BAB II

LANDASAN TEORI

2.1. Dehidrasi

2.1.1. Pengertian Dehidrasi

Dehidrasi dapat didefinisikan sebagai penurunan total kadar air dalam tubuh baik karena kehilangan cairan (Contoh: berkeringat, buang air kecil, dll.) atau berkurangnya asupan cairan (Contoh: sedikit mengonsumsi buah dan sayuran, kurang minum, dll.). Ketidakseimbangan cairan karena dehidrasi ataupun overhidrasi berhubungan dengan morbiditas dan mortalitas terutama pada orang dewasa [17]. Tubuh memerlukan asupan cairan dalam jumlah tertentu setiap hari agar dapat berfungsi secara normal. Kebutuhan harian minimum asupan air adalah sekitar 1 Liter. Namun kebutuhan air tidak bisa didasarkan atas asupan minimum tersebut. Kebutuhan ini bervariasi sesuai dengan berat badan dan aktivitas individu. Setiap individu membutuhkan jumlah asupan air sebesar 67% dari berat badan (*ounces*) ditambah 12 *ounces* untuk setiap 30 menit melakukan aktivitas berat [18]. Dehidrasi terjadi ketika tubuh kehilangan cairan lebih banyak dari jumlah asupan yang dikonsumsi [9].

Keseimbangan antara jumlah asupan dan kehilangan air serta keseimbangan elektrolit pada tubuh sangat penting untuk menjaga kesehatan. Air berperan untuk mengatur suhu tubuh, membantu pencernaan makanan, dan membantu mengatur keseimbangan asam-basa dalam tubuh. Sedangkan

elektrolit -partikel yang akan menjelma menjadi ion negatif dan positif saat larut dalam air- berguna untuk menjaga kinerja saraf dan membantu kontraksi otot [19].

2.1.2. Dampak Fisiologis Dehidrasi

Beberapa dampak fisiologis yang dapat ditimbulkan akibat dehidrasi antara lain sebagai berikut [20]:

1. *Physical performance*

   Pada tingkat dehidrasi yang rendah, seseorang yang melakukan aktivitas fisik akan mengalami penurunan performa terkait penurunan daya tahan, mudah lelah, kemampuan termoregulasi -proses yang melibatkan sistem otonomi saraf tubuh dalam menjaga keseimbangan antara pembentukan panas dan kehilangan panas agar dapat mempertahankan suhu tubuh dalam kisaran normal [21]- yang berubah, berkurangnya motivasi, dan merasa membutuhkan upaya lebih untuk melakukan sesuatu. Sedangkan pada tingkat dehidrasi yang lebih tinggi, seseorang dapat kehilangan 6% - 12% berat badannya karena keluarnya keringat atau pengeluaran urine yang tidak diimbangi *supply* air yang cukup.

2. *Cognitive performance*

   Tingkat dehidrasi yang ringan dapat menyebabkan gangguan *mood* dan fungsi kognitif. Gangguan tersebut menjadi masalah yang perlu diperhatikan khususnya bagi anak-anak, orang berusia lanjut, orang yang tinggal di daerah yang beriklim panas, dan orang yang melakukan

olahraga berat. Dehidrasi ringan dapat menyebabkan perubahan dalam beberapa aspek penting terkait fungsi kognitif seperti gangguan konsentrasi, kewaspadaan, dan masalah memori jangka pendek pada anak-anak (5-12 tahun), orang dewasa (18-45 tahun), dan pada orang tua (46-82 tahun). Sama seperti fungsi fisik, dehidrasi ringan hingga sedang dapat menyebabkan gangguan kinerja terkait tugas-tugas seperti memori jangka pendek, diskriminasi persepsi, kemampuan berhitung, pelacakan *visuomotor*, dan keterampilan psikomotorik. Walaupun demikian, dehidrasi ringan tidak memberikan dampak terhadap perubahan fungsi kognitif secara konsisten.

3. *Kidney function*

Ginjal memiliki peran yang sangat penting dalam mengatur keseimbangan air dan tekanan darah serta menyaring dan membuang limbah dari tubuh. Regulasi air dimediasi melalui hormon agar kisaran osmolalitas plasma dapat dipertahankan antara 275 hingga 290 mOsm/kg. Terjadinya kenaikan jumlah osmolalitas plasma dan aktivasi *osmoreseptor* (intraseluler) serta *baroreseptor* (ekstraseluler) akan merangsang pelepasan hipotalamus *arginin vasopresin* (AVP). AVP yang terdapat di ginjal berfungsi untuk mengurangi volume urine dan meningkatkan retensi air beserta urine. Penurunan jumlah osmolalitas plasma menyebabkan penghambatan pelepasan vasopresin, sehingga menyebabkan ginjal akan meningkatkan keluaran urine. Selain menjaga keseimbangan cairan, ginjal membutuhkan air untuk menyaring limbah

dari aliran darah dan melakukan pembuangan melalui urine. Ekskresi air melalui ginjal menghilangkan zat terlarut dari darah dan dikeluarkan dalam bentuk urine. Sehingga apabila tubuh kekurangan cairan, maka ginjal tidak dapat melakukan semua fungsi tersebut dengan baik.

4. *Heart function*

Volume darah, tekanan darah, dan detak jantung berhubungan erat dengan dehidrasi. Volume darah dipertahankan dan diatur secara ketat dengan menjaga keseimbangan antara asupan air dan keluaran air, seperti yang dijelaskan pada bagian fungsi ginjal. Pada tubuh yang sehat, perubahan dalam detak jantung dan vasokonstriksi berfungsi untuk menjaga keseimbangan efek fluktuasi volume darah. Volume darah dapat mengalami penurunan akibat kehilangan darah, seperti melalui donor darah, atau kehilangan air dalam tubuh dalam jumlah besar, seperti melalui keringat setelah berolahraga. Jadi, kurangnya asupan air secara berangsur dapat mengurangi volume darah yang berdampak pada menurunnya denyut jantung dan meningkatnya tekanan darah.

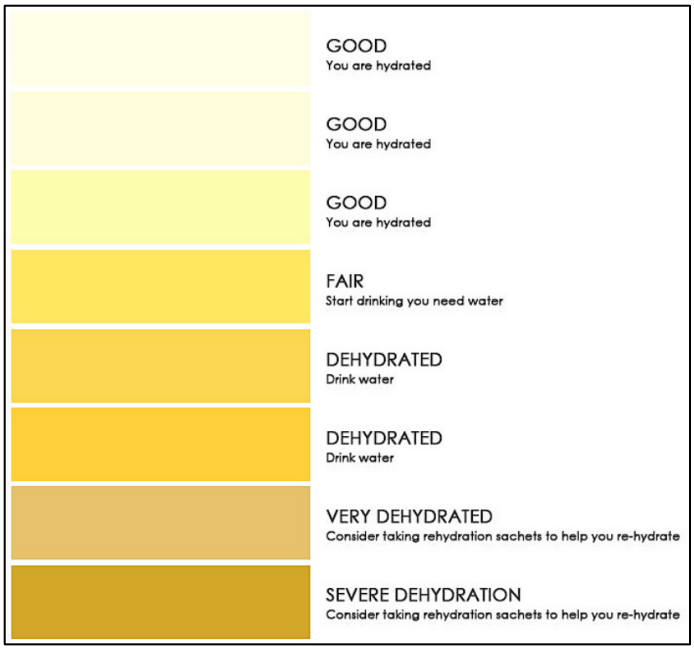
5. *Skin*

Salah satu fungsi asupan air yang cukup adalah untuk perbaikan kulit atau mengembalikan warna kulit, seperti akibat kecelakan atau sejenisnya. Setiap individu berusaha untuk menjaga agar kulit tetap dalam kondisi lembab, untuk mengurangi jerawat dan menghindari masalah kulit yang lainnya. Untuk itu diperlukan asupan air yang cukup untuk menjaga agar kulit tetap terhidrasi. Komposisi kulit terdiri dari sekitar 30% air yang

berkontribusi terhadap kelenturan, elastisitas, dan kekebalan. Oleh karena itu, apabila tubuh mengalami dehidrasi maka dapat menyebabkan berbagai masalah kesehatan kulit.

2.1.3. Tingkatan Dehidrasi

Beberapa studi yang telah dilakukan menyatakan bahwa warna urine berkorelasi dengan konsentrasi urine, sehingga dapat digunakan untuk memantau status hidrasi secara akurat. Perubahan substansial dalam warna urine terjadi setiap hari karena pengaruh perubahan jumlah asupan cairan. Hal ini menunjukkan bahwa individu dapat menggunakan warna urine untuk memantau status hidrasi mereka dengan cara yang sederhana [8].



Gambar 2.1 Tingkat Dehidrasi

Sumber: [8]

Perubahan warna urine sebesar dua tingkat pada skala delapan warna sesuai Gambar 2.1, dapat dicapai dengan adanya perubahan dalam asupan air sekitar 1200 mL/hari. Selain itu, perubahan warna urine juga berhubungan erat dengan perubahan volume urine dan massa jenisnya [22]. Sehingga penelitian ini menggunakan warna urine sebagai indikator status hidrasi.

2.2. Density

Massa jenis (*density*) atau disebut juga dengan istilah kerapatan massa adalah perbandingan antara massa suatu zat dengan volumenya yang biasa diungkapkan sebagai massa per satuan volume. Massa jenis merupakan properti unik yang dimiliki oleh setiap zat atau dapat dibilang massa jenis adalah ciri khas setiap zat. Hal ini menyebabkan zat yang berbeda jenis akan memiliki massa jenis yang berbeda pula [23]. *Density* dapat digambarkan melalui persamaan berikut:

$$\rho = \frac{m}{V}$$

Rumus 2.1 Massa Jenis

Sumber: [23]

Keterangan:

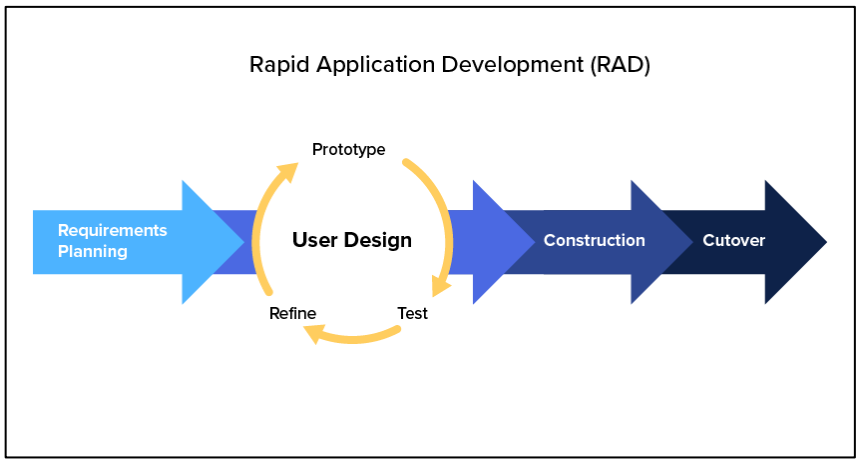
- ρ (rho) = *density* (g/cm^3 atau g/mL)
- m = massa benda (g)
- V = Volume (cm^3 atau mL)

Massa jenis urine dapat digunakan sebagai indikator dehidrasi. Massa jenis urine di kategorikan menjadi empat, yaitu normal apabila massa jenis urine lebih kecil dari 1,015 g/mL, dehidrasi ringan apabila massa jenis urine 1,016 g/mL –

1,025 g/mL, dehidrasi sedang apabila massa jenis urine 1,026 g/mL –1,030 g/mL, dan dehirasi berat apabila massa jenis urine lebih besar dari 1,030 g/mL [24].

2.3. Rapid Application Development (RAD)

RAD adalah salah satu metode pengembangan sistem dengan durasi yang singkat dan dapat menghasilkan sebuah sistem informasi yang berkualitas [25]. Penggunaan RAD sebagai metode pengembangan sistem memberikan beberapa keuntungan dibandingkan dengan metode tradisional (Contoh: *waterfall*), yaitu durasi pengembangan sistem yang lebih cepat, kualitas sistem yang lebih baik, dan biaya yang dibutuhkan untuk pemeliharaan lebih rendah. Selain itu, sistem yang dikembangkan bersifat mudah dipahami dan fleksibel terhadap perubahan kebutuhan pengguna. RAD melewati tahap *planning* dengan durasi yang singkat karena memberikan fokus utama pada tahap *development*, *testing*, dan *feedback* [26]. Metode ini memiliki empat tahapan utama, yaitu *Requirement Planning*, *User Design*, *Construction*, dan *Custover* yang dapat dilihat pada Gambar 2.2 [27].

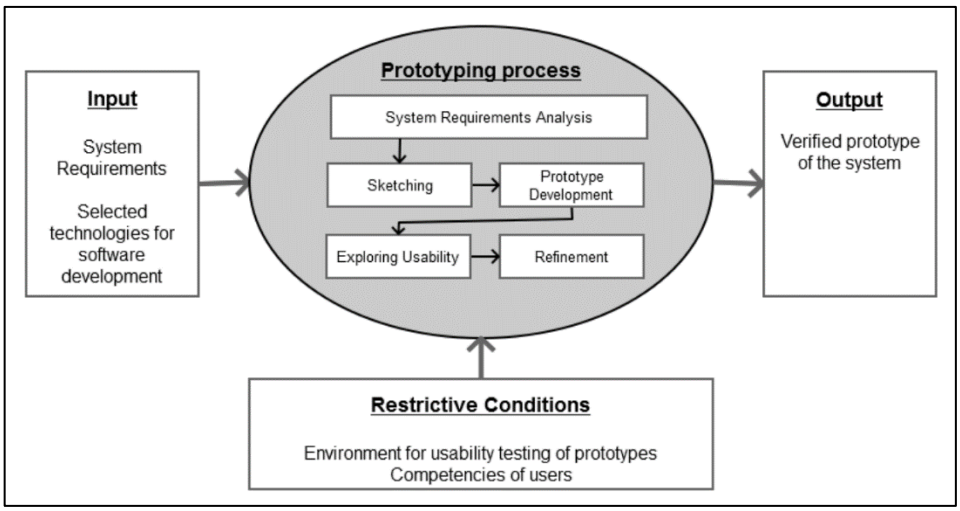


Gambar 2.2 Metode Pengembangan Sistem RAD

Sumber: [27]

2.4. Prototyping

Prototyping merupakan salah satu metode pengembangan sistem yang sangat baik digunakan untuk tujuan demonstrasi atau sebagai bagian dari proses pengembangan atau pembuatan sebuah sistem atau *software*. Proses pengembangan sistem dengan *prototyping* relatif cepat, sehingga banyak digunakan sebagai model kerja dasar untuk merekayasa kebutuhan pengguna secara berulang-ulang. Dengan demikian, maka dapat meminimalkan kesalahpahaman kebutuhan atau *requirement* antara pengguna dengan pihak pengembangan sistem. Akan tetapi, *prototyping* cenderung menggunakan algoritma dan bahasa pemrograman yang sederhana, agar proyek dapat diselesaikan dengan cepat, sehingga berdampak pada lemahnya keamanaan dari sistem [28]. Terdapat lima tahapan utama dalam metode *prototyping*, yaitu *System Requirement Analysis*, *Sketching*, *Prototype Development*, *Exploring Usability*, dan *Refinement* seperti yang dapat dilihat pada Gambar 2.3 [29].

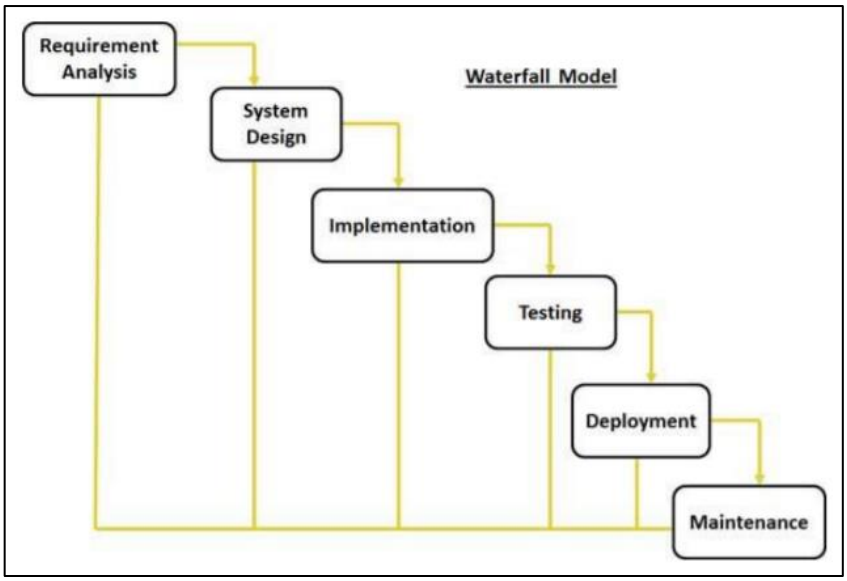


Gambar 2.3 Metode Pengembangan *Prototyping*

Sumber: [29]

2.5. Waterfall

Metode *waterfall* telah dikenal sebagai *the classic life cycle*, yang menyarankan pendekatan secara sistematis dan berurutan untuk melakukan pengembangan *software* [30]. Metode ini bersifat sekuensial linear, dikarenakan setiap tahap pengembangan dapat dilewati apabila tahap sebelumnya telah diselesaikan [31]. *Output* dari tahap sebelumnya menjadi masukkan bagi tahap berikutnya, sehingga membuat metode ini tidak fleksibel apabila terdapat perubahan kebutuhan pengguna karena harus melewati berbagai rangkaian mekanisme yang panjang. Terdapat enam tahapan utama dalam metode *waterfall*, yaitu *Requirement Analysis*, *System Design*, *Implementation*, *Testing*, *Deployment*, dan *Maintenance* yang dapat dilihat pada Gambar 2.4 [32].

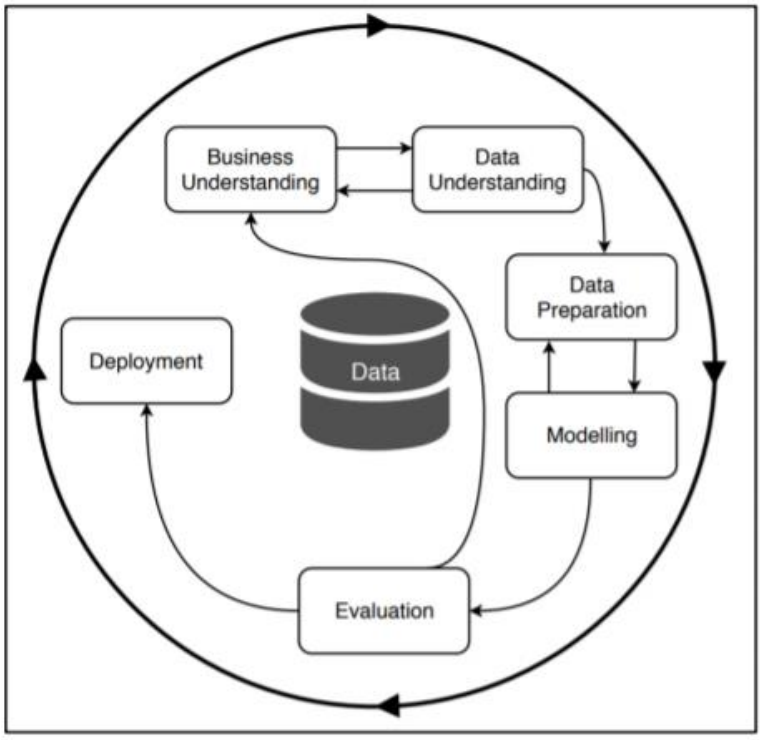


Gambar 2.4 Metode Pengembangan *Waterfall*

Sumber: [32]

2.6. Cross-Industry Standard Process for Data Mining (CRISP-DM)

CRISP-DM adalah suatu *framework* untuk menerjemahkan masalah bisnis ke dalam tugas *data mining* dan menghasilkan suatu proyek *data mining* yang independen, serta mencakup area bisnis dan teknologi [33]. CRISP-DM memiliki *life-cycle* yang tersusun dari enam tahapan, yaitu *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan *deployment* yang dapat dilihat pada Gambar 2.5.



Gambar 2.5 Tahapan CRISP-DM

Sumber: [33]

Enam tahapan dari CRISP-DM adalah sebagai berikut [34]:

1. *Business Understanding*

   Tahap ini merupakan tahapan paling awal yang bertugas untuk memahami tujuan dan kebutuhan dari bisnis, mempersiapkan strategi untuk mencapainya, dan merancang apa yang akan dibangun.

2. *Data Understanding*

   Tugas penting pada tahap ini adalah mengumpulkan data yang diperlukan, mengeksplorasi dan mendeskripsikan data, memeriksa kualitas data, dan mendeteksi pola yang menarik dari data.

3. *Data Preparation*

   Tugas penting pada tahap ini adalah mempersiapkan data yang meliputi *pre-processing*, menyaring data, dan melakukan ekstraksi fitur sesuai dengan tujuan proyek.

4. *Modeling*

   Tugas penting pada tahap ini adalah menerapkan teknik pemodelan yang sesuai dengan menentukan algoritma dan penyesuaian parameter agar diperoleh performa model yang maksimal.

5. *Evaluation*

   Pada tahap ini dilakukan pengujian model yang telah dibangun terhadap data baru dengan tujuan untuk mengetahui kualitas model sebelum dilakukan *deployment*.

6. *Deployment*

   Tahap ini merupakan tahapan terakhir yang bertugas untuk mempresentasikan model yang dibangun dalam bentuk khusus agar dapat digunakan oleh pengguna. Dalam beberapa kasus, tahap ini juga melibatkan konsumen. Penting bagi konsumen untuk memahami dalam memanfaatkan model yang telah dibuat.

2.7. Deep Learning

Deep learning merupakan sub-bidang dari *machine learning* yang berkaitan dengan algoritma yang mengikuti bagaimana sistem dasar otak manusia bekerja. *Deep learning* menganalisis data dengan representasi yang telah dipelajari, serupa dengan cara manusia memandang suatu masalah [35]. *Deep learning* berbeda dengan *machine learning* tradisional, di mana pada *machine learning* tradisional sistem diberikan sekumpulan fitur yang relevan untuk dianalisis. Sedangkan dalam *deep learning*, sistem hanya diberikan data mentah dan dapat dengan sendirinya melakukan rekayasa fitur. Performa *deep learning* umumnya akan meningkat apabila jumlah data yang digunakan untuk pelatihan ditingkatkan [36].

Berikut ini beberapa kelebihan penggunaan *deep learning* antara lain [35]:

1. *Maximum utilization of unstructured data*

   Format data yang berbeda-beda masih dapat digunakan untuk melatih model *deep learning* dan memperoleh *insight* yang relevan dengan tujuan pelatihan. Misalnya, penggunaan algoritma *deep learning* untuk mengungkap hubungan yang ada antara analisis industri, *public sentiment*,

dan sejenisnya untuk memprediksi harga saham yang akan datang dari organisasi tertentu.

2. *Elimination of the need for feature engineering*

   Rekayasa fitur dalam *machine learning* adalah fondasi dalam pengerjaan, karena akan meningkatkan akurasi dan kadang dalam prosesnya diperlukan pengetahuan tentang domain atau masalah tertentu. Keuntungan terbesar dalam menggunakan *deep learning* adalah kemampuannya menjalankan rekayasa fitur secara otomatis.

3. *High-quality results*

   Manusia sering melakukan kesalahan ataupun kecerobohan jika dalam melakukan tugas rutinitasnya, misalnya karena lelah, haus, maupun lapar. Tidak demikian dengan *neural network*, setelah mendapat pelatihan yang baik, model *deep learning* mampu melakukan ribuan tugas rutin dan berulang-ulang dalam waktu yang sangat singkat dibandingkan dengan waktu yang dibutuhkan manusia. Kualitas pekerjaannya pun tidak pernah menurun. Model d*eep learning* hanya memerlukan pelatihan yang berisi data mentah yang mewakili masalah yang ingin diselesaikan.

4. *Reusability*

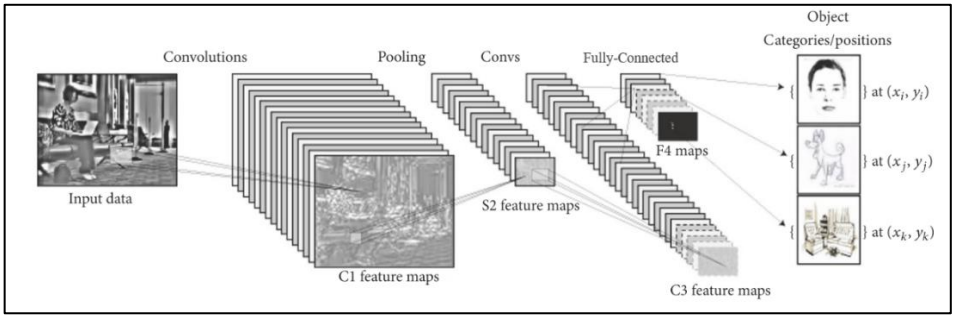
   Dengan *deep learning* apabila suatu model ingin dilatih menggunakan data tambahan, proses pelatihan tidak perlu diulang dari awal sehingga pembelajaran secara berkesinambungan dapat dilakukan. Hal ini sangat penting jika model yang dilatih berskala besar. Selain itu, untuk proyek yang

akan datang, model *deep learning* yang telah dibangun tersebut dapat digunakan kembali apabila masalah yang dihadapi masih relevan.

2.8. Convolutional Neural Network (CNN)

2.8.1. Algoritma CNN

CNN salah satu jenis algoritma *neural network* yang menggunakan orientasi *deep learning* yang banyak digunakan dalam masalah pengenalan objek atau gambar. CNN dapat menerima *input* berupa gambar, menentukan aspek atau fitur apa saja dalam sebuah gambar yang bisa digunakan oleh sistem untuk belajar mengenali gambar tersebut, dan membedakan antara satu gambar dengan yang lainnya. Yang membedakan antara CNN dan *neural network* adalah fitur yang digunakan pada CNN berupa nilai setiap piksel dari gambar yang di-*input*. Setiap gambar akan diubah menjadi matriks tiga dimensi yang terdiri dari lebar, tinggi, dan warna. CNN akan memecah setiap gambar menjadi gambar yang lebih kecil dan tumpang tindih. Selanjutnya memanfaatkan kernel konvolusi atau filter berukuran tertentu ke gambar tersebut untuk menemukan sebuah pola tertentu seperti dapat dilihat pada Gambar 2.6 [36].



Gambar 2.6 Contoh Arsitektur CNN dalam Pengenalan Objek atau Gambar

Sumber: [36]

Terdapat beberapa lapisan atau *layer neural network* yang terdapat pada CNN seperti pada Gambar 2.6. Lapisan yang ada pada CNN adalah sebagai berikut [36]:

1. *Convolutional Layer*

   Convolutional layer adalah lapisan yang melakukan ekstraksi fitur berupa piksel dari bagian yang paling penting pada area tertentu.

2. *Pooling Layer*

   Pooling layer bertujuan untuk menangkap informasi penting hasil dari ekstraksi fitur dan secara bertahap mengurangi dimensi representasi. Selanjutnya akan dilakukan pengurangan sejumlah parameter untuk mengurangi kompleksitas komputasi model. Metode *pooling* yang biasa dipakai adalah *Max Pooling* dengan mencari nilai terbesar dan *Average Pooling* dengan mencari nilai rata-rata.

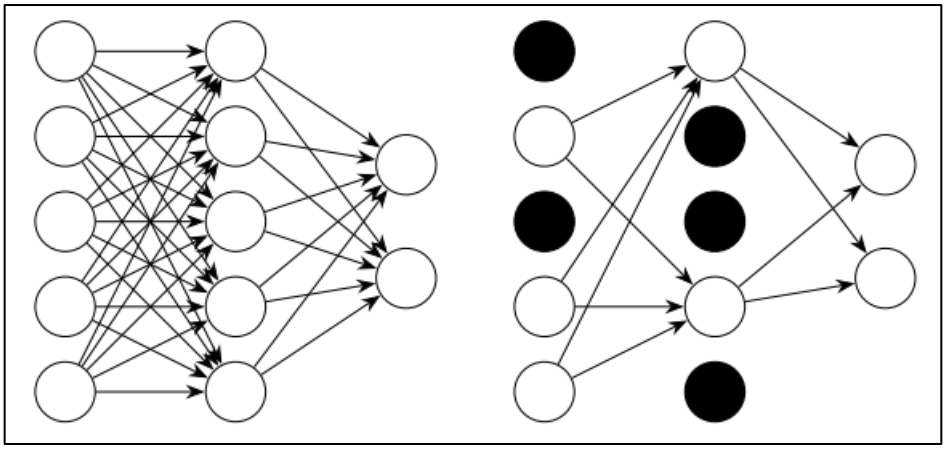
3. *Fully Connected Layer*

   Fully connected layer adalah lapisan dimana semua aktivitas dari *neuron* atau *layer* sebelumnya dikumpulkan agar dapat dilakukan komputasi dalam menentukan pembagian kelas. Melalui *fully connected layer* akan diperoleh probabilitas gambar *input* terhadap berbagai kelas sesuai dari pelatihan yang telah dilakukan.

2.8.2. Dropout

Dropout adalah salah satu teknik regularisasi dimana beberapa *neuron* akan dipilih secara acak untuk tidak dipakai selama pelatihan. Tujuan dari *dropout* adalah untuk mencegah *overfitting* dan mempercepat proses

pembelajaran khususnya pada jaringan yang memiliki banyak lapisan atau *neuron*. Dengan melakukan *dropout* beberapa *neuron* pada *layer* akan diubah nilainya menjadi 0 untuk setiap iterasi pada fase pelatihan seperti dapat dilihat pada Gambar 2.7 [37].



Gambar 2.7 Contoh *Dropout* 5% *Neuron*

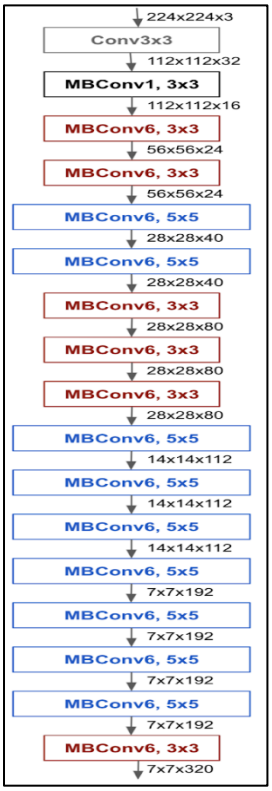
Sumber: [37]

Gambar 2.7 bagian kiri adalah jaringan dengan *fully connected layer*, sedangkan pada bagian kanan merupakan jaringan dengan *dropout neuron* sebesar 0,5 atau 5% [37].

2.8.3. *Pre-Trained* Model EfficientNet

EfficientNet adalah salah satu *pre-trained* model CNN yang dikembangkan oleh Google pada tahun 2019. EfficientNet termasuk salah satu teknik *transfer learning* yang dikhususkan untuk masalah pengenalan objek atau klasifikasi gambar. Model EfficientNet saat ini memiliki 8 model, yaitu EfficientNetB0 sampai EfficientNetB7 yang memiliki jumlah parameter yang semakin banyak dari 5,3 juta sampai 66 juta parameter diikuti dengan nilai

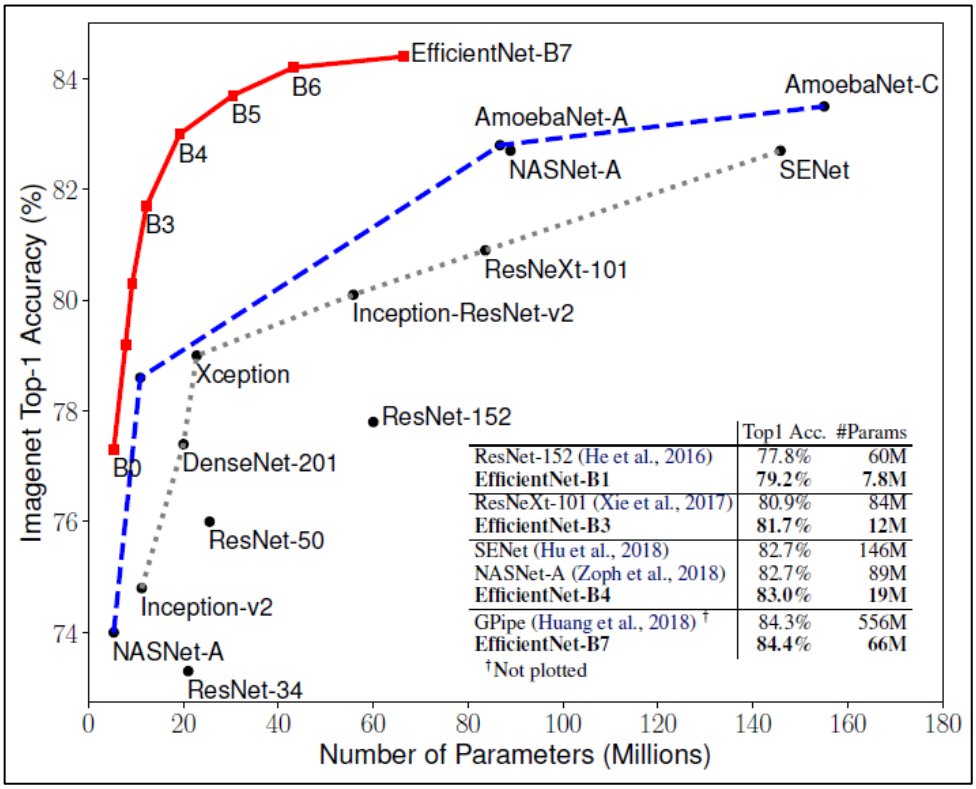
akurasi yang semakin tinggi. Arsitektur EfficientNet dapat dilihat pada Gambar 2.8 [16].



Gambar 2.8 Arsitektur EfficientNet

Sumber: [16]

Pengembangan EfficientNet difokuskan pada kecepatan waktu dan kemampuan komputasi. Hasil akurasi dari EfficientNet lebih tinggi dibandingkan dengan *pre-trained* model yang lain dengan jumlah parameter yang lebih sedikit. Selain itu, EfficientNet juga membutuhkan waktu komputasi yang lebih cepat seperti yang dapat dilihat pada Gambar 2.9 [16].



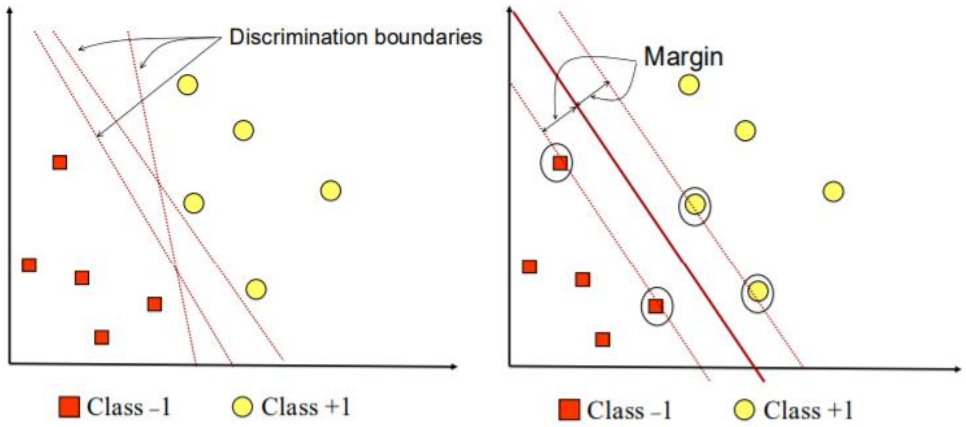
Gambar 2. 9 Grafik Perbandingan Berbagai *Pre-Trained* Model

Sumber: [16]

2.9. Support Vector Machine (SVM)

SVM adalah salah satu metode *machine learning* berupa *supervised learning* yang sudah digunakan secara luas di berbagai bidang, dengan keunggulan, seperti mudah diimplementasikan dan mudah dalam melakukan optimalisasi [38]. Konsep SVM secara sederhana dapat diartikan sebagai upaya mencari *hyperplane* terbaik yang bertujuan untuk memisahkan dua buah kelas pada *input space* [39]. Gambar 2.10 menunjukkan beberapa *pattern* yang menjadi anggota dari dua buah kelas, yaitu +1 dan –1. *Pattern* yang termasuk pada kelas –1 dilambangkan dengan warna merah (persegi), sedangkan *pattern* yang termasuk pada kelas +1, dilambangkan

dengan warna kuning (lingkaran). Masalah klasifikasi dapat diartikan sebagai upaya untuk mencari garis (*hyperplane*) terbaik yang membedakan antara kedua kelompok tersebut [39]. Beberapa pilihan garis pemisah (*discrimination boundaries*) dapat dilihat pada Gambar 2.10 bagian kiri.



Gambar 2.10 Mencari *hyperplane* terbaik yang memisahkan kedua kelas –1 dan +1

Sumber: [39]

Hyperplane terbaik yang memisahkan kedua kelas didapatkan dengan mengukur *margin hyperplane* tersebut dan mencari nilai maksimalnya. *Margin* merupakan jarak antara *hyperplane* terhadap *pattern* yang paling dekat dari setiap kelas. *Pattern* terdekat tersebut dikenal dengan nama *support vector*. Garis yang lebih tebal pada Gambar 2.10 bagian kanan adalah *hyperplane* terbaik, yaitu yang berada tepat di tengah-tengah kedua kelas, sedangkan titik merah dan kuning yang dilingkari adalah *support vector*. Upaya untuk menemukan koordinat *hyperplane* ini adalah tugas utama pada SVM.

Hyperplane terbaik dapat ditemukan dengan memaksimalkan *margin* pada *support vector* dengan *quadratic programming* yang mencari titik minimal persamaan, yang digambarkan melalui Rumus 2.2 berikut:

$$\min \tau(w) = \frac{1}{2}\|\vec{w}\|^2$$

Dengan *constraint*: $y_i(\overrightarrow{x_i}\vec{w} + b) - 1 \geq 0, \forall i$

Rumus 2.2 Maksimum *Margin* pada *Hyperplane*

Sumber: [40]

Keterangan:

- x_i = data *input*
- y_i = data *output* yang berasal dari x_i
- w, b = *unknown parameter* yang akan dicari nilainya

Rumus 2.2 merupakan *formula* untuk mencari *hyperplane* pada SVM yang berbasis linear. Kebanyakan kasus nyata di dunia adalah yang tidak linear [40]. Misalnya masalah pendeteksian status hidrasi berdasarkan warna urine. Terkait penyelesaian masalah yang tidak berbasis linear, SVM dapat menggunakan *kernel function*. Tabel 2.1 menunjukkan tiga buah *kernel function* yang umumnya dipakai dalam SVM.

Tabel 2.1 SVM *Kernel Function*

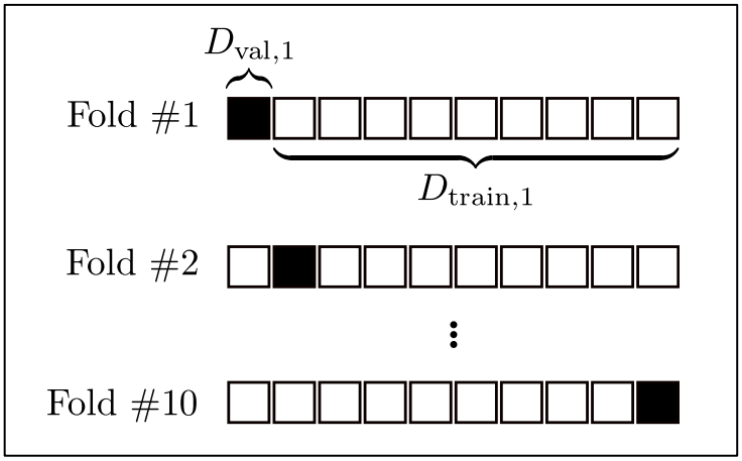
No	*Kernel Function*	*Formula*
1	*Linear*	${x_i}^T x_j + 1$
2	*Polynomial*	$({x_i}^T x_j + 1)^r$
3	*Radial Basis Function*	$\exp - \frac{\|\|x - x_i\|\|^2}{2\sigma^2}$

Sumber: [40]

Pada awalnya SVM hanya digunakan untuk melakukan klasifikasi dua kelas saja, tetapi saat ini SVM dapat menyelesaikan masalah klasifikasi lebih dari dua kelas yang dikenal dengan nama *multiclass* SVM [40].

2.10. K-Fold Cross Validation

K-Fold Cross Validation adalah salah satu cara untuk melakukan *cross validation* dengan melipat data sebanyak k-partisi dan melakukan iterasi eksperiman sebanyak k-kali. Dengan *k-fold cross validation* data akan dibagi menjadi dua bagian besar. Bagian pertama sebagai data *training* untuk melatih model, bagian kedua sebagai data *testing* untuk validasi model. Proses validasi diawali dengan membagi *dataset* menjadi k partisi secara acak. Kemudian dilakukan eksperimen sebanyak k-kali dan setiap pengulangan ekperimen data bagian ke-k digunakan sebagai data *testing* sedangkan bagian sisanya sebanyak (k-1) digunakan sebagai data *training* [41]. Gambar 2.11 berikut ini adalah contoh *k-fold cross validation* dengan k sebanyak 10.



Gambar 2.11 *10-Fold Cross Validation*

Sumber: [41]

Contoh pada gambar 2.11 dimana K = 10. Artinya data dibagi menjadi 10 lipatan atau partisi, 1 partisi dipakai sebagai data untuk pelatihan dan 9 partisi sisanya digunakan sebagai data untuk validasi. Penggunaan *k-fold cross validation* adalah untuk meningkatkan akurasi performa model serta untuk menghindari terjadinya *overfitting* [41].

2.11. Confusion Matrix

Confusion matrix merupakan salah satu metode untuk mengetahui hasil performa dari model yang telah dibangun terkait masalah klasifikasi. *Confusion matrix* dapat diterapkan pada *binary classification* dan juga pada *multiclass classification*. Matriks ini berisi perbandingan antara jumlah dari nilai yang diprediksi dan nilai actual dengan pembagian *True Negative* (TN), *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN) yang dapat dilihat pada Gambar 2.12 [42].

		Predicted	
		Negative	Positive
Actual	**Negative**	TN	FP
	Positive	FN	TP

Gambar 2.12 *Confusion Matrix*

Sumber: [42]

Pada Gambar 2.12, TN menunjukkan jumlah data yang secara akurat diklasifikasikan sebagai negatif. TP menunjukkan jumlah data yang secara akurat diklasifikasikan sebagai positif. FP menunjukkan jumlah data yang seharusnya negatif tetapi diklasifikasikan sebagai positif. FN menunjukkan jumlah data yang seharusnya positif tetapi diklasifikasikan sebagai negatif. Nilai akurasi dari suatu

model klasifikasi dapat dihitung dengan *formula* yang dapat dilihat pada Rumus 2.3 [42].

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN}$$

Rumus 2.3 Menghitung Akurasi dari *Confusion Matrix*

Sumber: [42]

2.12. Tools

2.12.1. Python

Bahasa pemrograman Python diciptakan oleh Guido Van Rossum pada tahun 1989. Pyhton merupakan sebuah *interpreter* yang dikembangkan sebagai proyek *open source*. Python merupakan bahasa pemrograman yang memiliki banyak fungsi, interaktif, dan mendukung pemrograman berorientasi objek [43]. Beberapa keunggulan bahasa pemrograman Python adalah sebagai berikut [44]:

1. *Cross-Platfrom Language*

   Program yang ditulis dengan Python dapat dijalankan di bawah banyak sistem operasi, yaitu Microsoft Windows, Linux dan Unix seperti Mac OS, disertai dengan dukungan yang hampir lengkap dari *library* atau modul pihak ketiga hanya dengan menyalin *source code* program.

2. *Data Structure Readability*

   Struktur kode Python mudah dipahami dengan tipe data yang lengkap, seperti *lists*, *tuples*, *sets*, *dictionaries*, *strings*, dan lain-lain.

3. *Fast and Powerful*

   Python telah menyediakan *standard library* yang lengkap dan masih banyak *library* yang dapat ditambahkan secara personal. Penambahan modul atau *library* di luar *standard library* dapat dilakukan secara gratis.

4. *Automation*

   Python dapat banyak membantu dalam melakukan tugas secara otomatis. Hal ini mungkin dilakukan karena *library* dan modul yang tersedia sangat lengkap, sehingga pengguna dapat menggunakan fungsi berbagai algoritma dengan mudah hanya dengan menggunakan *library* atau modul Python yang diperlukan.

2.12.2. Tensorflow

Tensorflow merupakan *framework machine learning* yang dikembangkan oleh tim Google Brain dengan *library open source* yang biasanya digunakan pada *machine learning* berskala besar. Dalam pengembangan suatu aplikasi, Tensorflow menyediakan API dalam bahasa pemrograman Python. Pada *framework* ini terdapat *construction phase* yang berkaitan dengan pembangunan model *machine learning* dan *execution phase* yang berkaitan dengan eksekusi model mulai dari *input* sampai *output*. Keuntungan dari penggunaan Tensorflow adalah penyediaan *high-level* API, seperti Keras yang membuat pengguna semakin mudah dalam penggunaan fungsi-fungsi yang disediakan. Selain itu, Tensorflow juga mendukung penggunaan GPU untuk mempercepat proses pelatihan model [45].

2.12.3. Flask

Flask merupakan web *framework* dari bahasa pemrograman Python dan termasuk *micro framework* karena tidak memerlukan *tools* atau *library* tambahan tetapi tetap mendukung *extensions* penambahan fitur pada aplikasi. Contohnya *form validation, encrypt password, login user, create session, logout user, upload handling*, dan lainnya. Keuntungan dari penggunaan Flask antara lain penyediaan fitur yang sederhana, fleskibel dan mudah dalam konfigurasi, serta didukung dengan pengembangan server *built-in*, sehingga cocok untuk diimplementasikan dalam pengembangan web [46].

2.13. Penelitian Terdahulu

Tabel 2.2 Penelitian Terdahulu

No	Jurnal	*Authors*	Hasil Penelitian	Kontribusi bagi Penelitian Ini
1	IEEE Access. *Boundary Delineation of MRI Images for Lumbar Spinal Stenosis Detection Through Semantic Segmentation Using Deep Neural Networks*, 7, 43487-43501. (2019). [47]	Ala S. Al-Kafri, Sud Sudirman, Abir Hussain, Dhiya Al-Jumeily, Friska Natalia, Hira Meidia, Nunik Afriliana, Wasfi Al-Rashdan, Mohammad Bashtawi, dan Mohammed Al-Jumaily	Penelitian tersebut menghasilkan sistem yang mampu mendeteksi kelainan stenosis tulang belakang melalui MRI *images*. Pengembangan model dilakukan menggunakan pendekatan *deep neural network*, yaitu *semantic segmentation* dengan CNN.	Berkontribusi sebagai referensi dalam penggunaan *deep learning* CNN untuk melakukan klasifikasi tingkat dehidrasi.

No	Jurnal	*Authors*	Hasil Penelitian	Kontribusi bagi Penelitian Ini
2	International Conference on Machine Learning. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, 36. (2019). [16]	Mingxing Tan dan Quoc V. Le	Penelitian tersebut menghasilkan *pre-trained* model CNN, yaitu EfficientNet yang telah teruji memiliki akurasi yang lebih tinggi daripada *pre-trained* model CNN terdahulu dalam pengenalan objek gambar.	Berkontribusi sebagai referensi pemilihan arsitektur EfficientNet pada model CNN.
3	Journal of Big Data. *A survey on Image Data Augmentation for Deep Learning*, 6(60). (2019). [48]	Connor Shorten dan Taghi M. Khoshgoftaar	Penelitian tersebut berhasil menemukan suatu metode untuk meningkatkan hasil akurasi, yaitu dengan penggunaan *data augmentation.* Selain itu, penelitian tersebut juga menggunakan *pre-trained* model CNN, yaitu VGG-16, ResNet, Inception-V3, dan DenseNet.	Berkontribusi sebagai alasan dilakukan penerapan *data augmentation* dan penggunaan *pre-trained model*.
4	Media Gizi Indonesia. Hubungan Asupan Cairan, Status Gizi Dengan Status Hidrasi Pada Pekerja Di Bengkel Divisi General, 12(1), 47-53. (2019). [24]	Nika Anita Sari dan Triska Susila Nindya	Penelitian tersebut berhasil membuktikan massa jenis urine berkorelasi dengan tingkat dehidrasi.	Berkontribusi sebagai metode yang digunakan untuk memperoleh deskripsi label tingkat dehidrasi pada dataset.

Perbandingan antara penelitian terdahulu dengan penelitian ini antara lain:

1. Penelitian pertama pada Tabel 2.3 menggunakan metode *semantic segmentation* CNN dalam pengembangan model. Sedangkan penelitian ini menggunakan *pre-trained model* CNN.
2. Penelitian kedua pada Tabel 2.3 menjadi referensi pemilihan arsitektur EfficientNet pada model CNN.
3. Penelitian ketiga pada Tabel 2.3 menggunakan arsitektur CNN VGG-16, ResNet, Inception-V3, dan DenseNet. Sedangkan penelitian ini menggunakan arsitektur EfficientNet.
4. Penelitian keempat pada Tabel 2.3 melakukan deteksi dehidrasi berdasarkan asupan cairan. Sedangkan penelitian ini melakukan deteksi dehidrasi melalui pendeketan deep learning berdasarkan warna urine.
5. Perbedaan lain antara penelitian terdahulu dengan penelitian ini terletak pada bentuk luaran penelitian. Pada penelitian ini dihasilkan bentuk luaran berupa aplikasi berbasis web yang dapat mendeteksi tingkat dehidrasi.