisang Segar	1.174	293	1.467	91	91	182

3.2. Pembuatan Model *Convolutional Neural Network* (*CNN*)

Pembuatan Model *CNN* pada penelitian ini dilakukan secara *trial and error* dalam artian selalu dilakukan perubahan / perbaikan komponen arsitektur model seperti jumlah *convolution layer*, *filter*, *kernel*, *stride* maupun *pooling layer* hingga mendapatkan hasil akurasi *training* yang memuaskan. Perubahan / perbaikan arsitektur ini dilakukan setelah model *CNN* digunakan untuk *training datasets* terlebih dahulu, sehingga memperlihatkan hasil akurasi maupun *loss training* dan *validation/testing*. Apabila hasil dikira memuaskan, maka model tersebut telah siap digunakan/di-*deploy* ke dalam program. Sehingga didapatkan model *summary* dari rancangan arsitektur model *Convolutional Neural Network* (*CNN*).Pembuatan model penelitian ini menciptakan total *parameters* sebanyak 552.470 dengan jumlah *trainable parameters* (parameter yang dapat dilatih) sebanyak 552.470. Besarnya nilai total *parameters* dan *trainable parameters* ditentukan oleh rancangan arsitektur model *Convolutional Neural Network* yaitu dari jumlah *filters* dan *kernel*. Rancangan hasil model ini merupakan hasil model yang telah dikompilasi dan merupakan hasil perbaikan dari model awal. Proses perbaikan arsitektur model dimaksudkan untuk mendapatkan model dengan akurasi terbaik dari semua percobaan. Pada *convolution layers* pertama memiliki nilai filter 64 dan kernel 9x9, *convolution layers* kedua memiliki nilai filter 128 dan kernel 6x6, *convolution layers* ketiga memiliki nilai filter 32 dan *kernel* 3x3, *dense* pertama memiliki nilai filter 16 dan *activation relu*, & *dense* kedua memiliki nilai filter 6 sesuai dengan jumlah *class* yang akan diklasifikasi dan *activation softmax*. Hasil ringkasan rancangan arsitektur model yang digunakan untuk melakukan *training* data yakni tertera pada Gambar 2. Arsitektur menggunakan model *sequential*. Model terdiri dari 3 *convolution layer*, 2 *maxpooling layer,* 1 *dropout*, 1 *flatten*, 2 *dense,* 1 *activation relu* dan 1 *activation softmax* pada *layer* terakhir.



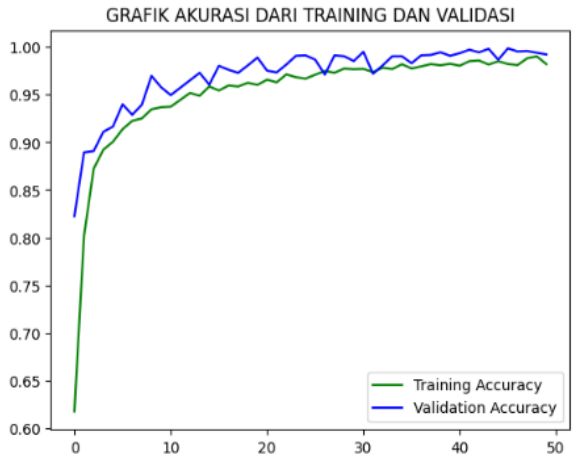
Gambar 2. Hasil Rancangan Arsitektur Model *CNN*

3.3. Training Model

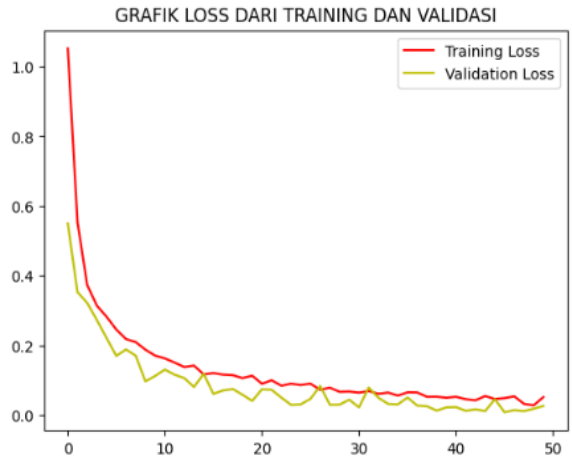
Hal yang menjadi faktor proses *training* memakan waktu yang lama yaitu karena banyaknya jumlah *convolution layers*, *filter*, *kernel*, dan *trainable parameters* pada model *CNN*, jumlah *datasets* yang ribuan bahkan

hingga puluhan ribuan, jumlah *batch* yang begitu kecil, jumlah *epochs* yang begitu banyak, namun semakin lama model melakukan *training*, maka semakin tinggi tingkat *accuracy* (akurasi) dan menurunnya tingkat *loss* pada model, sehingga akan menghasilkan model yang baik untuk melakukan klasifikasi.

Hasil proses *training* dan *validasi* digambarkan dengan plot grafik pada Gambar 3 dan gambar 4. Berdasarkan grafik pada Gambar 3 dan Gambar 4 didapatkan hasil bahwa model dapat dikatakan dalam kategori *Best Fit*. Karena selama proses *training*, grafik nilai *accuracy* kian meningkat dan nilai *loss* kian menurun sehingga memiliki *high accuracy* (0,9820) dan *low loss* (0,0521) yang mana masuk ke salah satu ciri – ciri model yang *Best Fit*, yang kemudian model dapat memahami kelompok data tanpa dipengaruhi oleh *noise* data. Berbeda dengan model *underfitting* yang mana memiliki *high loss* dan *low accuracy*, dan model *overfitting* yang memiliki *low loss* dan *low accuracy* pula.



Gambar 3. Grafik Akurasi dari *Training* dan Validasi



Gambar 4. Grafik *Loss* dari *Training* dan Validasi

Penggunaan 3 *convolution layer* pada penelitian ini menghasilkan total *parameters* dan *trainable parameters* sebanyak 552.470 yang memakan waktu *training* data selama 48,67 menit sebanyak 50 *epochs* dan 545 *steps per epochs* menghasilkan nilai *training accuracy* = 0,9820 (98,20%), *validation accuracy* = 0,9922 (99,22%), *training loss* = 0,0521 (05,21%), dan *validation loss* = 0,0265 (02,65%). Lamanya waktu *training* data dipengaruhi oleh prosesor yang digunakan pada perangkat penelitian dan juga arsitektur *layer* pada model. Sebanyak 91 dari 1.090 data *testing* bernilai *false positive* dan *false negative* atau gagal mengklasifikasikan data ke dalam *class*-nya masing-masing.

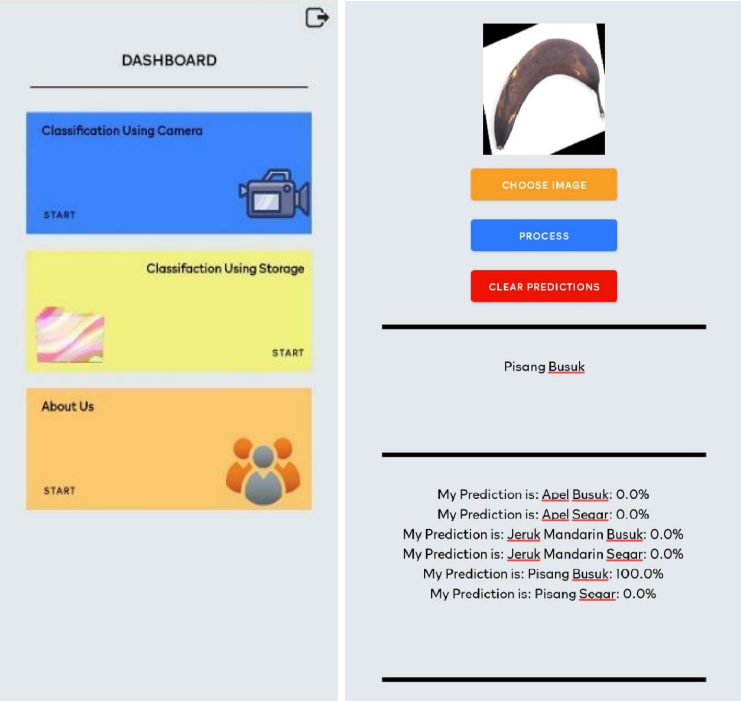
3.4. *Convert Keras Model* ke *TFLite* dan *Deploy Model*

Setelah selesai melakukan *training* model, maka model harus di-*convert* terlebih dahulu agar dapat digunakan pada *Android TFLite*, sehingga tahap uji coba model dapat dilakukan setelahnya. Model yang telah di-*Convert* ke *TFLite* kemudian dimasukkan ke dalam folder *Project Android Studio*, dengan cara klik kanan *mouse* pada folder *Project* > *New* > *Other* > *Tensor Flow Lite Model*. Selanjutnya tunggu hingga proses G*radle Build Indexing* selesai.

3.5. Hasil *Testing* (Klasifikasi dengan Data Baru)

Model diuji coba menggunakan aplikasi yang telah ter-*deploy* model berekstensi **TFLite*. Pengujian ini menggunakan menu *classification using storage* (klasifikasi menggunakan penyimpanan) dengan 50% *datasets testing* bersumber sama dan 50% *datasets testing* bersumber berbeda. Menu dan tampilan hasil klasifikasi melalui menu *Classification Using Storage* ditunjukkan pada Gambar 5. Setelah melakukan klasifikasi dengan menggunakan aplikasi yang telah ter-*deploy* model, maka didapatkan hasil *confusion matrix* tertera pada Tabel 2 dan Tabel 3.

P-ISSN: 1979-911X
E-ISSN: 2541-528X



Gambar 5. Menu dan tampilan hasil klasifikasi melalui menu *Classification Using Storage*

Tabel 2. *Confusion Matrix* Hasil Uji Coba Model dengan 50% *Datasets Testing* Bersumber Sama

		AKTUAL					
		Apel Busuk	**Apel Segar**	**Jeruk Mandarin Busuk**	**Jeruk Mandarin Segar**	**Pisang Busuk**	**Pisang Segar**
PREDIKSI	**Apel Busuk**	89	0	0	0	0	0
	Apel Segar	1	91	0	0	0	0
	Jeruk Mandarin Busuk	0	0	91	0	0	0
	Jeruk Mandarin Segar	0	0	0	91	0	0
	Pisang Busuk	0	0	0	0	91	0
	Pisang Segar	0	0	0	0	0	91

Tabel 3. *Confusion Matrix* Hasil Uji Coba Model Dengan 50% *Datasets Testing* Bersumber Berbeda

		AKTUAL					
		Apel Busuk	**Apel Segar**	**Jeruk Mandarin Busuk**	**Jeruk Mandarin Segar**	**Pisang Busuk**	**Pisang Segar**
PREDIKSI	**Apel Busuk**	67	11	0	0	0	1
	Apel Segar	4	73	0	1	0	0
	Jeruk Mandarin Busuk	7	2	81	27	0	2
	Jeruk Mandarin Segar	0	3	0	64	0	1
	Pisang Busuk	10	1	4	0	91	8
	Pisang Segar	2	1	6	0	0	79

Tabel 4. *Confussion Matrix* Hasil *Testing* Data Secara Menyeluruh

		AKTUAL					
		Apel Busuk	**Apel Segar**	**Jeruk Mandarin Busuk**	**Jeruk Mandarin Segar**	**Pisang Busuk**	**Pisang Segar**
PREDIKSI	**Apel Busuk**	166	14	16	4	4	2
	Apel Segar	3	161	7	16	7	1
	Jeruk Mandarin Busuk	18	7	172	9	0	2
	Jeruk Mandarin Segar	0	16	4	179	6	5
	Pisang Busuk	6	1	1	1	166	20
	Pisang Segar	7	1	0	1	17	170

Rincian hasil *testing* menggunakan *datasets* bersumber sama dan berbeda (secara menyeluruh) tertera pada Tabel 4. Model dapat mengklasifikasi gambar yang bersumber sama maupun berbeda dari *datasets training* dengan baik atau sebesar 91,65% dari 1090 citra. Perhitungan akurasi berdasarkan rumus (1) didapatkan hasil:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} = \frac{156 + 843}{156 + 843 + 5 + 86} = \frac{999}{1090}$$

$$= 91{,}65\%\ (0{,}9165)$$

Berdasarkan hasil uji coba klasifikasi menggunakan model berekstensi *TFLite* pada Tabel 2 dan Tabel 3, maka didapatkan hasil akurasi pada Tabel 5.

Tabel 5. Rincian Hasil *Testing* "Akurasi"

Hasil *Testing* "Akurasi"	
Datasets Sumber Sama	= **99.81% (0.9981)** = 544 / 545 = 89 + 450 (91+91+91+91+91) / Jumlah Data *Testing* = *True Positive* + *True Negative* / Jumlah Data *Testing*
Datasets Sumber Berbeda	= **88.99% (0.8899)** = 455 / 545 = 67 + 388 (73+81+64+91+79) / Jumlah Data *Testing* = *True Positive* + *True Negative* / Jumlah Data *Testing*
Hasil *Testing* Keseluruhan	= **91,65% (0,9165)** = 999 / 1090 = 544 + 455 / 1090 = *True Positive* + *True Negative* / Jumlah Data *Testing*

4. KESIMPULAN

Hasil perancangan arsitektur model pada penelitian ini dapat disimpulkan bahwa jumlah *batch size* mempengaruhi jumlah *steps per epochs*, semakin tinggi nilai *batch size*, semakin kecil jumlah *steps per epochs*. Selain itu, jumlah *filters* pada *convolution layers* juga mempengaruhi banyaknya total *parameters* pada model. Jumlah *parameters* tertinggi dihasilkan oleh *dense layer* pertama, semakin tinggi jumlah *filters* pada *dense*, semakin tinggi pula jumlah *parameters* yang dihasilkan. Sehingga jika hendak menurunkan *Total Parameters*, maka yang harus dilakukan adalah dengan mengurangi *dense layer*, dan *convolution layer filter*. Semakin banyak Total *Parameters*, semakin lama pula waktu penyelesaian *training* model. Penggunaan *Android* sebagai *testing*/penerapan model *Machine Learning* yang memerlukan *hardware* pendukung seperti kamera akan lebih mudah digunakan karena dapat dibawa kemana saja untuk percobaan penerapan sistemnya, dan serta hemat biaya. Hasil *training accuracy* model didapatkan nilai 0,9820 (98,20%), *validation accuracy* bernilai 0,9922 (99,22%), *training loss* sebesar 0,0521 (05,21%), dan *validation loss* sebesar 0,0265 (02,65%). Sedangkan hasil *testing* didapatkan bahwa model dapat mengklasifikasi gambar yang bersumber sama maupun berbeda dari *datasets*

training dengan baik atau sebesar 91,65% dari 1090 citra. Ini menunjukkan bahwa model masuk dalam kategori *excellent classification*. Testing dilakukan melalui aplikasi yang telah ter-*deploy* model machine learning didalamnya menggunakan *smartphone Android*. Klasifikasi buah segar dan busuk (apel, pisang dan jeruk) dilakukan dengan total data *testing* 1090 yang mana 999 citra berhasil diklasifikasi dan 91 citra gagal diklasifikasikan sehingga bernilai *false positive* dan *false negative* atau gagal mengklasifikasikan data ke dalam *class*-nya masing – masing.

Saran untuk penelitian lanjutan yaitu penerapan algoritma *Convolutional Neural Network* dengan tambahan metode maupun arsitektur model lain seperti *K-Nearest Neighbor*, *Support Vector Machine*, *GoogleNet*, *AlexNet*, *VGG16*, *MobileNetV2*, maupun *Xception* akan memperkaya jumlah *trainable parameters* / *datasets* yang akan di-*training*. Selain itu penambahan *datasets* citra asli dari buah yang akan diklasifikasi agar model menjadi lebih terampil dalam melakukan klasifikasi.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh civitas akademika Institut Sains & Teknologi AKPRIND Yogyakarta yang telah membantu dan berkontribusi hingga terwujudnya karya ilmiah/penelitian ini.

DAFTAR PUSTAKA

Asrafil, A., Paliwang, A., Septian, M. R. D., Cahyanti, M., & Swedia, R. (2020). Klasifikasi Penyakit Tanaman Apel Dari Citra Daun Dengan Convolutional Neural Network. *Sebatik*, (1410–3737), 207–212.

Azis, N., Herwanto, H., & Ramadhani, F. (2021). Implementasi Speech Recognition Pada Aplikasi E-Prescribing Menggunakan Algoritme Convolutional Neural Network. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *5*(2), 460. https://doi.org/10.30865/mib.v5i2.2841

Dwi Antoko, T., Azhar Ridani, M., & Eko Minarno, A. (2021). Klasifikasi Buah Zaitun Menggunakan Convolution Neural Network. *Komputika : Jurnal Sistem Komputer*, *10*(2), 119–126. https://doi.org/10.34010/komputika.v10i2.4475

Géron, A. (2017). *Hands-on Machine Learning*. *O'Reilly Media, Inc*.

Kulshrestha, S. (2019). What Is A Convolutional Neural Network? https://doi.org/10.1007/978-1-4842-5572-8_6

Kurniadi, B. W., Prasetyo, H., Ahmad, G. L., Aditya Wibisono, B., & Sandya Prasvita, D. (2021). Analisis Perbandingan Algoritma SVM dan CNN Untuk Klasifikasi Buah. *Senamika*, *2*(2), 1–11. Retrieved from https://conference.upnvj.ac.id/index.php/senamika/article/view/1564

LeCun, Y., Yoshua, B., & Hinton. (2015). Deep Learning. *Nature*, *521*(7553), 436–444.

Maulana, F. F., & Rochmawati, N. (2020). Klasifikasi Citra Buah Menggunakan Convolutional Neural Network. *Journal of Informatics and Computer Science (JINACS)*, *1*(02), 104–108. https://doi.org/10.26740/jinacs.v1n02.p104-108

Ouaknine, A. (2018). Review of Deep Learning Algorithms for Object Detection | by Arthur Ouaknine | Zyl Story | Medium. Retrieved October 1, 2022, from https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Pujoseno, J. (2018). IMPLEMENTASI DEEP LEARNING MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI ALAT TULIS (Studi Kasus : Gambar alat tulis (Ballpoint, Penghapus dan Penggaris. *Gastrointestinal Endoscopy*.

Rizki, Y., Medikawati Taufiq, R., Mukhtar, H., & Putri, D. (2021). Klasifikasi Pola Kain Tenun Melayu Menggunakan Faster R-CNN. *IT Journal Research and Development*, *5*(2), 215–225. https://doi.org/10.25299/itjrd.2021.vol5(2).5831

Rucci, M., & Casile, A. (2005). Fixational instability and natural image statistics: Implications for early visual representations. In *Network: Computation in Neural Systems* (Vol. 16, pp. 121–138). https://doi.org/10.1080/09548980500300507

Sabilla, A. I. (2020). *Arsitektur Convolutional Neural Network (Cnn) Untuk Klasifikasi Jenis Dan Kesegaran Buah Pada Neraca Buah*. *Tesis*. Retrieved from https://repository.its.ac.id/73567/1/05111850010020-Master_Thesis.pdf

Sena, S. (2018). Pengenalan Deep Learning Part 8 : Gender Classification using Pre-Trained Network (Transfer Learning) | by Samuel Sena | Medium. Retrieved October 1, 2022, from https://medium.com/@samuelsena/pengenalan-deep-learning-part-8-gender-classification-using-pre-trained-network-transfer-37ac910500d1

Syahrudin, A., & Kurniawan, T. (2018). Input dan output pada bahasa pemrograman python. *Jurnal Dasar Pemrograman Python Stmik*, 1–7. Retrieved from https://www.researchgate.net/profile/Tedi-Kurniawan-2/publication/338385483_INPUT_DAN_OUTPUT_PADA_BAHASA_PEMROGRAMAN_PYTHON/links/5e10643392851c8364b029c3/INPUT-DAN-OUTPUT-PADA-BAHASA-PEMROGRAMAN-PYTHON.pdf