

SKRIPSI

**DETEKSI MANIPULASI CITRA DENGAN *ERROR LEVEL ANALYSIS*
DAN *DEEP LEARNING***

**Disusun dan Diajukan Oleh
RIZKY ALFIANSYAH
D421 16 007**



**DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
MAKASSAR
2022**

KATA PENGANTAR

Assalamualaikum Warahmatullaahi Wabarakatuh

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, yang telah membagikan rahmat serta karunia- Nya, sehingga penulis bisa menyelesaikan laporan tugas akhir dengan judul "**Deteksi Pemalsuan Citra Dengan Error Level Analysis Dan Deep Learning**". Tak lupa pula Shalawat dan salam senantiasa tercurah kepada Rasulullah SAW yang mengantarkan manusia dari zaman kegelapan ke zaman yang terang benderang ini. Laporan tugas akhir ini ialah salah satu syarat untuk mendapatkan gelar Sarjana Strata Satu (S1) pada Departemen Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari bahwa penulisan ini tidak dapat terselesaikan tanpa dukungan dari berbagai pihak baik moril maupun materil. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada semua pihak yang telah membantu dalam penyusunan skripsi ini terutama kepada:

1. Kedua orang tua, Bapak Sukiman dan Ibu Sitti Wajida yang telah memberikan dukungan baik moril maupun materil serta senantiasa memberikan doa yang tiada henti-hentinya kepada penulis.
2. Ibu Dr. Ir. Ingrid Nurtanio, M.T., selaku dosen pembimbing I dan Bapak Dr. Indrabayu, S.T., M.T., M.Bus.Sys., selaku dosen pembimbing II, yang selalu menyediakan waktu, tenaga dan pikirannya yang luar biasa untuk mengarahkan penulis dalam penyusunan Tugas Akhir ini.

3. Bapak Dr. Amil Ahmad Ilham, S.T., M.I.T., selaku Ketua Departemen Teknik Informatika Fakultas Teknik Univeristas Hasanuddin atas bimbingannya selama masa perkuliahan.
4. Segenap Dosen dan Staff Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin, yang telah banyak membantu penulis selama masa perkuliahan.
5. AIMP Family, yang telah memberikan begitu banyak bantuan selama penelitian, pengambilan data, dan diskusi terkait penyusunan skripsi.
6. Saudara-saudari IGNITER 16, yang selalu menyemangati dan membantu penyelesaian skripsi ini serta mengisi hari-hari menjadi sangat menyenangkan.
7. Semua pihak atas dukungan dan bantuannya yang tidak dapat penulis tuliskan satu persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga skripsi ini dapat memberikan manfaat untuk mendorong penelitian penelitian selanjutnya.

Gowa, ____ 2022

Rizky Alfiansyah
D42116007

ABSTRAK

Pemalsuan citra digital telah menjadi salah satu taktik penyebar hoax yang merupakan ajang penyebar provokasi, menimbulkan kerusuhan serta kebencian. Untuk mengidentifikasi suatu citra asli atau palsu, sangat sulit dilihat dengan mata telanjang, diperlukan teknik-teknik khusus dan ketelitian tertentu agar dapat mengetahui dengan pasti suatu citra merupakan citra asli atau sudah mengalami modifikasi. Bagi orang awam, hal ini mungkin sulit untuk dilakukan. Mesin dapat melakukannya dengan mudah untuk sejumlah besar data dengan bantuan teknik kecerdasan buatan yang digunakan untuk mengembangkan sistem yang akan membantu dalam mengenali apakah citra tersebut asli atau palsu. Teknologi ini membutuhkan banyak data citra, dan setiap citra memiliki banyak pixel-pixel penyusunnya. **Metode Convolutional Neural Network (CNN)** adalah salah satu algoritma dari *Deep Learning* yang merupakan pengembangan dari *Multi Layer Perceptron (MLP)* yang dirancang untuk mengolah data citra. CNN bisa digunakan untuk mendeteksi dan mengenali object pada sebuah *image*. Oleh karena itu pada penelitian ini membahas tentang implementasi metode metode *Error Level Analysis* untuk *preprocessing* dan klasifikasi menggunakan metode *Convolutional Neural Network*. yang dilakukan pada sekumpulan data untuk mengetahui model yang paling cocok untuk pengembangan sistem pengklasifikasi citra asli dan palsu. **Hasil** pengujian deteksi citra, *system* mampu mendeteksi citra dengan total akurasi rata-rata mencapai 91,71% Real, 83,51% Splice, 72,26% Copy Move.

Kata Kunci : Deteksi pemalsuan citra, *Error Level Analysis*, *Convolutional Neural Network*

DAFTAR ISI

KATA PENGANTAR	i
ABSTRAK	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	iv
BAB I PENDAHULUAN	1
7.1. Latar Belakang	1
7.2. Rumusan Masalah.....	3
3.1. Tujuan Penelitian.....	4
3.1. Batasan Masalah	4
3.2. Manfaat penelitian	4
3.3. Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA.....	6
2.1. Citra Digital	6
2.2. Pemalsuan Citra Digital	7
2.3. <i>Error Level Analysis (ELA)</i>	7
2.4. <i>Deep Learning</i>	10
2.5. <i>Convolutional Neural Network (CNN)</i>	13
2.6. <i>Confusion Matrix</i>	24
BAB III METODOLOGI PENELITIAN	27
3.1. Tahap Penelitian.....	27
3.2. Waktu dan Lokasi Penelitian.....	28
3.3. Instrumen Penelitian.....	28
3.4. Teknik Pengambilan Data.....	29
3.5. Perancangan Sistem	29
BAB IV HASIL DAN PEMBAHASAN.....	45
4.1. Hasil Penelitian.....	45
4.2. Pembahasan	60
BAB V PENUTUP	62

5.1	Kesimpulan	62
5.2	Saran.....	63
	DAFTAR PUSTAKA	64
	LAMPIRAN	66

DAFTAR GAMBAR

Gambar 2.1	Algoritma perkiraan kualitas JPEG.....	7
Gambar 2.2	Ilustrasi nilai kualitas algoritma ELA.....	8
Gambar 2.3	Deep Neural Network.....	11
Gambar 2.4	Ilustrasi arsitektur Convolutional Neural Network (CNN).....	12
Gambar 2.5	Ilustrasi proses convolutional layer.....	13
Gambar 2.6	Ilustrasi weights dan bias pada layer.....	14
Gambar 2.7	Grafik fungsi Aktivasi ReLU.....	16
Gambar 2.8	Ilustrasi fungsi aktivasi softmax.....	17
Gambar 2.9	Operasi max pooling.....	18
Gambar 2.10	Confusion Matrix menampilkan total positive dan negative tuple....	20
Gambar 3.1	Tahapan Penelitian.....	22
Gambar 3.2	Flowchart tahapan testing dan training.....	25
Gambar 3.3	Contoh Data Training.....	26
Gambar 3.4	Hasil ELA dari gambar 3.3.....	27
Gambar 3.5	Identifikasi nilai RGB dari beberapa pixel.....	28

Gambar 3.6	Source Code proses ELA.....	30
Gambar 3.7	Arsitetur pembangunan model CNN.....	31
Gambar 3.8	Parameter Training.....	33
Gambar 3.9	Layer Arsitektur CNN.....	34
Gambar 3.10	Proses training data.....	35
Gambar 3.11	Source code testing data.....	35
Gambar 4.1	Identifikasi nilai RGB dari beberapa pixel.....	37
Gambar 4.2	Kurva akurasi dan kurva loss Training Accuracy.....	40
Gambar 4.3	Confusion Matrix.....	41

DAFTAR TABEL

Tabel 3.1 Tabel konversi RGB ke YCbCr.....	28
Tabel 3.2 Ilustrasi nilai kualitas JPG dengan kernel 8 x 8.....	29
Tabel 3.3 Confusion matrix model evaluation confusion matrix.....	36
Tabel 4.1 Tabel konversi RGB ke YCbCr.....	38
Tabel 4.2 Tabel Koordinat pixel dengan kernel 8 x 8.....	39
Tabel 4.3 Uji deteksi citra asli.....	41
Tabel 4.4 Uji deteksi citra Splice.....	44
Tabel 4.5 Uji deteksi citra Copy Move.....	46

BAB I

PENDAHULUAN

7.1. Latar Belakang

Menurut *EU High Level Expert Group (2018)*, berita palsu didefinisikan sebagai disinformasi, yaitu segala bentuk informasi yang tidak akurat, salah, ataupun menyesatkan yang dipresentasikan, dipromosikan, atau didesain. Di balik berita-berita palsu, terdapat beberapa alasan mengenai publikasi tersebut. Salah satunya adalah untuk mendapatkan keuntungan secara ekonomi, entah itu melalui peningkatan jumlah klik berita maupun membuat berita yang tidak seharusnya untuk menguntungkan salah satu pihak (Özgöbek and Gulla, 2018).

Pemalsuan dan perusakan informasi tumbuh secara eksponensial, karena semakin banyaknya platform media sosial. Informasi yang dipalsukan disebarluaskan melalui berbagai platform dan dalam berbagai bentuk. Platform media sosial merupakan lokasi yang sangat ideal untuk melakukan tindakan pemalsuan dan perusakan tersebut karena mengingat informasinya bisa menjangkau sejumlah besar audiens (Parikh et al., 2019).

Saat ini penyebaran informasi atau berita melalui media online tidak hanya dilakukan oleh situs berita yang sudah dikenal oleh masyarakat, namun oleh siapa saja pengguna internet dapat berperan dalam penyebaran suatu informasi. Sayangnya banyak informasi atau berita yang disebarluaskan secara individu atau

berkelompok lebih banyak yang tidak dapat dipertanggungjawabkan kebenarannya atau teindikasi *hoax*. *Hoax* merupakan informasi atau berita yang berisi hal-hal yang belum pasti atau yang benar-benar bukan merupakan fakta yang terjadi.

Kini informasi atau berita yang dianggap benar tidak lagi mudah ditemukan. Survey Mastel (2017) mengungkapkan bahwa dari 1.146 responden, 44,3% diantaranya menerima berita hoax setiap hari dan 17,2% menerima lebih dari satu kali dalam sehari. Bahkan media arus utama yang diandalkan sebagai media yang dapat dipercaya terkadang ikut terkontaminasi penyebaran hoax. Media arus utama juga menjadi saluran penyebaran informasi/berita hoax, masing-masing sebesar 1,20% (radio), 5% (media cetak) dan 8,70% (televisi). Tidak saja oleh media arus utama, kini hoax sangat banyak beredar di masyarakat melalui media online. Hasil penelitian yang dilakukan oleh Mastel (2017) menyebutkan bahwa saluran yang banyak digunakan dalam penyebaran hoax adalah situs web, sebesar 34,90%, aplikasi chatting (Whatsapp, Line, Telegram) sebesar 62,80%, dan melalui media sosial (Facebook, Twitter, Instagram, dan Path) yang merupakan media terbanyak digunakan yaitu mencapai 92,40%. Sementara itu, data yang dipaparkan oleh Kementerian Komunikasi dan Informatika menyebut ada sebanyak 800 ribu situs di Indonesia yang terindikasi sebagai penyebar hoax dan ujaran kebencian (Juditha, 2018).

Pemalsuan citra digital telah menjadi salah satu taktik penyebar hoax yang merupakan ajang penyebar provokasi, menimbulkan kerusuhan serta kebencian. Untuk mengidentifikasi suatu citra asli atau palsu, sangat sulit dilihat dengan mata telanjang, diperlukan teknik-teknik khusus dan ketelitian tertentu agar dapat mengetahui dengan pasti suatu citra merupakan citra asli atau sudah mengalami modifikasi. Bagi orang awam, hal ini mungkin sulit untuk dilakukan. Mesin dapat melakukannya dengan mudah untuk sejumlah besar data dengan bantuan teknik kecerdasan buatan yang digunakan untuk mengembangkan sistem yang akan membantu dalam mengenali apakah citra tersebut asli atau palsu. Teknologi ini membutuhkan banyak data citra, dan setiap citra memiliki banyak pixel-pixel penyusunnya. *Convolutional Neural Network* (CNN) adalah salah satu algoritma dari *Deep Learning* yang merupakan pengembangan dari *Multi Layer Perceptron* (MLP) yang dirancang untuk mengolah data citra. CNN bisa digunakan untuk mendeteksi dan mengenali object pada sebuah *image*. Oleh karena itu pada penelitian ini membahas tentang implementasi metode *deep learning* yang dilakukan pada sekumpulan data untuk mengetahui model yang paling cocok untuk pengembangan sistem pengklasifikasi citra asli dan palsu.

7.2. Rumusan Masalah

- a. Bagaimana Sistem dapat mendeteksi pemalsuan gambar.
- b. Bagaimana kinerja dari metode *Error Level Analysis* untuk mendeteksi adanya indikasi pemalsuan (*splice* atau *copy-move*) dalam gambar.

- c. Bagaimana hasil testing model dari system klasifikasi gambar asli dan palsu (*splice* atau *copy-move*) menggunakan metode *Deep Learning*.

3.1. Tujuan Penelitian

- a. Membuat system yang dapat mendeteksi gambar yang telah dimanipulasi.
- b. Mengetahui hasil kinerja dari *Error Level Analysis* dalam mendeteksi adanya indikasi pemalsuan *splice* atau *copy-move* dalam gambar.
- c. Mengetahui hasil testing model dari klasifikasi gambar asli dan palsu (*splice* atau *copy-move*) menggunakan metode *Deep Learning*.

3.1. Batasan Masalah

Terdapat beberapa batasan yang berlaku pada penambangan sistem ini, yaitu :

- a. Data mentah harus berupa citra dengan lossy compression (contohnya .jpg).
- b. Dambar bukan merupakan computer generated image (CGI).

3.2. Manfaat penelitian

- a. Meningkatkan kenyamanan dalam mendapatkan informasi yang sesuai dengan fakta.
- b. Masyarakat mendapatkan pertimbangan dalam menentukan apakah gambar merupakan asli atau palsu (*splice* atau *copy-move*).

3.3. Sistematika Penulisan

Untuk memberikan gambaran singkat mengenai isi tulisan secara keseluruhan, maka akan diuraikan beberapa tahapan dari penulisan secara sistematis, yaitu :

BAB I PENDAHULUAN

Bab ini menguraikan secara umum mengenai hal yang menyangkut latar belakang, perumusan masalah dan batasan masalah, tujuan, manfaat dan sistematika penulis.

BAB II TINJAUAN PUSTAKA

Bab ini berisi teori-teori terkait hal-hal yang mendasari dan yang berhubungan dengan Pengolahan Citra, Pemrosesan Citra dan metode-metode yang digunakan pada penelitian ini.

BAB III METODOLOGI PENELITIAN

Bab ini memaparkan tahapan penelitian, waktu dan lokasi penelitian, instrumen penelitian, teknik pengambilan data dan rancangan sistem serta penerapan metode *Deep Learning* dan juga teknik pengolahan data.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil penelitian dan pembahasan terkait pengolahan data yang telah dilakukan yang disertai dengan hasil penelitian.

BAB V PENUTUP

Bab ini berisi kesimpulan yang diperolah dari hasil penelitian yang telah dilakukan serta saran-saran untuk pengembangan sistem lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1. Citra Digital

Citra secara umum adalah merupakan suatu gambar, foto ataupun berbagai tampilan dua dimensi yang menggambarkan suatu visualisasi objek. Citra dapat diwujudkan dalam bentuk tercetak ataupun digital. Citra digital adalah larik angka-angka secara dua dimensional. Citra digital tersimpan dalam suatu bentuk larik (array) angka digital yang merupakan hasil kuantifikasi dari tingkat kecerahan masing-masing piksel penyusun citra tersebut.

Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (continue) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, scanner dan lain sebagainya sehingga bayangan objek yang disebut citra tersebut terekam (Prabowo and Abdullah, 2018).

Dalam komputer, setiap piksel diwakili oleh dua buah bilangan bulat (integer) untuk menunjukkan lokasinya dalam bidang citra dan sebuah nilai dalam bilangan bulat (integer) untuk menunjukkan cahaya atau keadaan terang-gelap piksel tersebut. Suatu piksel dapat ditunjukkan lokasinya dengan menggunakan koordinat (0,0) yang digunakan untuk posisi kiri atas dalam bidang citra. Tingkat pencahayaan suatu piksel dapat ditunjukkan dengan bilangan bulat yang besarnya 8 bit, dengan lebar selang nilai 0-255, di mana 0 untuk warna hitam, 255 untuk

warna putih. Citra digital yang diperoleh dari kamera digital akan menghasilkan intensitas pantulan yang menggambarkan terang dan gelap pada penampilan piksel-piksel penyusunannya selain itu juga akan memberikan informasi warna (Pradhitya, 2015).

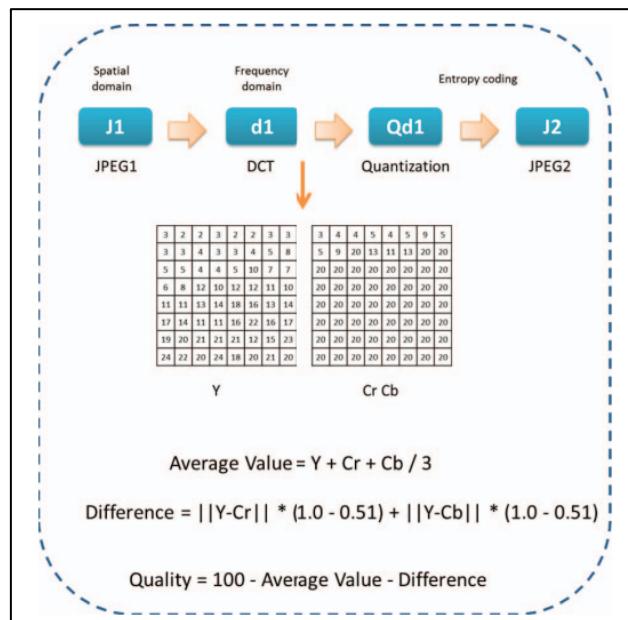
2.2. Pemalsuan Citra Digital

Pemalsuan citra sering tak dapat dikenali secara kasat mata karena citra hasil modifikasi sulit ditentukan keasliannya. Kasus modifikasi citra yang terjadi misalnya adalah kasus beredarnya foto-foto yang melibatkan pejabat, artis atau publik figur yang memuat konten-konten yang sensitif sehingga menimbulkan kehebohan di masyarakat. Kemunculan citra-citra digital yang dimodifikasi akan menimbulkan permasalahan-permasalahan dalam kehidupan sosial seperti penyebaran informasi-informasi yang tidak benar, sehingga sangat mudah terjadi kesalah pahaman. JPEG “Joint Photographic Experts Group” merupakan salah satu format citra yang paling sering digunakan, dimana JPEG adalah sebuah format citra yang memiliki format lossy (dimana metode untuk mengkompresi data dengan hasil perbedaan data yang tidak terlalu jauh dari data sebelum kompresi), tetapi banyaknya error yang tersimpan tidak sama dengan citra sebelum proses kompresi (Darmawan et al., 2019).

2.3. *Error Level Analysis (ELA)*

ELA adalah algoritma kompresi JPEG untuk deteksi forensik gambar. Seperti dibahas pada bagian III, koefisien frekuensi dari gambar dikuantisasi oleh tabel

kuantisasi dan diikuti oleh proses entropi encoded. Krawetz bersama timnya (*Hacker Factor Solution*) mengadopsi proses kuantisasi untuk mengembangkan algoritma untuk mendekati kualitas JPEG. Gambar. 2 menyajikan algoritma pendekatan kualitas JPEG. Melalui persamaan tersebut, perhitungan kualitas JPEG dapat diringkas dengan menghitung selisih antara nilai rata-rata dari tabel kuantisasi Y (*luminance*) dan CrCb (*chrominance*).



Gambar 2.1 Algoritma perkiraan kualitas JPEG

(Warif et al., 2015)

Menggunakan ELA, gambar selanjutnya dibagi menjadi blok 8 x 8 dan dikompresi ulang secara independen pada tingkat kesalahan yang diketahui seperti 95%. Setiap kotak harus memberikan tingkat kualitas yang kira-kira sama jika gambar tidak dimodifikasi sama sekali. Tingkat kesalahan adalah hilangnya informasi selama gambar disimpan dalam format JPEG. Ilustrasi nilai kualitas algoritma ELA

ditunjukkan pada Gambar 3. Pada gambar tersebut, perbedaan tingkat kesalahan pada blok-blok tertentu dapat menentukan area yang dimodifikasi.

Selain itu, tingkat kesalahan akan meningkat pada operasi penyimpanan ulang. Operasi resave resave selanjutnya dapat mengurangi potensi tingkat kesalahan dan ditunjukkan melalui hasil ELA yang lebih gelap. Setelah sejumlah operasi penyimpanan ulang, kisi persegi dapat mencapai tingkat kesalahan minimumnya. oleh karena itu frekuensi dan detail dapat hilang oleh setiap operasi penyimpanan ulang.

81	81	81	81	81	81	81	81	81
81	81	81	81	81	81	81	81	81
90	90	90	81	81	81	81	81	81
90	90	90	81	81	81	81	81	81
90	90	90	81	81	81	81	81	81
90	90	90	81	81	81	81	81	81
81	81	81	81	81	81	81	81	81
81	81	81	81	81	81	81	81	81

Gambar 2.2 Ilustrasi nilai kualitas algoritma ELA

(Warif et al., 2015)

Namun, jumlah kesalahan dalam ELA terbatas pada blok 8 x 8. Kesalahan blok telah mencapai minimum lokalnya jika tidak ada perubahan pada tingkat kualitas. Piksel tidak pada minimum lokalnya jika ada sejumlah besar biaya yang kemudian dapat mengidentifikasi perubahan. Misalnya, menyimpan 90% lagi akan menghasilkan gambar yang hampir sama untuk penghematan satu kali sebesar

81%. Selain itu, area biru atau merah yang dirancang oleh ELA mencerminkan perubahan yang dilakukan oleh perangkat lunak komersial seperti photoshop dan GIMP. Kesalahan yang dihasilkan oleh teknik ELA dapat membantu mengidentifikasi manipulasi pada gambar JPEG. Bagaimanapun, teknik ELA dapat diringkas sebagai proses evaluasi tingkat kualitas kisi-kisi yang dikuadratkan di dalam gambar.

Berdasarkan karakteristik ELA, teknik ini diyakini dapat berguna untuk citra forensik area. Untuk selanjutnya, bagian berikut menjelaskan hasil eksperimen dan pembahasannya (Warif et al., 2015).

2.4. Deep Learning

Belakangan ini *Deep Learning* menjadi sorotan dalam pengembangan Machine Learning. Alasannya karena *Deep Learning* telah mencapai hasil yang luar biasa dalam visi komputer. *Deep Learning* merupakan cabang dari *Machine Learning* yang terinspirasi dari *kortek* manusia dengan menerapkan jaringan syaraf buatan yang memiliki banyak hidden layer. *Convolutional Neural Network(CNN)* merupakan salah satu metode dalam *Deep Learning* yang dibuat untuk menutupi kelemahan dari metode sebelumnya. Terdapat beberapa kelemahan dalam metode sebelumnya, tetapi dengan model ini sejumlah parameter bebas dapat dikurangi dan deformasi gambar input seperti translasi, rotasi dan skala dapat ditangani (Santoso and Ariyanto, 2018).

Algortima pada *Deep Learning* memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi secara otomatis. Hal ini berarti algoritma yang dimilikinya

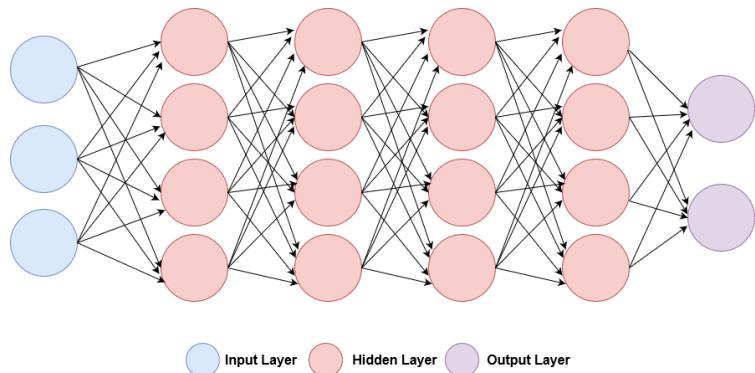
secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Algortima ini dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (*supervised*), tanpa pengawasan (*unsupervised*), dan semi terawasi (*semi supervised*). Teknik dan algortima *deep learning* telah digunakan dalam beberapa bidang seperti klasifikasi gambar, klasifikasi video, deteksi objek, pengenalan pola, *text-to-speech*, *natural language processing*, robotika, klasifikasi teks, dan lain sebagainya.

Jaringan saraf yang dimiliki oleh *Deep Learning* terbentuk dari hirarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (*multi layer*). Berdasarkan hal itulah *Deep Learning* dapat digunakan untuk memecahkan masalah kompleks yang lebih rumit dan terdiri dari sejumlah besar lapisan transformasi non-linier. *Deep Learning* bekerja berdasarkan jaringan dan prosedural optimal yang digunakan pada arsitektur. Setiap *output* dari lapisan per lapisan yang tersembunyi dapat dipantau dengan menggunakan grafik khusus yang dirancang untuk setiap *output* neuron. Kombinasi dan rekombinasi dari setiap neuron yang saling terhubung dari semua unit lapisan tersembunyi dilakukan menggunakan gabungan dari fungsi aktivasi. Prosedur-prosedur tersebut dikenal sebagai Transformasi Non-Linier yang digunakan untuk prosedur optimal untuk menghasilkan bobot optimal pada setiap unit lapisan guna mendapatkan

nilai target yang dibutuhkan.

Dalam proses perancangan, apabila jumlah saraf yang ditambahkan sangat banyak, hal tersebut tidak akan pernah cocok untuk menyelesaikan setiap masalah. Persoalan terpenting dalam *Deep Learning* adalah jaringan sarafnya dilatih dengan cara penurunan gradien secara sederhana. Pada saat kita menambahkan lapisan jaringan yang semakin banyak, maka sebaliknya penurunan dari gradien semakin berkurang sehingga dapat mempengaruhi nilai outpunya. Jaringan Saraf Tiruan (*Artificial Neural Network*) adalah jaringan saraf yang biasanya menggunakan jaringan seperti umpan maju (*feed forward*) atau *recurrent network* yang hanya memiliki 1 atau 2 lapisan tersembunyi. Tetapi, jika lapisan jaringan sarafnya lebih dari 2 *layer* atau bahkan mencapai ratusan lapisan itu lah yang disebut sebagai *Deep Learning*. Pada Jaringan Saraf Tiruan, arsitektur jaringan yang dimilikinya kurang kompleks dan membutuhkan lebih banyak informasi tentang data input sehingga dapat menentukan algoritma mana yang dapat digunakan.

Dalam Jaringan Saraf Tiruan terdiri dari beberapa algoritma yaitu *Model Hebb*, *Perceptron*, *Adaline*, *Forward Propagation*, dll. Sedangkan pada algoritma jaringan saraf *Deep Learning* tidak memerlukan informasi apapun terhadap data yang akan dipelajarinya, dan algoritmanya dapat secara mandiri melakukan tuning (penyetelan) dan pemilihan model yang paling optimal (Dadang, 2018).



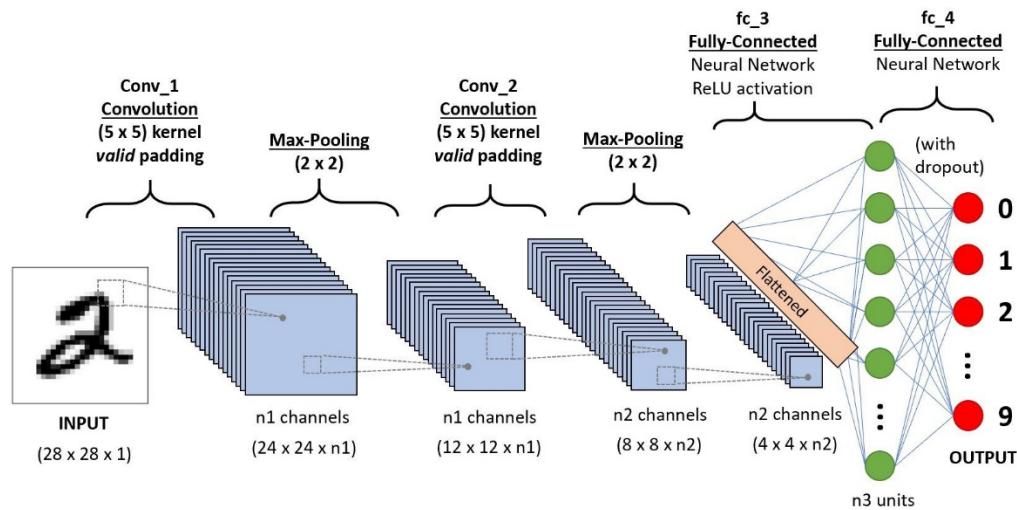
Gambar 2.3. Deep Neural Network

2.5. Convolutional Neural Network (CNN)

CNN adalah tipe network yang berbasis feedforward, yang alur informasinya hanyalah satu arah, yaitu dari masukan ke keluaran. Walaupun ada beberapa jenis arsitektur CNN, pada umumnya, CNN memiliki beberapa convolutional layer dan pooling layer. Kemudian, diikuti oleh satu atau lebih fully connected layer. Pada klasifikasi gambar, masukan pada CNN adalah dalam bentuk gambar, sehingga setiap pixel-nya dapat diolah.

Secara singkat, convolutional layer digunakan sebagai pengekstrasi fitur yang mempelajari representasi fitur tersebut dari gambar yang menjadi masukan pada CNN. Sedangkan, pooling layer bertugas untuk mengurangi resolusi spasial dari peta-peta fitur. Umumnya, sebelum fully connected layer, terdapat tumpukan beberapa convolutional dan pooling layer yang berfungsi untuk mengekstrak representasi fitur yang lebih abstrak. Setelahnya, fully connected layer akan menginterpretasi fitur-fitur tersebut dan melakukan fungsi-fungsi yang

membutuhkan high-level reasoning. Klasifikasi pada akhir CNN akan menggunakan fungsi softmax (Gunawan et al., 2018).



Gambar 2.4. Ilustrasi arsitektur *Convolutional Neural Network* (CNN)

(Gunawan et al., 2018).

Tahap pertama dalam arsitektur CNN adalah tahap konvolusi. Kemudian dilanjutkan fungsi aktivasi menggunakan fungsi aktivasi ReLu (*Rectifier Linear Unit*), kemudian dilanjutkan dengan proses *pooling*. Proses ini diulang terus menerus sampai didapatkan peta fitur yang cukup untuk dilanjutkan ke *fully connected neural network*, sehingga dapat dihasilkan output class. Penjelasan secara detail dari tahap arsitektur CNN tersebut dijelaskan sebagai berikut:

2.5.1. *Convolutional Layer*

Convolution layer merupakan proses utama yang mendasari jaringan arsitektur CNN dan terdiri atas kernel. Kernel-kernel pada lapisan ini disebut

filter konvolusi. Kernel berfungsi mempelajari fitur-fitur lokal pada *feature map*. Tahap *convolutional layer* melakukan operasi konvolusi pada output dari layer sebelumnya. Konvolusi adalah istilah matematis dimana pengaplikasian sebuah fungsi pada output fungsi lain secara berulang. Persamaan konvolusi merupakan persamaan pada dua fungsi argument bernilai riil. Operasi konvolusi $s(t)$ dapat ditunjukkan pada persamaan berikut:

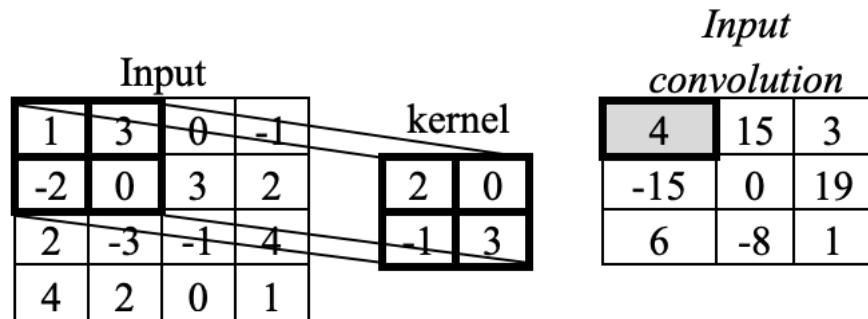
$$s(t) = \sum_a I(a).K(t - a) \quad (2.1)$$

Dimana $I(a)$ adalah input dan $K(a)$ adalah kernel. *Input convolution layer* merupakan gambar yang direpresentasikan menjadi sebuah matriks. Operasi konvolusi menghasilkan nilai tinggi dan rendah pada posisi tertentu pada *feature map*. Posisi tertentu dari konvolusi kernel, merupakan perkalian untuk setiap nilai pada sel kernel dan nilai piksel gambar yang tumpang tindih dengan sel kernel (Khan et al., 2018).

Adapun operasi perkaliannya menggunakan persamaan sebagai berikut:

$$h_{i,j} = \sum_{k=1}^m \sum_{l=1}^m W_{k,l} X_{i+k-1, j+l-1} \quad (2.2)$$

Dimana m adalah lebar dan tinggi kernel, h adalah input convolution, x adalah input, dan w adalah convolutional kernel. Sebagai contoh, ilustrasi pada proses convolution layer dengan nilai stride 2 dapat dilihat pada gambar 2.5.



Gambar 2.5. Ilustrasi proses *convolutional layer* (Khan et al., 2018)

Adapun perhitungan nilai pada input convolution yang diperoleh dengan menggunakan persamaan 2.4 adalah sebagai berikut:

$$h_{1,1} = (1 \cdot 2) + (3 \cdot 0) + (-2 \cdot -1) + (0 \cdot 3) = 4$$

$$h_{1,2} = (3 \cdot 2) + (0 \cdot 0) + (0 \cdot -1) + (3 \cdot 3) = 15$$

$$h_{1,3} = (0 \cdot 2) + (-1 \cdot 0) + (3 \cdot -1) + (2 \cdot 3) = 3$$

$$h_{2,1} = (-2 \cdot 2) + (0 \cdot 0) + (2 \cdot -1) + (-3 \cdot 3) = -15$$

$$h_{2,2} = (0 \cdot 2) + (3 \cdot 0) + (-3 \cdot -1) + (-1 \cdot 3) = 0$$

$$h_{2,3} = (3 \cdot 2) + (2 \cdot 0) + (-1 \cdot -1) + (4 \cdot 3) = 19$$

$$h_{3,1} = (2 \cdot 2) + (-3 \cdot 0) + (4 \cdot -1) + (2 \cdot 3) = 6$$

$$h_{3,2} = (-3 \cdot 2) + (-1 \cdot 0) + (2 \cdot -1) + (0 \cdot 3) = -8$$

$$h_{3,3} = (-1 \cdot 2) + (4 \cdot 0) + (0 \cdot -1) + (1 \cdot 3) = 1$$

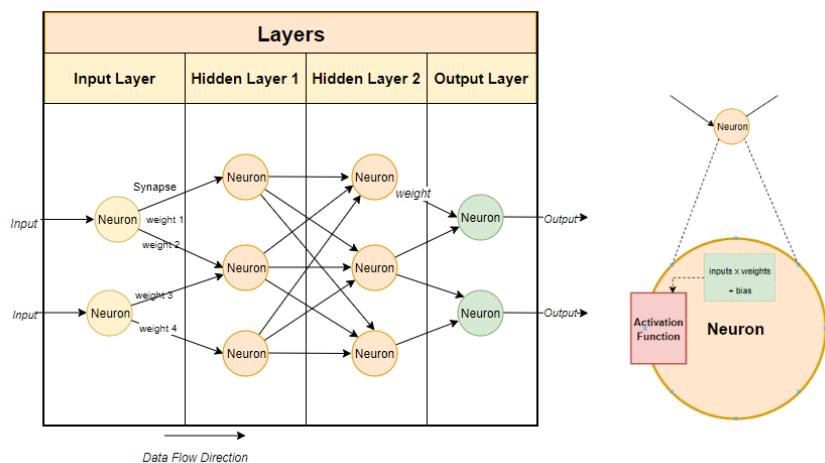
Dari perhitungan diatas, di peroleh matriks input convolution sebagai berikut :

$$\begin{matrix} 4 & 15 & 3 \\ -15 & 0 & 19 \\ 6 & -8 & 1 \end{matrix}$$

(Khan et al., 2018).

2.5.2. *Weight* dan *Bias*

Ketika *input* ditransmisikan antar *neuron*, *weights* diterapkan dan diteruskan ke fungsi aktivasi bersama dengan bias. Untuk lebih jelasnya perhatikan ilustrasi *weights* dan *bias* pada gambar 2.6.



Gambar 2.6. Ilustrasi *weights* dan *bias* pada *layer* (Malik, 2019)

Weights mengontrol sinyal (atau kekuatan koneksi) antara dua neuron. Dengan kata lain, bobot menentukan seberapa besar pengaruh input terhadap output. Saat jaringan neural dilatih pada set pelatihan, jaringan akan diinisialisasi dengan satu set *weights*. *Weights* kemudian dioptimasi selama periode pelatihan untuk menghasilkan *weights* yang optimal.

Bias adalah nilai konstanta yang ditambahkan ke layer *input* dengan *weights*. *Bias* digunakan untuk mengimbangi hasil dan untuk menggeser hasil fungsi aktivasi ke nilai positif atau negatif.

Dalam proses perhitungannya, neuron akan menghitung *weighted sum* dari *input* dengan persamaan:

$$Y = \sum(\text{weight} * \text{input}) + \text{bias} \quad (2.3)$$

Dimana *input* merupakan :

$$x_1, x_2, \dots, x_n \quad (2.4)$$

Dan *weight* adalah :

$$w_1, w_2, \dots, w_n \quad (2.7)$$

Maka *weighted sum* dapat dihitung dengan persamaan :

$$x_1w_1 + x_2w_2 + \dots + x_nw_n + \text{bias} \quad (2.8)$$

2.5.3. Fungsi Aktivasi

Fungsi aktivasi merupakan operasi matematik yang dikenakan pada sinyal output *y*. Fungsi aktivasi berfungsi menentukan apakah suatu *neuron* aktif atau tidak berdasarkan *weighter sum* dari suatu *input*. Beberapa jenis fungsi aktivasi yang sering digunakan pada *deep learning* adalah *sigmoid*, *Tanh*, *algebraic sigmoid*, *ReLU*, *noisy ReLU*, *Leaky ReLU/PReLU*, *Randomized Leaky ReLU*, dan *Eksponential Linear Unit*. Penelitian ini

menggunakan dua fungsi aktivasi yaitu fungsi aktivasi ReLU dan fungsi aktivasi *softmax*. Penjelasan dari kedua fungsi tersebut adalah sebagai berikut:

1. ReLU

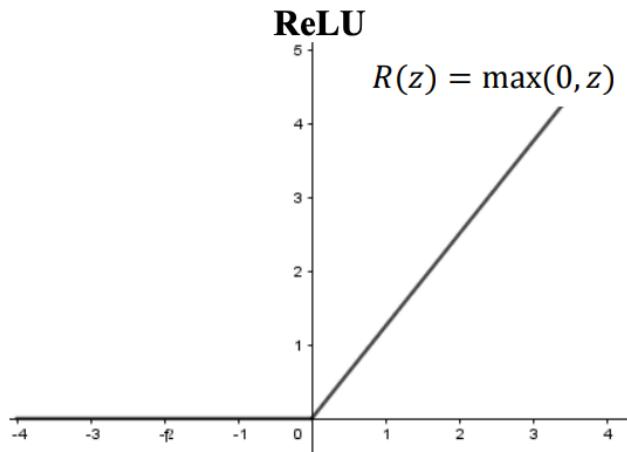
Fungsi aktivasi *Rectifier Linear Unit*(ReLU) merupakan fungsi aktivasi sederhana yang memiliki kepentingan praktis khusus karena perhitungannya yang cepat. Kelebihan fungsi aktivasi ReLU dibandingkan dengan fungsi aktivasi lain adalah sebagai berikut.

- Fungsi aktivasi ReLU merupakan fungsi aktivasi default ketika mengembangkan *multilayer perceptron* dan *convolutional neural network*.
- Fungsi aktivasi ReLU mengatasi masalah *gradient descent* yang hilang, yang memungkinkan model belajar lebih cepat dan berkinerja lebih baik.
- Menemukan cara melatih jaringan dengan lebih cepat, sehingga mengurangi kemungkinan terjadinya *overfitting*.

Fungsi aktivasi ReLU memetakan input ke 0 jika negatif dan mempertahankan nilainya jika positif. Representasi fungsi ReLU adalah sebagai berikut:

$$- f_{ReLU}(x) = \max(0, x) \quad (2.9)$$

Adapun grafiknya dapat dilihat pada gambar 2.7.



Gambar 2.7 Grafik Fungsi Aktivasi ReLU (Khan et al., 2018).

2. *Softmax*

Fungsi aktivasi *softmax* merupakan fungsi input vektor dari bilangan real K , yang kemudian dinormalkan menjadi distribusi probabilitas yang terdiri atas probabilitas K yang proposional ke eksponensial input. Komponen vektor pada *softmax* memiliki interval $(0,1)$. Fungsi softmax merupakan lapisan yang menghubungkan antara *fully connected layer* dengan *dense connection*. *Softmax* berfungsi untuk menghitung probabilitas pada setiap kelas target yang memungkinkan dan akan membantu menentukan kelas target pada input yang diberikan. Nilai *softmax* berada pada rentang probabilitas output dari 0 hingga 1 dan jumlah semua probabilitas sama dengan satu.

Definisi fungsi *softmax* $\sigma = \mathbb{R}^K \rightarrow \mathbb{R}^K$ dituliskan sebagai persamaan berikut:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.9)$$

dimana $i = 1, \dots, K$ dan $z = (z_1, \dots, z_K) \in \mathbb{R}^K$. Adapun contoh ilustrasi fungsi aktivasi *softmax* dapat dilihat pada gambar 2.11 sebagai berikut.

$$\begin{array}{ccc} \text{Logits score} & \text{Softmax} & \text{Probabilitas} \\ y = \begin{cases} 2.0 \\ 1.0 \\ 0.1 \end{cases} \longrightarrow \sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} & \longrightarrow & \begin{array}{l} P=0.7 \\ P=0.2 \\ P=0.1 \end{array} \end{array}$$

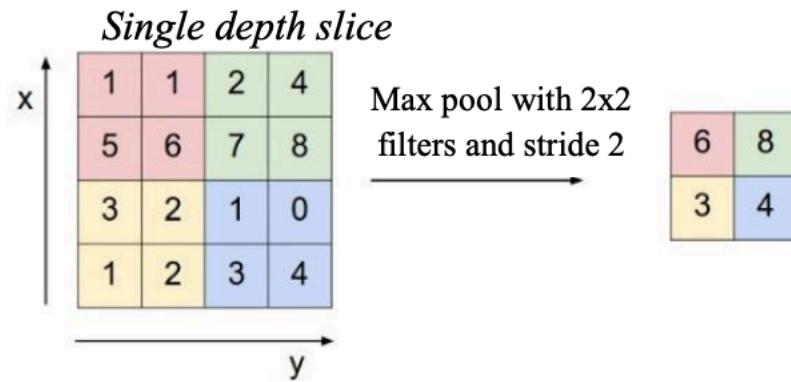
Gambar 2.8 ilustrasi fungsi aktivasi *softmax*

Logits score menunjukkan lapisan *neuron* terakhir sebagai *output* mentah pada lapisan terakhir neural network sebelum proses aktivasi berlangsung. Setelah *output* diproses dengan *softmax* akan menghasilkan nilai probabilitas dengan jumlah 1(Khan et al., 2018).

3. *Pooling layer*

Pooling merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling*. *Pooling Layer* biasanya berada setelah *conv layer*. Pada dasarnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan secarabergantian bergeser pada seluruh area *feature map*. Dalam *pooling layer*

terdapat dua macam *pooling* yang biasa digunakan yaitu *average pooling* dan *max-pooling*. Nilai yang diambil pada *average pooling* adalah nilai rata-rata, sedangkan pada *max-pooling* adalah nilai maksimal. Lapisan *pooling* yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam arsitektur CNN dapat secara progresif mengurangi ukuran volume *output* pada *feature*



map, sehingga mengurangi jumlah parameter pada jaringan untuk mengendalikan *overfitting*. Lapisan *pooling* bekerja pada setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya dengan menggunakan filter dengan ukuran 2x2 yang diaplikasikan dengan langkah (*stride*) sebanyak dua dan beroperasi pada setiap irisan dari inputnya. Berikut ini adalah contoh gambar operasi *max-pooling*:

Gambar 2.9 Operasi *max pooling* (Khan et al., 2018)

Output dari proses *pooling* adalah sebuah matriks dengan

dimensi yang lebih kecil dibandingkan dengan citra awal. Lapisan *pooling* akan beroperasi pada setiap irisan kedalam volume input secara bergantian. Operasi *max-pooling* pada gambar di atas menggunakan ukuran filter 2x2. Masukan pada proses tersebut berukuran 4x4. Dari masing-masing 4 angka pada input operasi tersebut diambil nilai maksimalnya kemudiandilanjutkan membuat ukuran *output* baru menjadi ukuran 2x2.

4. *Fully Connected Layer*

Fully connected layer merupakan sebuah lapisan dimana semua *neuron* aktivasi dari lapisan sebelumnya terhubung dengan neuron lapisan selanjutnya. Pada dasarnya lapisan ini biasanya digunakan pada MLP (*Multi Layer Perceptron*) yang mempunyai tujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Perbedaan *fully connected layer* dengan konvolusi biasa adalah *neuron* pada lapisan konvolusi terhubung hanya ke daerah tertentu sedangkan *fully connected* memiliki *neuron* yang semuanya terhubung (Khan et al., 2018). Namun keduanya masih menggunakan operasi dot.

5. *Crossy Entropy Loss Function*

Loss Function merupakan salah satu komponen penting dalam *neural network*. *Loss* menggambarkan kemungkinan kesalahan yang

dihasilkan oleh model. *Loss Function* yang baik adalah fungsi yang diharapkan menghasilkan *error* yang paling rendah. Ketika suatu model memiliki kelas yang cukup banyak, perlu adanya cara untuk mengukur perbedaan antara probabilitas hasil hipotesis dan probabilitas kebenaran yang asli. *Categorical Cross Entropy* merupakan salah satu pilihan terbaik untuk menghitung nilai *loss* pada permasalahan *multi-class classification*. *Categorical Cross Entropy* (CE) biasa juga disebut *Softmax Loss* yang merupakan gabungan dari *softmax activation* dan *cross-entropy loss*. Berikut merupakan rumus *Categorical Cross Entropy* (Gomez, 2018).

$$CE = - \sum_i^C t_i \log (s_i) \quad (2.10)$$

6. *Precision, Recall* dan *Akurasi*

Precision, *recall* dan *akurasi* digunakan untuk mengukur kinjra sistem, *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. *Recall* merupakan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. *Akurasi* merupakan tingkat kedekatan antara nilai prediksi dengan nilai aktual.

2.6. *Confusion Matrix*

Confusion matrix menurut Han dan Kamber (2011) dapat diartikan sebagai suatu alat yang memiliki fungsi untuk melakukan analisis apakah classifier tersebut baik dalam mengenali tuple dari kelas yang berbeda. Nilai dari *True-Positive* dan *True-Negative* memberikan informasi ketika classifier dalam melakukan klasifikasi data bernilai benar, sedangkan *False-Positive* dan *False-Negative* memberikan informasi ketika classifier salah dalam melakukan klasifikasi data.

		Predicted class		Total
Actual class	yes	no		
	yes	TP	FN	P
	no	FP	TN	N
Total	P'	N'		P + N

Gambar 2.10 Confusion Matrix menampilkan total positive dan negative tuple

(Fibrianda and Bhawiyuga, 2018)

TP (True Positive) → Jumlah data dengan nilai sebenarnya positif dan nilai prediksi positif

FP (False Positive) → Jumlah data dengan nilai sebenarnya negatif dan nilai prediksi positif

FN (False Negative) → Jumlah data dengan nilai sebenarnya positif dan nilai prediksi negatif

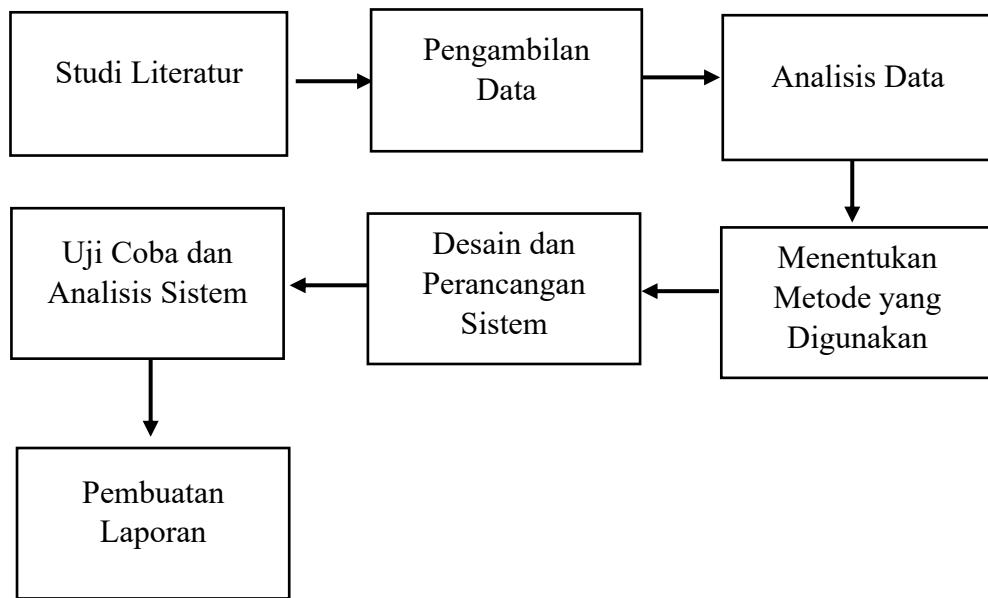
TN (True Negative) → Jumlah data dengan nilai sebenarnya negatif dan nilai prediksi negatif

BAB III

METODOLOGI PENELITIAN

3.1. Tahap Penelitian

Tahap pada penelitian ini sebagaimana diperlihatkan pada Gambar 3.1



Gambar 3.1. Tahapan Penelitian

Adapun uraian tahap penelitian pada **Gambar 3.1** sebagai berikut :

1. Studi Literatur merupakan tahap awal yang dilakukan untuk mencari referensi penelitian-penelitian terkait topik yang dilakukan yaitu metode untuk mengklasifikasi citra asli dan palsu.
2. Pengambilan data gambar yang terdiri dari dataset gambar asli dan palsu yaitu dataset CASIA 2.0 image. Data gambar yang diperoleh akan digunakan sebagai data latih dan data uji pada sistem yang akan dibuat.

3. Analisis data dilakukan untuk memilih data yang layak untuk dijadikan bahan penelitian.
4. Menentukan metode yang digunakan yaitu *Deep Learning* dengan algoritma *Convolutional Neural Network* (CNN).
5. Perancangan sistem dilakukan dengan pembuatan rancangan alur kerja sistem. Implementasi sistem dilakukan sesuai dengan rancangan alur kerja sistem yang telah dibuat. Sistem dibuat dengan menggunakan bahasa pemrograman Python dan algoritma *Convolutional Neural Network* (CNN).
6. Uji coba sistem dilakukan untuk mengetahui seberapa akurat sistem yang dibuat.
7. Tahap akhir dari penelitian ini adalah melakukan penulisan laporan penelitian dalam bentuk skripsi sebagai bahan publikasi.

3.2. Waktu dan Lokasi Penelitian

Waktu penelitian dimulai sejak disetujuinya proposal penelitian ini pada bulan Desember 2019 hingga proses penulisan pada bulan September 2021. Penelitian ini dilakukan di Laboratorium Kecerdasan Buatan Departemen Teknik Informatika Universitas Hasanuddin. Data citra diambil situs penyedia dataset yaitu kaggle.

3.3. Instrumen Penelitian

Instrumen yang digunakan pada penelitian ini adalah :

1. Perangkat Lunak (*Software*)

- Windows 10 pro
- Google Colab
- Jupyter notebook
- Python 3.8

2. Perangkat Keras (*Hardware*)

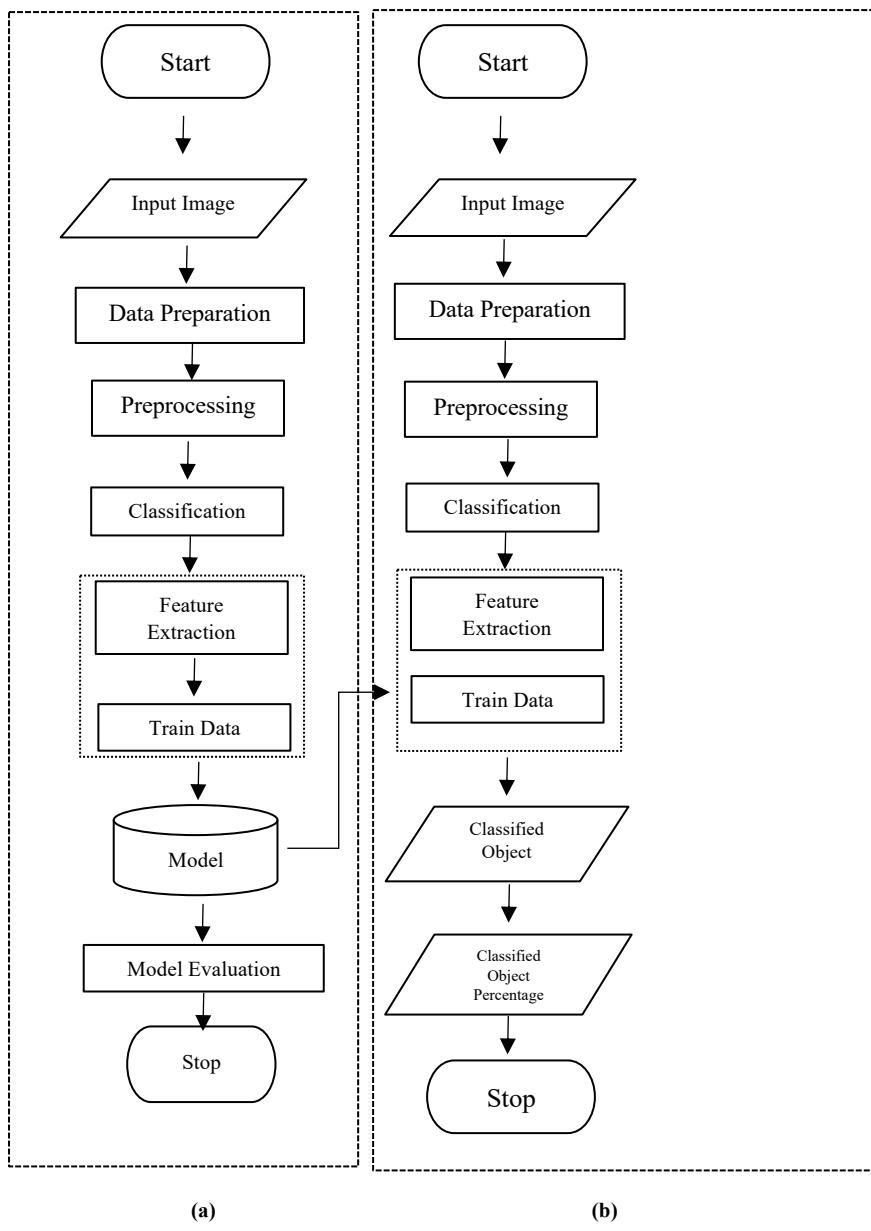
- Laptop Asus A456UR, CPU Intel Core i5 7200U 2.50 GHz, RAM 8 GB, GPU Nvidia GEFORCE 930MX, Windows 10 Pro.

3.4. Teknik Pengambilan Data

Data yang diambil berupa citra dengan resolusi 348 x 256 sebanyak 7492 gambar *Authentic* dan 5125 gambar *Tampered* yang nantinya akan dibagi menjadi data *training* dan *validation*. Data citra asli dan palsu diperoleh melalui sumber kaggle (<https://www.kaggle.com/divg07/casia-20-image-tampering-detection-dataset>) (<https://github.com/wenbihan/coverage>).

3.5. Perancangan Sistem

Dalam perancangan sistem ini terbagi menjadi tiga bagian, yaitu *training* dan *testing* seperti yang ditunjukkan pada gambar berikut :



Gambar 3.2 Flowchart tahapan traning dan testing.

Berdasarkan gambar 3.2 rancangan sistem dapat diuraikan sebagai berikut

3.5.1. Tahapan *Training*

Data yang digunakan pada penelitian ini berupa citra manusia, hewan dan benda. Data citra diperoleh dari (<https://www.kaggle.com/divg07/casia-20>-

[image-tampering-detection-dataset](#)(<https://github.com/wenbihan/coverage>)

sebanyak 4711 citra.

1. Input Data

Untuk mengawali proses *training*, *dataset* yang telah disiapkan dibagi menjadi 2 folder sesuai dengan label *Real*, *Splice*, dan *Copy Move* yang diberi pada masing-masing data yaitu sebanyak 1000 data *Real*, 1000 data *Splice*, dan 297 Data *Copy Move*. *Data training* yang akan digunakan sebanyak 80% dari jumlah data dan 20% untuk data validasi. *Data validation* digunakan untuk proses validasi akurasi pada model. Dalam tiap *epoch*, setelah selesai dilakukan *training*, dilanjutkan dengan proses validasi. Contoh data *training* dari setiap folder dapat dilihat pada gambar berikut :



(a) *Real*



(b) *Splice*

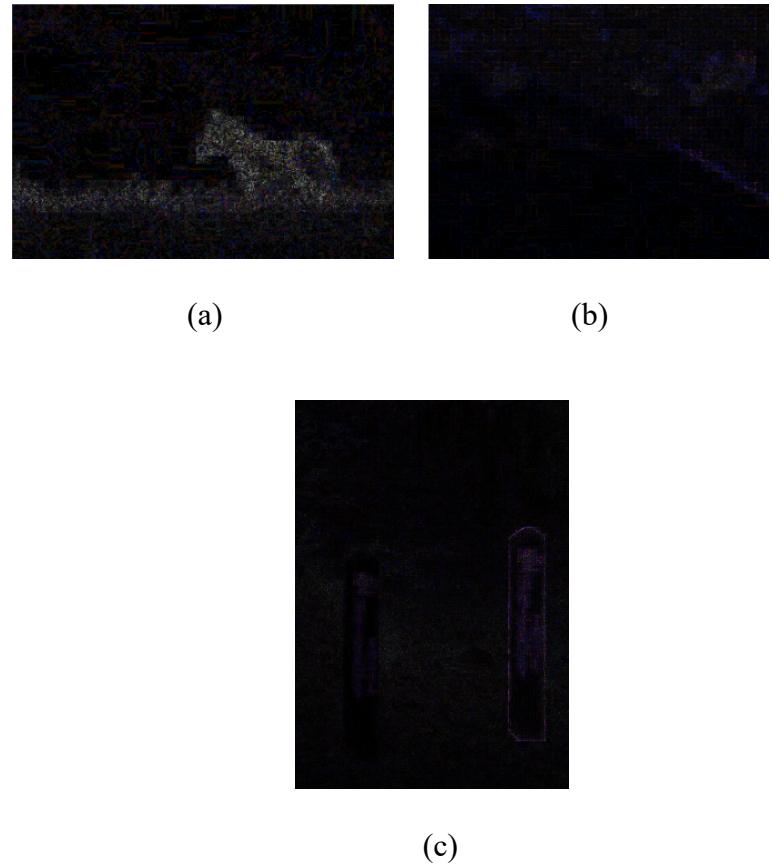


(b) *Copy Move*

Gambar 3.3 Contoh Data *Training*.

2. *Data Preparation*

Data Preparation dilakukan dengan konversi data mentah ke gambar ELA. Konversi data mentah ke gambar hasil ELA merupakan cara yang digunakan untuk menambah efisiensi pelatihan dari model CNN. Efisiensi ini dapat tercapai karena hasil gambar ELA mengandung informasi yang tidak berlebihan seperti gambar aslinya. Fitur yang dihasilkan oleh gambar ELA sudah difokuskan pada bagian gambar yang memiliki level *error* di atas batas. Selain itu, *pixel-pixel* pada gambar ELA cenderung memiliki warna yang mirip atau justru sangat kontras dengan *pixel-pixel* di dekatnya, sehingga pelatihan model CNN menjadi lebih efisien.



Gambar 3.4 Hasil ELA dari gambar 3.3

Proses yang dilakukan pada tahap ini adalah sebagai berikut:

Langkah 1. Menentukan citra RGB yang menjadi objek deteksi

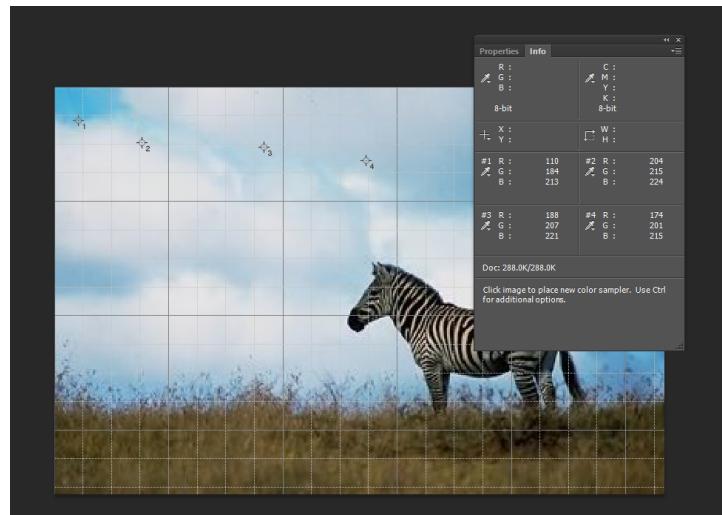
Langkah 2. Konversi nilai RGB ke YCbCr (*luminance* dan *chrominance*)

Adapun rumus konversi RGB ke YCbCr adalah sebagai berikut

:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (3.1)$$

Berikut merupakan ilustrasi konversi gambar RGB ke YCbCr



Gambar 3.5 Identifikasi nilai RGB dari beberapa pixel

dari gambar di atas diperoleh nilai RGB dari salah satu pixel nya kemudian dimasukkan ke dalam tabel di bawah untuk memperoleh nilai YCbCr :

Tabel 3.1. Tabel konversi RGB ke YCbCr

Y	R	Nilai	G	Nilai	B	Nilai	16
	0.257	110	0.504	184	0.098	213	
	28.27		92.736		20.874		
				157.88			
	R	Nilai	G	Nilai	B	Nilai	

Cb	-0.148	110	- 0.291	184	0.439	213	128	
	-16.28		-53.544		93.507			
151.683								
Cr	R	Nilai	G	Nilai	B	Nilai	128	
	0.439	110	- 0.368	184	-0.071	213		
	48.29		-67.712		-15.123			
	93.455							

Dari tabel di atas didapatkan nilai $Y = 157.88$, $Cb = 151.683$

dan $Cr = 93.455$

Langkah 3. Menghitung nilai *Error Level Analysis* pada grid (1,1) sampai dengan kordinat pixel (8,8) menggunakan rumus yang ada pada gambar 2.1 :

Tabel 3.2. Ilustrasi nilai kualitas JPG dengan kernel 8 x 8

81	81	81	81	81	81	81	81
81	81	81	81	81	81	81	81
90	90	90	81	81	81	81	81
90	90	90	81	81	81	81	81
90	90	90	81	81	81	81	81
90	90	90	81	81	81	81	81
81	81	81	81	81	81	81	81
81	81	81	81	81	81	81	81

$$\mu = (Y + Cb + Cr) / 3 \quad (3.2)$$

$$\Delta = \| Y-Cr \| * (1.0-0.51) + \| Y-Cb \| * (1.0-0.51) \quad (3.3)$$

$$Qn = 100 - \mu - \Delta \quad (3.4)$$

Nilai μ mewakili nilai rata-rata dari YCrCb , dan Δ mewakili nilai *difference* dari tabel quantisasi Y dengan CbCr serta Qn adalah kualitas dari gambar yang telah di konversi ke ELA. Perhitungan terus dilakukan sampai lebar pixel dalam citra kurang dari 8 pixel, sehingga tidak memungkinkan perhitungan pada grid 8x8. Setiap blok/pixel/persegi harus memberikan tingkat kualitas yang sama jika gambar tersebut bukan merupakan gambar yang telah dimanipulasi.

Berikut merupakan potongan *source code* untuk proses ELA :

```
def convert_to_ela_image(path, quality):
    temp_filename = 'temp_file_name.jpg'
    ela_filename = 'temp_ela.png'

    image = Image.open(path).convert('RGB')
    image.save(temp_filename, 'JPEG', quality = quality)
    temp_image = Image.open(temp_filename)

    ela_image = ImageChops.difference(image, temp_image)

    extrema = ela_image.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff

    ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)

    return ela_image
```

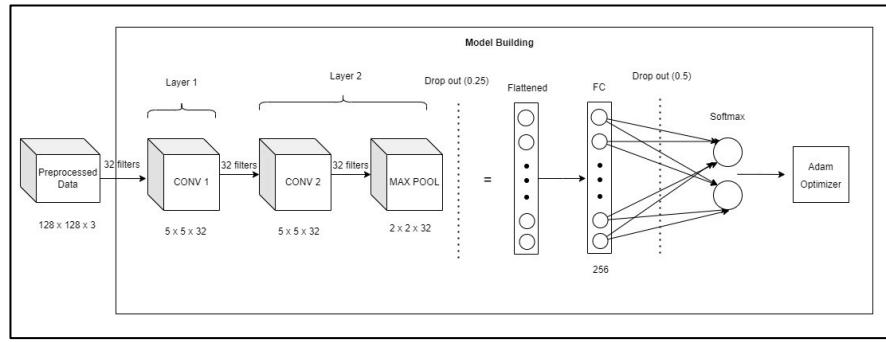
Gambar 3.6 Potongan *Source Code* proses ELA

3. *Preprocessing*

Tahap *Preprocessing* dilakukan untuk mengurangi dimensi atau ukuran citra. Pada proses ini gambar yang telah dikonversi ke ELA tersebut akan di-*resize* menjadi gambar dengan ukuran 128 x 128. Setelahnya, dilakukan perubahan ukuran gambar. Langkah selanjutnya adalah melakukan normalisasi dengan membagi setiap nilai RGB dengan angka 255.0 untuk melakukan normalisasi, agar CNN lebih cepat konvergen (mencapai global minimum dari nilai loss milik data validasi) karena nilai dari setiap nilai RGB hanya berkisar antara 0 dan 1. Langkah selanjutnya adalah dengan mengubah label pada suatu data, di mana 0 merepresentasikan real, 1 merepresentasikan *splice* dan 2 merepresentasikan *copy-move* menjadi categorical value. Setelah itu dilakukan pembagian data latih dan data validasi menggunakan pembagian 80% untuk data latih dan 20% untuk data validasi.

4. *Training Model*

Langkah selanjutnya adalah menggunakan data latih dan data validasi untuk melakukan pelatihan model deep learning dengan menggunakan CNN. Optimasi diterapkan selama pelatihan adalah Adam optimizer, yang merupakan salah satu metode *adaptive learning rate*. Arsitektur lengkap yang digunakan pada bagian model building dapat dilihat pada gambar di bawah.



Gambar 3.7 Arsitektur pembangunan model CNN

a. Feature Extraction

Pada model deep learning yang digunakan layer pertama CNN terdiri dari convolutional layer dengan ukuran kernel sebesar 5x5 dan jumlah filter sebanyak 32. Layer kedua CNN terdiri dari convolutional layer dengan ukuran kernel sebesar 5x5 dan jumlah filter sebanyak 32, dan Max Pooling layer dengan ukuran 2x2. Kedua convolutional layer yang digunakan menggunakan kernel *initializer glorot uniform*, dan fungsi aktivasi ReLU untuk membuat neuron yang ada pada convolutional layer melakukan seleksi sehingga dapat menerima sinyal yang berguna dari data masukan.

Setelahnya, layer MaxPooling ditambahkan dropout sebesar 0.25 untuk mencegah overfitting. Layer berikutnya merupakan fully connected layer dengan jumlah neuron sebanyak 256 dan fungsi aktivasi ReLU. Setelah fully connected

layer , akan ditambah dropout sebesar 0.5 untuk mencegah overfitting. Layer output yang digunakan memiliki fungsi aktivasi *softmax*.

Pada arsitektur yang digunakan, hanya dua convolutional layer yang dibutuhkan, karena hasil yang dihasilkan dari proses konversi menjadi gambar ELA dapat menonjolkan fitur-fitur penting untuk mengetahui apakah sebuah gambar asli atau sudah mengalami modifikasi dengan baik.

b. *Classification*

Pada tahap ini hasil dari ekstraksi fitur akan menjadi input pada proses ini, dimana proses *Fully Connected Layer (FC)* yang memiliki susunan yaitu *input layer, hidden layer, output layer, activation function, dan loss function*. Input ini nantinya berbentuk vektor sehingga *activation map* yang hasilnya berbentuk *array* harus melewati proses *reshape*. Hasil dari *reshape* kemudian digunakan sebagai *input* pada *Fully Connected Layer* pada proses klasifikasi. Pada *Classification Layer* ini, ada beberapa tahapan yang diterapkan yaitu *flatten* yang berfungsi untuk merubah output dari proses konvolusi berupa matriks menjadi vektor. Selanjutnya diteruskan pada proses klasifikasi menggunakan jumlah neuron yang telah ditentukan. Layer *dropout* juga digunakan untuk mencegah

terjadinya *overfitting* (kondisi dimana model terlalu sempurna mendeskripsikan data training). *Dropout* bekerja untuk menghilangkan beberapa neuron yang tidak terlalu berguna sehingga dapat mempercepat proses *training*. Kelas citra kemudian diklasifikasikan menjadi 2 kelas berdasarkan nilai dari neuron menggunakan activation *softmax*.

```
#Epoch
epochs = 20

#Batch Size
batch_size = 16

#Learning Rate
init_lr = 1e-4

#Optimizer
optimizer = Adam(learning_rate = init_lr, decay = init_lr/epochs)
```

Gambar 3.8 Parameter yang digunakan untuk *training*

Gambar 3.7 merupakan penentuan parameter awal yang akan digunakan untuk proses *training* model. Untuk *Epoch* berjumlah 20 serta ukuran *batch size* nya yaitu 32, batch size adalah jumlah sampel yang disebarluaskan ke dalam arsitektur neural network. *Sample per epoch* adalah jumlah sampel yang digunakan dalam tahap pelatihan. Jumlah sampel yang digunakan sebanyak 4711 data. Dalam penelitian ini, digunakan 2 kelas yaitu kelas *Real* dan *Fake*. Kemudian learning rate menggunakan nilai 0,0001.

Berikut merupakan *layer* arsitektur CNN yang digunakan dalam penelitian ini :

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 124, 124, 32)	2432
conv2d_1 (Conv2D)	(None, 120, 120, 32)	25632
max_pooling2d (MaxPooling2D)	(None, 60, 60, 32)	0
dropout (Dropout)	(None, 60, 60, 32)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 256)	29491456
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 3)	771
<hr/>		
Total params: 29,520,291		
Trainable params: 29,520,291		
Non-trainable params: 0		

Gambar 3.9 Layer Arsitektur CNN yang digunakan

c. *Model Evaluation*

Setelah menentukan arsitektur CNN yang digunakan proses *Training* pun dilakukan. Proses *training* ini bertujuan untuk melatih algoritma CNN dalam mengenali *dataset*-nya dan membentuk sebuah model berdasarkan pelatihan tersebut.

```

hist = model.fit(X_train,
                  Y_train,
                  batch_size = batch_size,
                  epochs = epochs,
                  validation_data = (X_val, Y_val),
                  callbacks=[save]
                )

Epoch 1/20
115/115 [=====] - 9s 43ms/step - loss: 0.7806 - accuracy: 0.6385 - val_loss: 0.5003 - val_accuracy: 0.
8304
Epoch 00001: val_loss improved from inf to 0.50034, saving model to ./best-model-01-0.6385-0.8304.h5
Epoch 2/20
115/115 [=====] - 4s 36ms/step - loss: 0.4966 - accuracy: 0.8108 - val_loss: 0.4181 - val_accuracy: 0.
8457
Epoch 00002: val_loss improved from 0.50034 to 0.41806, saving model to ./best-model-02-0.8108-0.8457.h5
Epoch 3/20
115/115 [=====] - 4s 36ms/step - loss: 0.4356 - accuracy: 0.8340 - val_loss: 0.3839 - val_accuracy: 0.
8391
Epoch 00003: val_loss improved from 0.41806 to 0.38388, saving model to ./best-model-03-0.8340-0.8391.h5
Epoch 4/20
115/115 [=====] - 4s 35ms/step - loss: 0.3848 - accuracy: 0.8568 - val_loss: 0.4435 - val_accuracy: 0.
8304
Epoch 00004: val_loss did not improve from 0.38388
Epoch 5/20
115/115 [=====] - 4s 35ms/step - loss: 0.3423 - accuracy: 0.8819 - val_loss: 0.4142 - val_accuracy: 0.
8152
Epoch 00005: val_loss did not improve from 0.38388
Epoch 6/20
115/115 [=====] - 4s 35ms/step - loss: 0.3055 - accuracy: 0.8933 - val_loss: 0.3012 - val_accuracy: 0.
9000
Epoch 00006: val_loss improved from 0.38388 to 0.30123, saving model to ./best-model-06-0.8933-0.9000.h5

```

Gambar 3.10 Proses *training* data

Setelah proses *training* dilakukan kemudian diperoleh sebuah model yang nantinya akan digunakan untuk *testing* data. Proses *testing* yang dilalui kurang lebih sama dengan proses *training* yaitu input data, preprocessing, kemudian data diklasifikasikan menggunakan model dari proses *training*. Proses ini bertujuan menguji seberapa akurat sebuah model yang dibentuk dari hasil *training*. Data yang digunakan pada proses *testing* berbeda dengan data pada proses *training*.

Berikut merupakan potongan *source code* untuk melakukan *testing* pada model yang telah diperoleh :

```
fake_image_path = 'D:\SKRIPSI\SKRIPSI\label_in_wild\images\5770.jpg'
image = prepare_image(fake_image_path)
image = image.reshape(-1, 128, 128, 3)
y_pred = model.predict(image)

y_pred_class = np.argmax(y_pred, axis = 1)[0]
display(Image.open(fake_image_path))
img = Image.open(fake_image_path)
img = img.resize((640, 512), Image.ANTIALIAS)
print(f'{y_pred[1]}\n{class_names[y_pred_class]} Confidence: {np.amax(y_pred) * 100:.2f} %')
display(img)

Class: fake Confidence: 99.76
```



Gambar 3.11 *source code testing data*

5. Analisis Kerja Sistem

Untuk melihat kinerja dari sistem klasifikasi bakteri dapat dilakukan dengan menghitung performa prediksi yang dilakukan sistem terhadap data citra inputan pada proses training dengan menggunakan bantuan *confusion matrix*.

Hasil evaluasi model ditampilkan dalam *confusion matrix* seperti pada tabel berikut :

Tabel 3.3 confusion matrix model evaluation

Confusion matrix

		True Label		
		Real	Splice	Copy Move
Predicted Label	Real	TP	FP	FP
	Splice	FN	TN	TN
	Copy Move	FN	TN	TN

Untuk menghitung nilai akurasi sistem, dapat dilakukan dengan persamaan:

$$Recall = \frac{TP}{TP+FN} \quad (3.1)$$

$$Precision = \frac{TP}{TP+FP} \quad (3.2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3.3)$$

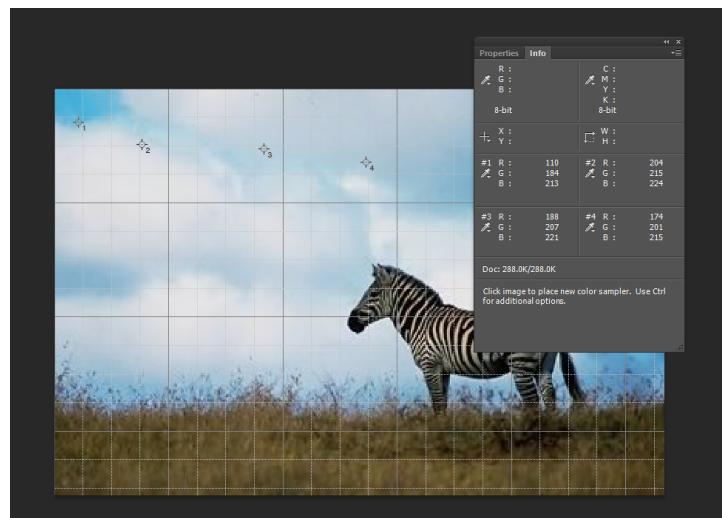
BAB IV

HASIL DAN PEMBAHASAN

4.1. Hasil Penelitian

4.1.1 Hasil Operasi Pada Citra

1. Konversi nilai RGB ke YCbCr (*luminance dan chominance*)



Gambar 4.1 gambar yang diidentifikasi nilai RGB dari beberapa pixel nya

dari gambar di atas diperoleh nilai RGB dari salah satu pixel nya kemudian dimasukkan ke dalam tabel di bawah untuk memperoleh nilai YCbCr :

Tabel 4.1 Tabel konversi RGB ke YCbCr

Y	R	Nilai	G	Nilai	B	Nilai	16	
	0.257	110	0.504	184	0.098	213		
28.27		92.736			20.874			
157.88								
Cb	R	Nilai	G	Nilai	B	Nilai	128	
	-0.148	110	- 0.291	184	0.439	213		
-16.28		-53.544			93.507			
151.683								
Cr	R	Nilai	G	Nilai	B	Nilai	128	
	0.439	110	- 0.368	184	-0.071	213		
48.29		-67.712			-15.123			
93.455								

Dari tabel di atas didapatkan nilai Y = 157.88 , Cb = 151.683 dan Cr = 93.455

2. Menghitung nilai *Error Level Analysis*

Pada tahap ini, nilai *Error Level Analysis* dihitung pada grid (1,1) sampai dengan kordinat pixel (8,8) menggunakan rumus yang ada pada gambar 2.1 :

Tabel 4.2. Tabel koordinat pixel dengan kernel 8 x 8

1.1							
							8.8

$$\mu = (157.88 + 151.683 + 93.455) / 3$$

$$\mu = 134.34$$

$$\Delta = ((157.88 - 93.455) * 0.49) + ((157.88 - 151.683) * 0.49)$$

$$\Delta = 31.568 + 3.037$$

$$\Delta = 34.605$$

$$Qn = 100 - 134.34 - 34.605$$

$$Qn = -68,945$$

3. Preprocessing

a. Resize Image

Tahap *Preprocessing* dilakukan untuk mengurangi dimensi atau ukuran citra. Pada proses ini gambar yang telah dikonversi ke ELA tersebut akan di-*resize* menjadi gambar dengan ukuran 128 x 128. Setelahnya, dilakukan perubahan ukuran gambar.

b. Normalisasi Data

Langkah selanjutnya adalah melakukan normalisasi dengan membagi setiap nilai RGB dengan angka 255.0 untuk melakukan normalisasi, agar CNN lebih cepat konvergen (mencapai global minimum dari nilai loss milik data validasi) karena nilai dari setiap nilai RGB hanya berkisar antara 0 dan 1.

c. Train Test Split

Langkah selanjutnya adalah dengan mengubah label pada suatu data, di mana 2 merepresentasikan *Copy-move*, 1 merepresentasikan *Splice*, dan 0 merepresentasikan real menjadi categorical value. Setelah itu dilakukan pembagian data latih dan data validasi menggunakan pembagian 80% untuk data latih dan 20% untuk data validasi.

4. Proses Train Data

Langkah selanjutnya adalah menggunakan data latih dan data validasi untuk melakukan pelatihan model deep learning dengan menggunakan CNN. Optimasi diterapkan selama pelatihan adalah Adam optimizer, yang merupakan salah satu metode *adaptive learning rate*.

a. Feature Extraction

Pada model deep learning yang digunakan layer pertama CNN terdiri dari convolutional layer dengan ukuran kernel sebesar 5x5 dan jumlah filter sebanyak 32. Layer kedua CNN terdiri dari convolutional layer dengan ukuran kernel sebesar 5x5 dan jumlah filter sebanyak 32, dan Max Pooling layer dengan ukuran 2x2. Kedua convolutional layer yang digunakan menggunakan kernel *initializer glorot uniform*, dan fungsi aktivasi ReLU untuk membuat neuron yang ada pada convolutional layer melakukan seleksi sehingga dapat menerima sinyal yang berguna dari data masukan.

Setelahnya, layer MaxPooling ditambahkan dropout sebesar 0.25 untuk mencegah overfitting. Layer berikutnya merupakan fully connected layer dengan jumlah neuron sebanyak 256 dan fungsi aktivasi ReLU. Setelah fully connected layer , akan ditambah dropout sebesar 0.5 untuk mencegah overfitting. Layer output yang digunakan memiliki fungsi aktivasi *softmax*.

Pada arsitektur yang digunakan, hanya dua convolutional layer yang dibutuhkan, karena hasil yang dihasilkan dari proses konversi menjadi gambar ELA dapat menonjolkan fitur-fitur penting untuk mengetahui apakah sebuah gambar asli atau sudah mengalami modifikasi dengan baik.

b. *Classification*

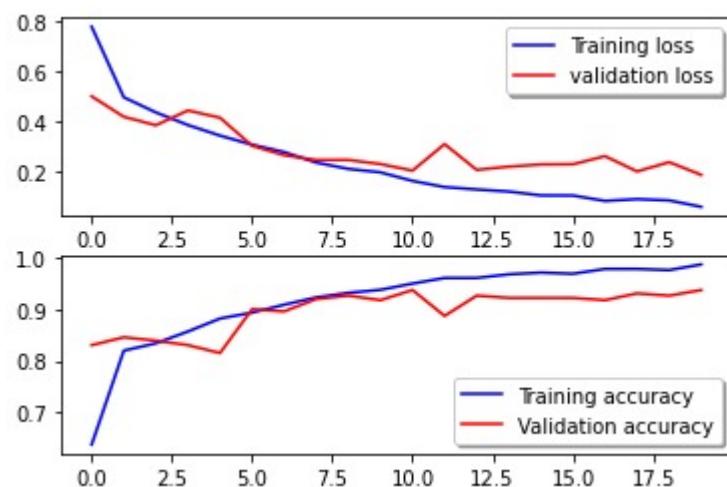
Pada tahap ini hasil dari ekstraksi fitur akan menjadi input pada proses ini, dimana proses *Fully Connected Layer (FC)* yang memiliki susunan yaitu *input layer, hidden layer, output layer, activation function, dan loss function*. Input ini nantinya berbentuk vektor sehingga *activation map* yang hasilnya berbentuk *array* harus melewati proses *reshape*. Hasil dari *reshape* kemudian digunakan sebagai *input* pada *Fully Connected Layer* pada proses klasifikasi. Pada *Classification Layer* ini, ada beberapa tahapan yang diterapkan yaitu *flatten* yang berfungsi untuk merubah output dari proses konvolusi berupa matriks menjadi vektor. Selanjutnya diteruskan pada proses klasifikasi menggunakan jumlah neuron yang telah ditentukan. Layer *dropout* juga digunakan untuk mencegah terjadinya *overfitting* (kondisi dimana model terlalu sempurna mendeskripsikan data training). *Dropout* bekerja untuk menghilangkan beberapa neuron yang tidak terlalu berguna sehingga dapat mempercepat proses *training*. Kelas citra kemudian

diklasifikasikan menjadi 2 kelas berdasarkan nilai dari neuron menggunakan activation *softmax*.

4.1.2 Hasil Pengujian Sistem

Bab ini menyajikan hasil dari pengujian sistem deteksi manipulasi gambar dengan *Error Level Analysis* dan klasifikasi menggunakan *Deep Learning*. Setelah proses training dilakukan kemudian diperoleh sebuah model yang nantinya akan digunakan untuk testing data. Proses testing yang dilalui kurang lebih sama dengan proses training yaitu input data, preprocessing, kemudian data diklasifikasikan menggunakan model dari proses training. Proses ini bertujuan menguji seberapa akurat sebuah model yang dibentuk dari hasil training. Data yang digunakan pada proses testing berbeda dengan data pada proses training.

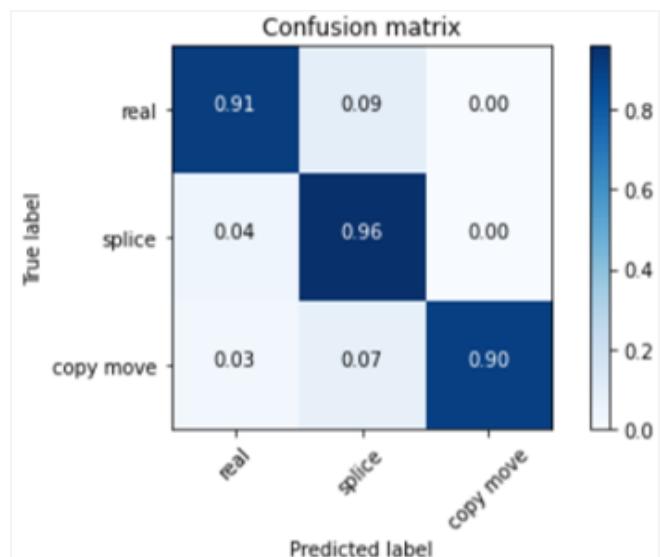
Hasil yang didapatkan dari metode yang diajukan memiliki akurasi maksimum sebesar 98,64%. Gambar kurva akurasi dan kurva loss dapat dilihat pada gambar di bawah.



Gambar 4.2 Kurva Akurasi dan Kurva Loss *Training Accuracy*

Dapat dilihat bahwa akurasi terbaik didapatkan pada *epoch* ke-20 dengan akurasi 0,9864. Jumlah *epoch* pelatihan yang dibutuhkan sedikit untuk mencapai konvergen, karena penggunaan fitur gambar hasil konversi ELA membuat pelatihan model menjadi jauh lebih efisien, dan normalisasi yang dilakukan pada nilai RGB untuk setiap pixel juga mempercepat konvergensi dari model CNN.

Pada penelitian ini, parameter untuk mengukur tingkat keberhasilan model adalah nilai akurasi. Nilai akurasi model dapat ditentukan dengan melakukan pengujian menggunakan data testing. *Confusion* matrix akan digunakan untuk membantu dalam melakukan perhitungan akurasi sistem. Adapun hasil kinerja sistem terhadap 460 data testing dapat dilihat pada gambar berikut.



Gambar 4.3 *Confusion Matrix*

Gambar 4.3 menunjukkan akurasi dari sistem yang telah dibuat. Dimana, hasil akurasi tertinggi terdapat pada *class splice* dengan akurasi sebesar 96% sedangkan hasil akurasi terendah terdapat pada *class copy-move* dengan akurasi sebesar 90%. Sehingga akurasi rata-rata setelah menjumlah hasil akurasi dari setiap kelas kemudian dibagi dengan jumlah kelas didapatkan sebesar 92.33%.

4.1.3 Hasil Pengujian Deteksi Citra

Pada pengujian pemalsuan citra, menggunakan masing-masing 10 object gambar aktual yaitu Real, Splice, dan Copy Move. Dan akan diuji sehingga mendeteksi Prediksi citra Real, Splice, atau Copy Move.

Berikut tabel data pengujian citra;

Tabel 4.3 Uji deteksi citra asli

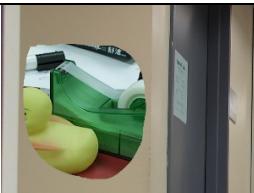
NO	GAMBAR	AKTUAL	PREDIKSI	AKURASI (%)
1		Real	Real	87,95
2		Real	Real	86,34

3		Real	Real	99,77
4		Real	Real	94,62
5		Real	Real	92,55
6		Real	Real	99,77

7		Real	Splice	42,33
8		Real	Real	93,32
9		Real	Real	97,88
10		Real	Real	92,56

Pada tabel 4.3 pengujian deteksi citra real mampu mendekripsi citra dengan baik, dengan nilai rata-rata mencapai 91,71%.

Tabel 4.4 Uji deteksi citra Splice

NO	GAMBAR	AKTUAL	PREDIKSI	AKURASI (%)
1		Splice	Splice	97,68
2		Splice	Splice	90,19
3		Splice	Copy Move	58,45
4		Splice	Splice	88,44
5		Splice	Copy Move	55,72

6		Splice	Splice	96,17
7		Splice	Splice	59,21
8		Splice	Splice	97,52
9		Splice	Splice	94,29
10		Splice	Splice	97,41

Pada tabel 4.4 pengujian deteksi citra Splice mampu mendeteksi citra dengan baik, dengan nilai rata-rata mencapai 83,51%.

Tabel 4.5 Uji citra Copy Move

NO	GAMBAR	AKTUAL	PREDIKSI	AKURASI (%)
1		Copy Move	Copy Move	99,63
2		Copy Move	Copy Move	54,82
3		Copy Move	Copy Move	96,40
4		Copy Move	Copy Move	85,85

5		Copy Move	Copy Move	41,04
6		Copy Move	Copy Move	45,82
7		Copy Move	Copy Move	74,02
8		Copy Move	Copy Move	85,45

9		Copy Move	Copy Move	89,92
10		Copy Move	Copy Move	89,64

Pada tabel 4.5 pengujian deteksi citra Copy Move mampu mendeteksi citra dengan baik, dengan nilai rata-rata mencapai 76,26%.

4.2. Pembahasan

Pada sub bab ini akan dibahas mengenai analysis dari akurasi sistem dalam deteksi pemalsuan gambar. Kesalahan deteksi terjadi akibat kurangnya data training yang bervariasi serta adanya kemiripan nilai kualitas pixel dalam class gambar real, splice, dan copy move, Sehingga menyebabkan deteksi kurang akurat dan kesalahan hasil deteksi Real, Splice, dan Copy Move, seperti pada contoh hasil pengujian berikut.

	Splice	Copy Move	58,45
	Copy Move	Copy Move	54,82

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian deteksi citra gambar, sistem mampu mendeteksi gambar dengan total akurasi rata-rata mencapai 91,71% Real, 83,51% Splice, 72,26% Copy Move. dengan menggunakan metode *Error Level Analysis* untuk *preprocessing* dan klasifikasi menggunakan metode *Convolutional Neural Network*.

Adapun kesimpulan dari penelitian ini sebagai berikut;

1. Sistem ini mampu mendeteksi pemalsuan gambar dengan cara preproceccing gambar terlebih dahulu menggunakan metode *Error Level Analysis*, kemudian melakukan Classification dengan menggunakan metode *Convolutional Neural Network*.
2. Metode *Error Level Analysis* mampu mendeteksi berbagai macam jenis citra mulai dari gambar Real, Splice, dan copy move.
3. Klasifikasi pemalsuan gambar dilakukan dengan menggunakan algoritma *Convolutional Neural Network*. Parameter yang digunakan yakni Input size 128 x 128 dengan 3 channel, epoch 20, batch size 16, dan learning rate 0,0001. Sehingga akurasi perhitungan algoritma *Convolutional Neural Network* dapat di implementasikan dengan baik.

5.2 Saran

Adapun saran untuk pengembangan penelitian selanjutnya sebagai berikut;

1. Dapat menampilkan hasil area pemalsuan citra.
2. Penambahan data training yang lebih banyak dan bervariasi, agar deteksi citra dapat lebih akurat.

DAFTAR PUSTAKA

- Baum, F. (n.d.). Exploring Different Types of Neural Networks – Qualcomm Developer Network. Retrieved December 9, 2021, from <https://developer.qualcomm.com/blog/exploring-different-types-neural-networks>.
- Darmawan, I.G.N.B., Sasmita, G.M.A., Buana, P.W., 2019. Pengembangan Metode Pendekripsi Modifikasi Citra Menggunakan Metode Error Level Analysis. J. Ilm. Merpati (Menara Penelit. Akad. Teknol. Informasi) 7, 29.
- Fibrianda, M., Bhawiyuga, A., 2018. Analisis Perbandingan Akurasi Deteksi Serangan Pada Jaringan Komputer Dengan Metode Naïve Bayes Dan Support Vector Machine (SVM) | Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer [WWW Document]. URL <https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/2559> (accessed 1.6.22).
- Gunawan, A., Lovenia, H., Pramudita, A., 2018. Deteksi Pemalsuan Gambar dengan ELA dan Deep Learning. <https://doi.org/10.13140/RG.2.2.28571.52006>
- Juditha, C., 2018. Hoax Communication Interactivity in Social Media and Anticipation (Interaksi Komunikasi Hoax Di Media Sosial Serta Antisipasinya). Pekommas 3, 261723. <https://doi.org/10.30818/jpkm.2018.2030104>
- Özgöbek, Ö., Gulla, J.A., 2018. Towards an Understanding of Fake News.

Parikh, S.B., Khedia, S.R., Atrey, P.K., 2019. A Framework to Detect Fake Tweet Images on Social Media, in: 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM). Presented at the 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), pp. 104–110.
<https://doi.org/10.1109/BigMM.2019.00-37>

Prabowo, D.A., Abdullah, D., 2018. Deteksi dan Perhitungan Objek Berdasarkan Warna Menggunakan Color Object Tracking. Pseudocode 5, 85–91.
<https://doi.org/10.33369/pseudocode.5.2.85-91>

Pradhitya, R., 2015. Pembangunan aplikasi deteksi dan tracking warna virtual drawing menggunakan algoritma color filtering (diploma). Universitas Komputer Indonesia.

Santoso, A., Ariyanto, G., 2018. Implementasi Deep Learning berbasis Keras untuk Pengenalan Wajah. Emitor: Jurnal Teknik Elektro 18, 15–21.
<https://doi.org/10.23917/emitor.v18i01.6235>

Warif, N.B.A., Idris, Mohd.Y.I., Wahab, A.W.A., Salleh, R., 2015. An evaluation of Error Level Analysis in image forensics, in: 2015 5th IEEE International Conference on System Engineering and Technology (ICSET). Presented at the 2015 5th IEEE International Conference on System Engineering and Technology (ICSET), pp. 23–28. <https://doi.org/10.1109/ICSEngT.2015.7412439>

LAMPIRAN

1. Source Code

Source code untuk penelitian ini dapat dilihat pada pranala berikut.

<https://github.com/RizkyAlfiansyah/skripsipython.git>

2. Dataset

Dataset yang digunakan pada penelitian ini dapat dilihat pada pranala berikut.

<https://www.kaggle.com/divg07/casia-20-image-tampering-detection-dataset>

<https://github.com/wenbihan/coverage>