PENGARUH REGULARIZATION PADA DETEKSI PNEUMONIA MENGGUNAKAN CNN

PRAKTIK KERJA LAPANGAN

JEHEZKIEL LUDWIG HADIWIDJAJA
311710012

TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MA CHUNG
MALANG
2021

LEMBAR PENGESAHAN

PENGARUH REGULARIZATION PADA DETEKSI PNEUMONIA MENGGUNAKAN CNN

Oleh:
JEHEZKIEL LUDWIG HADIWIDJAJA
311710012

dari
TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MA CHUNG

Dosen Pembimbing

Windra Swastika, S.Kom., MT., Ph.D.
20070039

Dekan Fakultas Sains dan Teknologi

Dr. Kestrilia Rega Prilianti, S.Si., M.Si.
20120035

KATA PENGANTAR

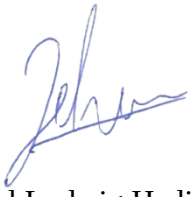
Puji dan syukur dipanjatkan kepada Tuhan Yang Maha Esa atas karunia-Nya laporan Praktik Kerja Lapangan (PKL) ini dapat dibuat dan diselesaikan dengan baik. Laporan dengan judul "Pengaruh Regularization Pada Deteksi Pneumonia Menggunakan CNN" disusun berdasarkan apa yang telah dilakukan pada saat praktik kerja lapangan selama 2 bulan 3 minggu mulai 15 Juni 2020 hingga 4 Agustus 2020.

Dalam penyusunan laporan praktik kerja lapangan ini Penulis memberikan ucapan terima kasih kepada berbagai pihak yang sudah membantu dalam proses pembuatan laporan ini, antara lain:

1. Orang tua dan teman-teman yang selalu memberi dukungan dan semangat kepada Penulis sehingga praktik kerja lapangan dapat terselesaikan,
2. Ibu Dr. Kestrilia Rega Prilianti, S.Si., M.Si. selaku Dekan Fakultas Sains dan Teknologi Universitas Ma Chung,
3. Bapak Hendry Setiawan, S.T., M.Kom. selaku Kepala Program Studi Teknik Informatika Universitas Ma Chung,
4. Bapak Mochamad Subianto, S.Kom., M.Cs. selaku pembimbing akademik,
5. Bapak Windra Swastika, S.Kom., MT., Ph.D. selaku pembimbing praktik kerja lapangan.

Laporan ini adalah salah satu prasyarat kelulusan. Dalam laporan ini tidak menutup kemungkinan terdapat beberapa kesalahan. Oleh sebab itu kritik dan saran sangat membantu bagi penulis untuk memperbaiki kekurangan yang ada. Demikian semoga praktik kerja lapangan ini bisa bermanfaat bagi semua pihak.

Malang, 23 Januari 2021



Jehezkiel Ludwig Hadiwidjaja

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab I
Pendahuluan

1.1 Latar Belakang

Pneumonia adalah penyakit radang paru-paru yang disebabkan oleh infeksi dan bisa menimbulkan gejala ringan hingga berat. Beberapa gejala tersebut seperti batuk berdahak, demam, dan sesak napas. Pneumonia juga dikenal dengan istilah paru-paru basah (Pane, 2020). Pane menjelaskan bahwa seseorang yang terinfeksi pneumonia dapat menyebabkan peradangan pada kantong-kantong udara (alveoli) di salah satu atau kedua paru-paru. Akibat dari hal tersebut, alveoli bisa dipenuhi cairan atau nanah sehingga menyebabkan penderitanya sulit bernapas. Pneumonia bisa terjadi akibat adanya infeksi bakteri, virus, dan jamur. Pada orang dewasa, pneumonia paling sering disebabkan oleh infeksi bakteri. Selain itu, pneumonia juga bisa dipicu oleh sumbatan saluran napas akibat tumor atau penyakit paru obstruksi kronis (PPOK).

Pneumonia merupakan salah satu masalah kesehatan utama pada orang-orang dewasa di banyak negara. Kasus pneumonia tidak mengenal kriteria usia ataupun jenis kelamin, pneumonia dapat menyerang siapapun, terutama pada orang yang memiliki daya imun yang menurun. Penyakit pneumonia dapat menular dari sejumlah kuman atau virus yang dilepaskan saat batuk atau bersin ke udara lalu kuman atau virus tersebut dihirup orang lain. Tak menutup kemungkinan, pneumonia juga dapat berkembang melalui infeksi yang disebabkan ketika kuman memasuki paru-paru melalui aliran darah (Anggriawan, 2016). Pengobatan pneumonia berbeda-beda tergantung dari jenis pneumonia. Umumnya pengobatan pneumonia bertujuan untuk mengatasi infeksi, meredakan gejala, dan mencegah komplikasi. Bagi penderita disarankan untuk banyak beristirahat, mengonsumsi makanan begizi seimbang, serta banyak minum air putih agar tidak kekurangan cairan. Jika pneumonia sudah parah, penderita perlu ditempatkan dalam ruang perawatan intensif dan dipasangkan ventilator yang merupakan mesin untuk membantu pernafasan. Salah satu cara untuk mendiagnosa dan menentukan tingkat

keparahan pneumonia seseorang adalah rontgen atau x-ray. X-ray dilakukan untuk memastikan kondisi paru-paru dan luas area paru yang mengalami infeksi atau peradangan.

Dalam bidang kesehatan, rontgen atau x-ray adalah prosedur pemeriksaan dengan menggunakan radiasi gelombang elektromagnetik guna menampilkan gambaran bagian dalam tubuh. Gambaran dari benda padat seperti tulang atau besi ditampilkan sebagai area berwarna putih, sedangkan udara yang terdapat pada paru-paru akan tampak berwarna hitam, dan gambaran dari lemak atau otot ditampilkan dengan warna abu-abu. Pada tanggal 9 November 1895, seorang fisikawan asal Jerman Wilhelm Conrad Rontgen tak sengaja menemukan x-ray (Pratama, 2018).

Penelitian ini membandingkan hasil citra x-ray bagian dada untuk mengidentifikasi terjangkit pneumonia atau tidak, dengan atau tanpa regularization. Regularization adalah teknik yang digunakan untuk meminimalisasi error dari generalisasi tanpa mempengaruhi error yang terlalu besar pada training. Ada berbagai cara melakukan regularisasi, dalam penelitian ini menggunakan regularisasi dengan cara Image Augmentation, yaitu menambah data latih dengan cara merotasi, memperbesar bentuk gambar dan dengan cara Dropout, yaitu mengurangi hasil data latih.

Sebelumnya sudah ada yang melakukan penelitian mengklasifikasi pneumonia pada citra x-ray paru-paru dengan CNN. Penelitian tersebut menguji kinerja CNN dalam menangani dataset baru yang diperoleh dari platform Kaggle. Berdasarkan hasil pengujian, diperoleh rata-rata nilai akurasi dan rata-rata nilai loss secara sekuensial sebesar 89,58% dan 47,43%.

Penelitian yang lain juga melakukan klasifikasi penyakit pneumonia menggunakan metode CNN dengan optimasi Adaptive Momentum. Data yang digunakan berasal dari Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. Hasil klasifikasi mencapai 99.98% untuk data latih dengan epoch sebesar 100 dan akurasi pada data uji sebesar 78% yang berarti model tersebut dapat mengklasifikasi dengan benar sebanyak 78% dari data uji.

1.2 Batasan Masalah

Batasan masalah dari pengerjaan penelitian ini ialah sebagai berikut.

1. Penelitian dilakukan dengan menggunakan data dari platform Kaggle dengan total 5216 citra x-ray dada.
2. Arsitektur model CNN dibuat menggunakan *library* Keras.

1.3 Rumusan Masalah

Sesuai dengan latar belakang yang sudah dijelaskan di atas, rumusan masalah yang digunakan pada penelitian ini untuk mengetahui apakah regularisasi mempengaruhi hasil dari deteksi pneumonia dengan menggunakan CNN.

1.4 Tujuan

Tujuan dari pengerjaan penelitian ini ialah menguji pengaruh regularisasi pada deteksi pneumonia.

1.5 Manfaat

Manfaat dari pengerjaan penelitian ini ialah sebagai berikut.

1. Bagi Universitas Ma Chung khususnya Program Studi Teknik Informatika dapat mempersiapkan lulusan yang berkompeten dan siap kerja dengan memberikan bekal kepada mahasiswa selama proses kegiatan Praktik Kerja Lapangan.
2. Bagi Mahasiswa dapat menerapkan ilmu yang telah diperoleh selama belajar di Universitas Ma Chung dan memberikan pengalaman dunia kerja.

Bab II
Tinjauan Pustaka

2.1 Pneumonia

Pneumonia adalah penyakit radang paru-paru yang disebabkan oleh infeksi dan bisa menimbulkan gejala ringan hingga berat. Beberapa gejala sersebut seperti batuk berdahak, demam, dan sesak napas. Pneumonia juga dikenal dengan istilah paru-paru basah (Pane, 2020). Akibat dari pneumonia adalah penderitanya sulit bernapas. Pneumonia bisa terjadi akibat adanya infeksi bakteri, virus, dan jamur. Pneumonia paling sering disebabkan oleh infeksi bakteri. Selain itu, pneumonia juga bisa dipicu oleh sumbatan saluran napas akibat tumor atau penyakit paru obstruksi kronis (PPOK).

Pneumonia merupakan salah satu masalah kesehatan utama di banyak negara. Kasus pneumonia tidak mengenal kriteria usia ataupun jenis kelamin, pneumonia dapat menyerang siapapun, terutama pada orang yang memiliki daya imun yang menurun. Penyakit pneumonia dapat menular dari sejumlah kuman atau virus yang dilepaskan saat batuk atau bersin ke udara lalu kuman atau virus tersebut dihirup orang lain. Pneumonia juga dapat berkembang melalui infeksi yang disebabkan ketika kuman memasuki paru-paru melalui aliran darah (Anggriawan, 2016). Ada berbagai jenis pengobatan berbeda untuk mengobati pneumonia tergantung dari jenis pneumonia tersebut. Umumnya pengobatan pneumonia bertujuan untuk mengatasi infeksi, meredakan gejala, dan mencegah komplikasi. Bagi penderita disarankan untuk banyak beristirahat, mengonsumsi makanan begizi seimbang, serta banyak minum air putih agar tidak kekurangan cairan. Jika pneumonia sudah parah, penderita perlu ditempatkan dalam ruang perawatan intensif dan dipasangkan ventilator yang merupakan mesin untuk membantu pernafasan.

2.2 X-ray

Salah satu cara untuk mendiagnosa dan menentukan tingkat keparahan pneumonia seseorang adalah rontgen atau x-ray. X-ray dilakukan untuk memastikan kondisi paru-paru dan luas area paru yang mengalami infeksi atau peradangan. Dalam bidang kesehatan, rontgen atau x-ray adalah prosedur pemeriksaan dengan menggunakan radiasi gelombang elektromagnetik guna menampilkan gambaran bagian dalam tubuh. Gambaran dari benda padat seperti tulang atau besi ditampilkan sebagai area berwarna putih, sedangkan udara yang terdapat pada paru-paru akan tampak berwarna hitam, dan gambaran dari lemak atau otot ditampilkan dengan warna abu-abu. Pada tanggal 9 November 1895, seorang fisikawan asal Jerman Wilhelm Conrad Rontgen tak sengaja menemukan x-ray (Pratama, 2018).

Gambar 2.1 X-ray

Gambar 2.1 sebelah kiri adalah gambar x-ray dada yang normal, gambar yang tengah adalah gambar x-ray dada yang mengalami pneumonia oleh bakteri, dan sebelah kanan adalah gambar x-ray dada yang mengalami pneumonia oleh virus.

2.3 Artificial Intelligence

Kecerdasan buatan (*Artificial Intelligence* atau AI) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu entitas buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (games), logika fuzzy, jaringan saraf tiruan dan robotika.

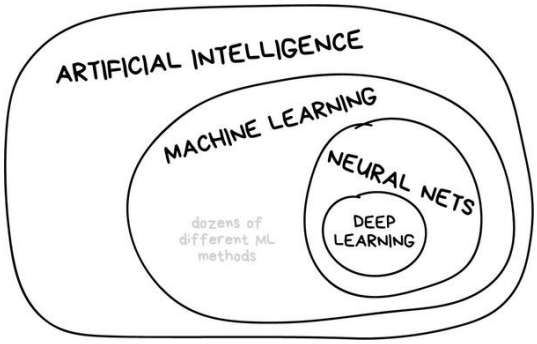
AI lebih banyak dianggap sebagai robot berbentuk menyerupai manusia padahal AI tidak selalu berbentuk robot yang menyerupai manusia. Ada 3 jenis kecerdasan buatan, yaitu *symbol-manipulating* AI, *neural* AI, dan *neural networks* (PT Dewaweb, 2018). *Symbol-manipulating* AI dibuat dengan menentukan aturan / *rule* yang sesuai dengan keadaan. *Symbol-manipulating* AI cocok dalam memprediksi keadaan dengan aturan yang jelas (Dickson, 2020). *Neural* AI merupakan jenis AI yang sangat populer dikalangan ilmuwan komputer pada akhir tahun 80-an. Berbeda dengan *symbol-manipulating* AI, *neural* AI tidak direpresentasikan lewat simbol, tetapi lebih ke neuron buatan dan koneksinya, semacam otak yang direkonstruksi. Pengetahuan yang dikumpul dipecah menjadi bagian kecil (disebut *neuron*) dan kemudian dihubungkan kembali menjadi kelompok-kelompok. Sistem saraf dilatih dan distimulasi agar jaringan saraf bisa mengumpulkan pengalaman dan tumbuh menjadi lebih besar. *Neural networks* adalah rangkaian algoritma mengenali hubungan dalam suatu data melalui proses cara kerja seperti otak manusia (Chen, 2020). *Neural networks* dapat beradaptasi dengan adanya perubahan *input* sehingga *neural networks* memberikan hasil yang terbaik.

2.4 Machine Learning

Machine Learning adalah bagian dari kecerdasan buatan. *Machine Learning* adalah proses komputer untuk belajar dari data. Tanpa adanya data, komputer tidak akan bisa belajar apa-apa. Semua pengetahuan *machine learning* pasti akan melibatkan data. Data bisa saja sama, akan tetapi algoritma dan pendekatannya berbeda-beda untuk mendapatkan hasil yang optimal. Metode *machine learning* dibagi menjadi 4, yaitu *supervised learning*, *unsupervised learning*, *semi-unsupervised learning*, dan *reinforcement learning* (Fahriz, 2019).

2.5 Deep Learning

Deep Learning adalah salah satu algoritma dari *Machine Learning* atau berada di lingkup *Machine Learning*. Sama halnya dengan *Machine Learning*, *Deep Learning* adalah metode pembelajaran yang dilakukan oleh mesin dengan cara meniru bagaimana sistem dasar otak manusia bekerja.



Gambar 2.2 Ruang lingkup AI, *machine learning*, *deep learning*

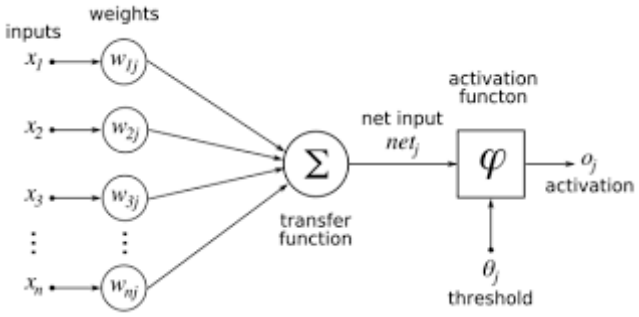
Sumber (https://id.quora.com/Apa-perbedaan-Machine-Learning-dan-Deep-Learning)

Gambar 2.2 menggambarkan ruang lingkup *Artificial Intelligence*, *Machine Learning*, *Deep Learning*. AI adalah bidang ilmu yang luas dan memiliki berbagai varian salah satunya adalah *Machine Learning*. *Deep Learning* adalah salah satu algoritma dari *Machine Learning* yang menggunakan *Artificial Neural Network*.

2.6 Artificial Neural Network

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan merupakan sebuah teknik pengolahan informasi yang terinspirasi oleh cara kerja sistem saraf biologis, yaitu sel otak manusia dalam memproses informasi. Teknik ini adalah struktur sistem pengolahan informasi yang bersifat unik dan beragam untuk tiap aplikasi. Neural Network terdiri dari sejumlah besar elemen pemrosesan informasi (neuron) yang saling terhubung dan bekerja bersama-sama untuk menyelesaikan sebuah masalah tertentu, yang pada umumnya adalah masalah klasifikasi ataupun prediksi (Widiputra, 2016).

Cara kerja dari Neural Network seperti manusia belajar dengan menggunakan contoh atau juga biasa disecut *supervised learning*. Proses belajar dalam sistem melibatkan penyesuaian koneksi yang ada pada neuron, dalam *Neural Network* penyesuaian tersebut dilakukan dengan menyesuaikan nilai bobot seperti pada gambar berikut.



Gambar 2.3 *Artificial neural network*

Sumber (https://dosen.perbanas.id/artificial-neural-network/)

Arsitektur dasar dari *Artificial Neural Network* seperti pada Gambar 2.3, terdapat beberapa *input* yang memiliki beban (*weight*) lalu ditotal semua ke *transfer function*. Dari sana dimasukkan ke dalam *activation function* yang hasilnya merupakan *output*. Rumus dari Gambar 2.2, *transfer function* adalah sebagai berikut.

$$z = \sum_{i=1}^{n} x_i w_{ij} + b_j \quad (2.1)$$

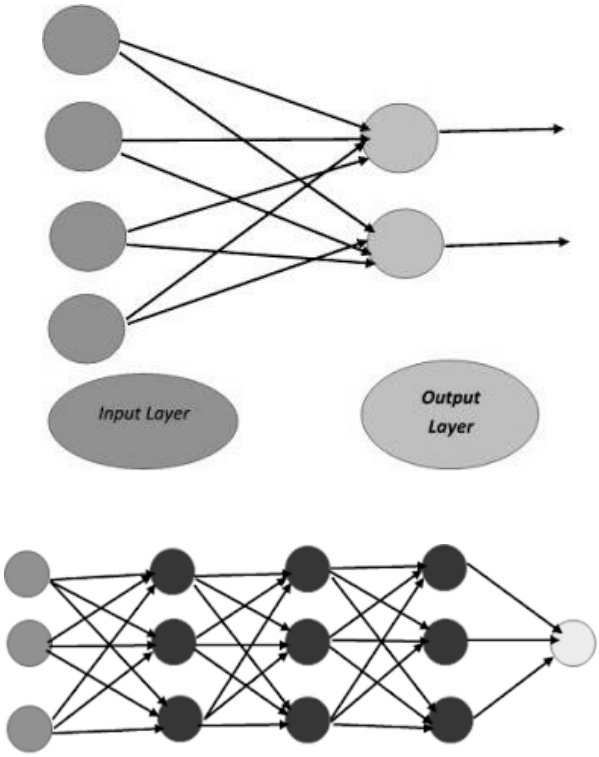
$z =$ *transfer function*

$x =$ *input*

$w =$ beban (*weight*)

$b =$ bias

Single Layer Perceptron (SLP) adalah arsitektur ANN yang hanya memiliki dua *layer* yaitu *input* dan *output*. Berbeda dengan *Multi Layer Perceptron* (MLP) terdapat *hidden layer* di antara *input* dan *output*.



Gambar 2.4 *Single Layer Perceptron* (SLP) dan *Multi Layer Perceptron* (MLP)

Sumber (https://www.i2tutorials.com/what-is-single-layer-perceptron-and-difference-between-single-layer-vs-multilayer-perceptron/)

Berdasarkan proses kerja dari *neural network* adalah *feed forward*. *Feed forward* adalah pergerakan sinyal satu arah dari *input layer* ke *output layer*, tidak ada nilai yang kembali ataupun pengulangan. *Feed forward* biasa digunakan untuk *pattern generation*, *pattern recognition*, dan klasifikasi (Singh, 2018).

Fungsi aktivasi digunakan untuk mengubah nilai dari *input* menjadi nilai *ouput*. Ada beberapa fungsi aktivasi yang biasa digunakan, seperti *linear activation function* dan *sigmoid activation function*. Fungsi aktivasi linear adalah fungsi identifikasi, jadi nilai dari *input* yang akan menjadi nilai *output*. Berbeda dengan fungsi aktivasi *sigmoid*, nilai dari *input* dimasukkan ke dalam fungsi *sigmoid*, hasil dari fungsi tersebut yang menjadi nilai *ouput*. Fungsi *sigmoid* adalah sebagi berikut.

$$y = \sigma(z) = \frac{1}{1+e^{-z}} \tag{2.2}$$

Jika *feed forward* digunakan untuk menghasilkan nilai yang diinginkan, *backpropagation* digunakan untuk memperbaiki nilai dengan meng-*update* bobot sehingga pada saat *feed forward* berikutnya bisa menghasilkan nilai yang lebih baik.

Awalnya seperti yang sudah dijelaskan pada *feed forward*, nilai dari hasil *feed forward* dibandingkan dengan nilai yang sebenarnya, selisih dari kedua nilai tersebut disebut nilai *error*. Untuk menghitung ulang *feed forward* maka diperlukan meng-*update* bobot dengan menggunakan turunan dan nilai *error*. Rumus untuk memperbaiki bobot adalah sebagai berikut.

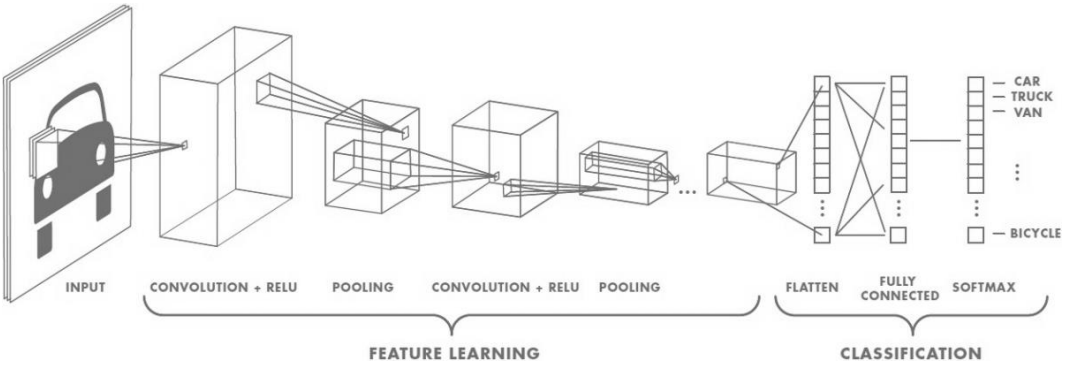
$$w_{new} = w_{old} - \alpha \frac{\partial E}{\partial w} \tag{2.3}$$

Di mana α adalah *learning rate* (antara 0 hingga 1) yang menentukan seberapa cepat proses pembelajaran dari model dan $\frac{\partial E}{\partial w}$ adalah nilai turunan parsial *E* terhadap *w*. Rumus dari turunan parsial adalah seperti berikut.

$$\frac{\partial E}{\partial w_{ij}} = -(t_j - y_j) y_j (1 - y_j) x_i \tag{2.4}$$

Di mana *t* adalah nilai yang sebenarnya sehingga dari rumus di atas dapat menentukan nilai bobot yang baru, lalu melakukan proses *feed forward* kembali dengan nilai bobot yang baru (Adam, 2019).

2.7 Convolutional Neural Network



Gambar 2.5 Arsitektur CNN

Sumber (https://au.mathworks.com/discovery/convolutional-neural-network-matlab.html)

Arsitektur dari CNN dibagi menjadi 2 bagian besar, *Feature Learning* dan *Classification* (Sena, 2017). *Feature Learning* mentranslasikan suatu input menjadi ciri dari input tersebut yang berbentuk angka-angka dalam vektor. Lapisan ini terdiri dari *Convolutional Layer* dan *Pooling Layer*.

- *Convolutional Layer* akan menghitung output dari neuron yang terhubung ke daerah lokal dalam input. *Rectified Linear Unit* (ReLU) akan menghilangkan *vanishing gradient* dengan cara menerapkan fungsi aktivasi elemen sebagai $f(x) = \max(0, x)$ alias aktivasi elemen akan dilakukan saat berada di ambang batas 0.
- *Pooling Layer* adalah lapisan yang mengurangi dimensi sehingga mempercepat komputasi karena parameter semakin sedikit dan dapat mengatasi *overfitting*. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. *Max Pooling* untuk menentukan nilai maksimum tiap pergeseran filter, sementara *Average Pooling* akan menentukan nilai rata-ratanya.

Classification berguna untuk mengklasifikasikan tiap neuron yang telah diekstraksi fitur pada sebelumnya. Lapisan ini terdiri dari *Flatten, fully-connected*, dan *Softmax*. *Flatten* membentuk ulang fitur (*reshape feature map*) menjadi sebuah vektor agar bisa kita gunakan sebagai input dari *fully-connected layer*. *Fully-connected* akan menghitung skor kelas.

Fungsi *Softmax* menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan dan akan membantu untuk menentukan kelas target untuk input yang diberikan. Selain ReLU dan *Softmax*, fungsi lain yaitu LeakyReLU adalah ReLU yang memperoleh sedikit nilai gradien positif saat unit tidak aktif.

Stride adalah salah satu filter *hyperparameter* yang menentukan jumlah pergeseran filter pada *convolution layer*. Semakin kecil *stride* maka akan semakin detail informasi yang didapat tetapi membutuhkan komputasi yang lebih. Namun dengan menggunakan *stride* yang kecil tidak selalu mendapatkan hasil yang bagus. *Padding* adalah filter *hyperparameter* untuk menentukan jumlah piksel berisi 0 yang akan ditambahkan di setiap sisi dari *input* (Sena, 2017).

2.7.1 ShallowNet

ShallowNet (Schirrmeister, et al., 2017) adalah salah satu arsitektur CNN yang memiliki 1 *Convolutional layer*, 2 *Pooling layer*, dan 2 *Fully-Connected layer* untuk *input shape* 64x64 dan 1 *Convolutional layer*, 3 *Pooling layer*, dan 2 *Fully-Connected layer* untuk *input shape* 128x128. *Activation function* arsitektur ShallowNet adalah ReLu dan *Optimizer*-nya adalah RMSProp dan Adam.

Tabel 2.1 Arsitektur ShallowNet (*Input* 64×64)

Layer (type)	Output Shape
Conv2D	(None, 64, 64, 32)
Activation	(None, 64, 64, 32)
MaxPooling2D	(None, 32, 32, 32)
MaxPooling2D	(None, 16, 16, 32)
Flatten	(None, 8192)
Dense	(None, 4000)
Dense	(None, 2)
Activation	(None, 2)

Tabel 2.2 Arsitektur ShallowNet (*Input* 128×128)

Layer (type)	Output Shape
Conv2D	(None, 128, 128, 32)
Activation	(None, 128, 128, 32)
MaxPooling2D	(None, 64, 64, 32)
MaxPooling2D	(None, 32, 32, 32)
MaxPooling2D	(None, 16, 16, 32)
Flatten	(None, 8192)
Dense	(None, 4000)
Dense	(None, 2)
Activation	(None, 2)

2.7.2 MicroVGGNet

Arsitektur MicroVGGNet (Simonyan & Zisserman, 2015) memiliki 4 *Convolutional layer*, 2 *Pooling layer*, dan 2 *Fully-Connected layer*. *Activation function* MicroVGGNet adalah ReLu, *Optimizer*-nya Adam, dan *input shape*-nya adalah 64x64.

Tabel 2.3 Arsitektur MicroVGGNet

Layer (type)	Output Shape
Conv2D	(None, 64, 64, 32)
Activation	(None, 64, 64, 32)
Conv2D	(None, 64, 64, 32)
Activation	(None, 64, 64, 32)
MaxPooling2D	(None, 32, 32, 32)
Conv2D	(None, 32, 32, 64)
Activation	(None, 32, 32, 64)
Conv2D	(None, 32, 32, 64)
Activation	(None, 32, 32, 64)
MaxPooling2D	(None, 16, 16, 64)
Flatten	(None, 16384)
Dense	(None, 5000)
Dense	(None, 2)
Activation	(None, 2)

2.7.3 BaselineNet

Arsitektur BaselineNet (Yi, et al., 2016) adalah arsitektur CNN yang menggunakan *input shape* 64x64. Ada beberapa jenis BaselineNet yang diuji pada praktik kerja lapangan ini. *Optimizer* BaselineNet adalah Adam dan RMSProp. *Activation function*-nya ReLu dan LeakyReLu.

Tabel 2.4 Arsitektur BaselineNet

Layer (type)	Output Shape
Conv2D	(None, 64, 64, 32)
MaxPooling2D	(None, 32, 32, 32)
Conv2D	(None, 32, 32, 64)
MaxPooling2D	(None, 16, 16, 64)
Conv2D	(None, 16, 16, 128)
MaxPooling2D	(None, 8, 8, 128)
Conv2D	(None, 8, 8, 128)
MaxPooling2D	(None, 4, 4, 128)
Flatten	(None, 2048)
Dense	(None, 1000)
Dense	(None, 2)

Tabel 2.5 Arsitektur BaselineNet (LeakyRelu)

Layer (type)	Output Shape
Conv2D	(None, 64, 64, 32)
LeakyReLU	(None, 64, 64, 32)
MaxPooling2D	(None, 32, 32, 32)
Conv2D	(None, 32, 32, 64)
LeakyReLU	(None, 32, 32, 64)
MaxPooling2D	(None, 16, 16, 64)
Conv2D	(None, 16, 16, 128)
LeakyReLU	(None, 16, 16, 128)
MaxPooling2D	(None, 8, 8, 128)
Conv2D	(None, 8, 8, 128)
LeakyReLU	(None, 8, 8, 128)
MaxPooling2D	(None, 4, 4, 128)
Flatten	(None, 2048)
Dense	(None, 1000)
LeakyReLU	(None, 1000)
Dropout	(None, 1000)
Dense	(None, 2)

Tabel 2.6 Arsitektur BaselineNet (No Padding)

Layer (type)	Output Shape
Conv2D	(None, 62, 62, 32)
MaxPooling2D	(None, 31, 31, 32)
Conv2D	(None, 29, 29, 64)
MaxPooling2D	(None, 14, 14, 64)
Conv2D	(None, 12, 12, 128)
MaxPooling2D	(None, 6, 6, 128)
Conv2D	(None, 4, 4, 256)
MaxPooling2D	(None, 2, 2, 256)
Flatten	(None, 1024)
Dense	(None, 500)
Dropout	(None, 500)
Dense	(None, 500)
Dropout	(None, 500)
Dense	(None, 500)
Dropout	(None, 500)
Dense	(None, 2)

2.8 Regularization

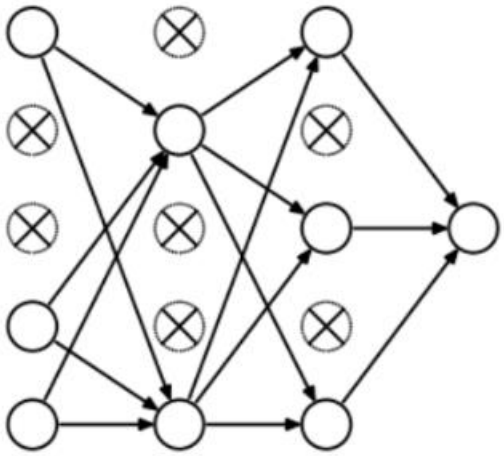
Regularization atau regularisasi adalah sebuah teknik yang mengubah algoritma pembelajaran model lebih baik dengan meminimalisasi kesalahaan (*error*) (Gylberth, 2018). Ada beberapa teknik regularisasi, seperti L1 *regularization*, L2 *regularization*, *dropout*, *data augmentation*, dan lainnya. L1 *regularization* juga biasa disebut *Lasso Regression* dan L2 *regularization* disebut *Ridge Regression*. Perbedaan dari kedua regularisasi tersebut adalah ada pada *penalty term* (Nagpal, 2017). Berikut rumus dari *Ridge Regression*.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \quad (2.5)$$

Berikut rumus dari *Lasso Regression.*

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \quad (2.6)$$

Dropout merupakan salah satu teknik yang banyak dipakai dan menghasilkan hasil yang baik. *Data augmentation* adalah teknik lain untuk mengurangi *error* dengan cara menambah data latih yang didapat melalui seperti merotasi citra, *flipping*, *scaling*, *shifting*, dll.



Gambar 2.6 Penggunaan *dropout*

Sumber (https://www.kaggle.com/sid321axn/regularization-techniques-in-deep-learning)

Gambar 2.7 Penggunaan *data augmentation*

Sumber (https://www.kaggle.com/sid321axn/regularization-techniques-in-deep-learning)

Dalam proses pada *convolutional neural network* diinginkan hasil dengan nilai *error* yang kecil. Nilai *error* tersebut berasal dari *loss function*. Beberapa *loss function* adalah sebagai berikut.

1. Binary Crossentropy

$$CE = -\sum_{i=1}^{C'=2} t_i \log(s_i) = -t_1 \log(s_1) - (1 - t_1) \log(1 - s_1) \quad (2.7)$$

Loss function ini digunakan untuk masalah klasifikasi biner. Klasifikasi yang di mana nilai target berada pada set {0, 1} atau hanya klasifikasi untuk 2 kelompok saja.

2. Categorical Crossentropy

$$CE = -\log\left(\frac{e^{s_i}}{\sum_j^C e^{s_i}}\right) \quad (2.8)$$

Categorical Crossentropy dapat digunakan untuk masalah klasifikasi *multi-class*. Klasifikasi yang di mana nilai target berada pada set {0, 1, 2, 3, …, n} atau bisa digunakan untuk klasifikasi lebih dari 2 kelompok.

3. Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \quad (2.9)$$

MSE memberi informasi seberapa dekat garis regresi dengan sekumpulan titik. Jarak dari titik ke garis regresi (*error*) dikuadratkan agar menghindari adanya hasil negatif serta memberi bobot lebih pada perbedaan yang lebih besar. MSE bekerja dengan mencari rata – rata kesalahan (*error*) dari sekumpulan titik.

4. Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i| \quad (2.10)$$

MAE mengukur besarnya rata – rata kesalahan dalam serangkaian prediksi, tanpa mempertimbangkan arahnya. Hampir sama dengan MSE, namun pada MAE jarak dari titik ke garis regresi (*error*) tidak dikuadratkan, namun diabsolutkan.

2.9 T-Test

T-test adalah salah satu metode pengujian statistik untuk menguji hipotesis. Jika ada hipotesis (H_0) maka terdapat juga alternatif (H_1). Untuk menunjukkan kebenaran atau kesalahan dari hipotesis diperlukan bukti yang signifikan (Baron, 2014). Suatu bukti hanya dapat dibuktikan dengan adanya data. Metode *t-test* akan menunjukkan besarnya pengaruh variabel independen secara parsial terhadap suatu variabel yang independen pula. *T-test* biasa digunakan untuk menguji data yang memiliki jumlah sedikit, yaitu kurang dari 30 (Kusaribe, 2020). *T-test* bisa digunakan untuk melihat perbedaan yang signifikan antara satu kelompok data dengan kelompok data lainnya. Ada beberapa macam *t-test*, yaitu *one sample test*, *paired sample test*, dan *independent sample test*. *One sample test* adalah teknik untuk menguji suatu nilai memiliki perbedaaan signifikan terhadap suatu rata-rata kelompok data. *Paired sample test* adalah digunakan untuk membandingkan suatu rata-rata dua variabel dalam satu kelompok data. *Independent sample test* adalah membandingkan kedua kelompok data memeiliki perbedaan yang signifikan.

2.10 Python

Gambar 2.8 Logo Python

Python adalah bahasa pemrograman agar pengguna dapat bekerja dengan cepat dan lebih efektif. Python dikembangkan sejak tahun 1989 dan dirilis pada tahun 1991 oleh Guido van Rossum. Penamaan Python diambil dari nama Monty Python yang merupakan sekelompok lawak dan dikarenakan Guido van Rossum menggemarinya. Dibandingkan bahasa pemrograman lain, Python memiliki sedikit *keyword* dan struktur yang sederhana, memiliki lebih dari 140.000 *library*, dan dikembangkan sebagai proyek *open source* atau gratis, bisa digunakan siapa saja (Sidiq, 2020).

2.11 NumPy

Gambar 2.9 Logo NumPy

NumPy merupakan proyek *open source* yang bertujuan untuk mengaktifkan komputasi numerik dengan Python. NumPy dibuat pada tahun 2005 yang dari awal dibuat memiliki fokus ke *library* numerik dan *numarray*. NumPy kependekan dari Numerical Python merupakan salah satu library yang berguna membantu pengguna Python untuk menganalisis. Selain Python, NumPy juga terintegrasi dengan bahasa pemrograman lain seperti C/C++ dan Fortran.

2.12 OpenCV

Gambar 2.10 Logo OpenCV

OpenCV (Open Source Computer Vision Library) adalah salah satu *machine learning software library* OpenCV dibangun untuk menyediakan infrastruktur umum dalam aplikasi citra komputer dan mempercepat penggunaan mesin dalam menangkap citra. OpenCV mempermudah bisnis dalam penggunaan dan modifikasi kode.

2.13 Keras

Gambar 2.11 Logo Keras

Keras adala sebuah API (Application Programming Interface) yang didesain sederhana, konsisten, meminimalkan jumlah tindakan pengguna yang diperlukan untuk kasus yang umum, dan memberikan pesan kesalahan yang jelas. Keras juga merupakan *framework* yang paling banyak digunakan dalam tim kejuaraan dalam Kaggle karena Keras lebih mudah dalam memulai eksperimen, mewadahi pengguna untuk merealisasikan ide dalam berkompetisi. Keras dibangun dengan TensorFlow 2.0. Model Keras dapat diubah menjadi JavaScript dan dijalankan pada *browser*.

2.14 Google Colab

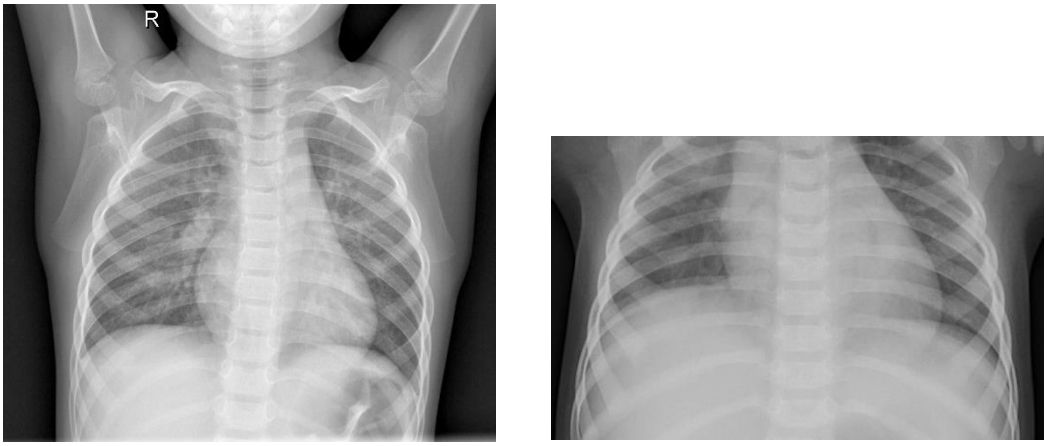
Gambar 2.12 Logo Google Colab

Colaboratory atau biasa disebut Colab adalah sebuah *website* yang memungkinkan penggunanya untuk membuat dan menjalankan Python melalui *browser* dengan mudah (tidak perlu mengatur apa-apa), gratis untuk menggunakan GPU, dan dapat dibagikan bersama pengguna lain. Tiap dokumen yang dibuat bukan halaman yang statik tetapi merupakan Colab Notebook untuk menulis dan menjalankan baris kode yang dibuat.

Bab III
Perancangan Sistem

3.1 Dataset Pneumonia

Dataset yang digunakan adalah dataset citra x-ray dada yang terdiri dari dua jenis gambar, yaitu normal dan yang terkena pneumonia. Total gambar ada sebanyak 1341 untuk gambar yang normal dan 3875 untuk gambar yang terkena pneumonia. Dataset didapat dari *website* Kaggle dengan *link* https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia. Berikut contoh gambarnya.



Gambar 3.1 Contoh dataset

3.2 Arsitektur Model

Penelitian ini dilakukan dengan menggunakan 12 jenis model arsitektur CNN dan masing-masing model dilakukan 5 kali percobaan. Penelitian ini dilakukan hingga epoch ke-20 kecuali model dengan 150 epoch (BaselineNet Regularization with No Padding 150 Epoch, BaselineNet Regularization with No Padding 150 Epoch LeakyReLU, dan BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU).

Tabel 3.1 12 Arsitektur Model

No	Model	Input Resolution	Optimizer	Activation Function	Regularization
1	ShallowNet	64×64	RMSprop	ReLU	None
2	ShallowNet 128×128	128×128	RMSprop	ReLU	None
3	ShallowNet Adam	64×64	Adam	ReLU	None
4	MicroVGGNet	64×64	Adam	ReLU	None
5	BaselineNet	64×64	Adam	ReLU	None
6	BaselineNet Regularization	64×64	Adam	ReLU	Image Augmentation + Dropout Layers
7	BaselineNet Regularization with LeakyReLU	64×64	Adam	LeakyReLU	Image Augmentation + Dropout Layers
8	BaselineNet Regularization with No Padding	64×64	Adam	ReLU	Image Augmentation + Dropout Layers
9	BaselineNet Regularization with No Padding RMSprop	64×64	RMSprop	ReLU	Image Augmentation + Dropout Layers
10	BaselineNet Regularization with No Padding 150 Epoch	64×64	Adam	ReLU	Image Augmentation + Dropout Layers
11	BaselineNet Regularization with No Padding 150 Epoch LeakyReLU	64×64	Adam	LeakyReLU	Image Augmentation + Dropout Layers
12	BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU	64×64	Adam	LeakyReLU	More Image Augmentation + Dropout Layers

3.3 Regularisasi

Jenis regularisasi yang digunakan pada penelitian ini adalah sebagai berikut.

1. Dropout layers
2. Data augmentation

   Pada regularisasi biasa untuk model ke-6 hingga ke-11, data augmentasi yang digunakan adalah:

   - Rotasi 10°

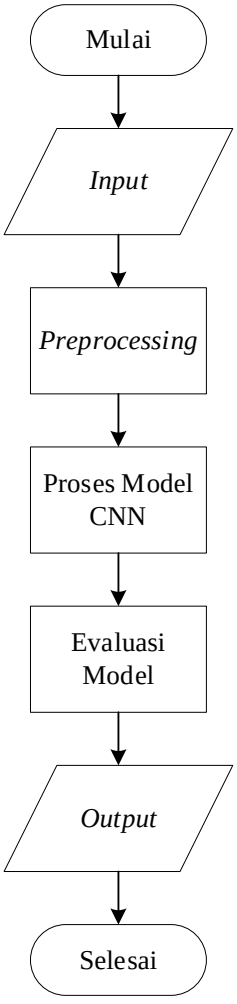
- Memperbesar 0.1x
- Menggeser gambar secara horizontal sebesar 0.1x
- Menggeser gambar secara vertikal sebesar 0.1x

Pada regularisasi+ untuk model ke-12, data augmentasi yang digunakan adalah 4 poin di atas ditambah dengan:

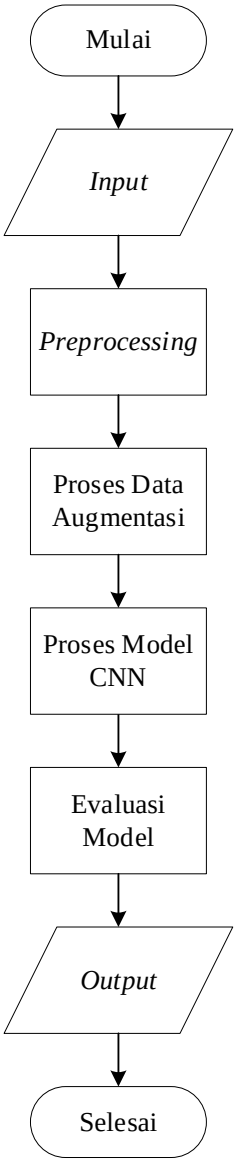
- Membalik gambar (flip) secara horizontal
- Membalik gambar (flip) secara vertikal

3.4 Desain Sistem

Sistem dibangun menjadi dua, yaitu model yang tidak menggunakan regularisasi dan model yang menggunakan regularisasi. Berikut gambar dari *flowchart*.



Gambar 3.2 *Flowchart* sistem tanpa regularisasi



Gambar 3.3 *Flowchart* sistem menggunakan regularisasi

Sistem dimulai dengan *input* berupa gambar dari dataset. Pada *preprocessing*, Ukuran gambar diubah sesuai dengan arsitektur yang akan digunakan. Gambar diberi label Pneumonia untuk yang terkena Pneumonia dan Normal untuk yang tidak terkena Pneumonia. Data dibagi menjadi data latih dan data uji dengan perbandingan 9:1. Pada sistem yang menggunakan regularisasi, data dilakukan augmentasi, sedangkan pada sistem yang tidak menggunakan regularisasi tidak dilakukan augmentasi. Lalu, proses model CNN sesuai dengan arsitektur yang telah dipaparkan di bab sebelumnya. Model dievaluasi *loss* dan *accuracy*-nya dan hasil berupa *confusion matrix*. Dari hasil yang ada dilakukan uji statistik dan didapatkan *output*

hasil analisa berupa berpengaruh atau tidak penggunaan regularisasi pada model yang sudah dibuat.

3.5 Pengujian dan Evaluasi

Pengujian yang dilakukan menggunakan *accuracy* dan *loss*. *Accuracy train* dan *test*, *loss train* dan *test* menjadi pertimbangan apakah model sudah baik dan apakah menggunakan regularisasi lebih baik daripada tidak menggunakan regularisasi. Nilai *accuracy* dan *loss* didapatkan dari *confusion matrix*.

Tabel 3.2 *Confusion Matrix*

Kondisi Sesungguhnya	Hasil Prediksi	
	Terkena Pneumonia	Normal
Terkena Pneumonia	*True Positive*	*False Negative*
Normal	*False Positive*	*True Negative*

Hasil prediksi di sini adalah hasil keluaran/output dari model pada data *test* yang akan dibandingkan dengan kondisi sesungguhnya. Setelah dibandingkan semua, akan didapatkan angka-angka seperti *accuracy* dan *loss*. Semakin tinggi angka *accuracy*, maka model semakin baik. Semakin rendah angka *loss* model semakin baik. Berikut adalah rumus dari *accuracy* dan *loss*.

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (3.1)$$

$$Loss = \frac{(FP+FN)}{(TP+FP+FN+TN)} \quad (3.2)$$

Keterangan:

TP = *True Positive* (hasil prediksi benar bahwa hasil berupa positif)

FP = *False Positive* (hasil prediksi salah / negatif, hasil yang benar berupa positif)

TN = *True Negative* (hasil prediksi benar bahwa hasil berupa negatif)

FP = *False Negative* (hasil prediksi salah / positif, hasil yang benar berupa negatif)

Uji statistik dilakukan menggunakan pengujian T-Test yaitu Independent T-Test menggunakan software SPSS. Pengujian ini dilakukan untuk membandingkan model yang menggunakan regularisasi dan model yang tidak menggunakan regularisasi, yaitu BaselineNet dan BaselineNet Regularization. Pengujian

dilakukan pada hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy*. Hipotesis awal (H_0) adalah kedua model sama dan H_1, kedua model tidak sama. Jika hasil pengujian mendapatkan nilai Sig. (2-*tailed*) di atas 0,05, maka dapat diartikan bahwa nilai data antara kedua model adalah homogen atau sama (gagal menolak H_0). Namun jika nilai Sig. ((2-*tailed*) di bawah 0,05 maka ada perbedaan yang signifikan pada kedua model (berhasil menolak H_0).

Pada *independent t-test* terdapat dua bentuk pengujian berdasarkan kesamaan varian (*equal variances assumed* dan *equal variances not assumed*). Pada kedua kelompok data dianggap memiliki kesamaan varian, perhitungan t-nya adalah sebagai berikut.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{3.3}$$

dengan

$$s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}} \tag{3.4}$$

$\bar{x}_1 =$ rata-rata kelompok data pertama

$\bar{x}_2 =$ rata-rata kelompok data kedua

$n_1 =$ banyaknya kelompok data pertama

$n_2 =$ banyaknya kelompok data kedua

$s_1 =$ standar deviasi kelompok data pertama

$s_2 =$ standar deviasi kelompok data kedua

$s_p =$ *pooled* standar deviasi

dan rumus *degrees of freedom* $df = n_1 + n_2 - 2$. Jika pada kedua kelompok data dianggap memiliki ketidaksamaan varian, maka perhitungan t-nya adalah sebagai berikut.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{3.5}$$

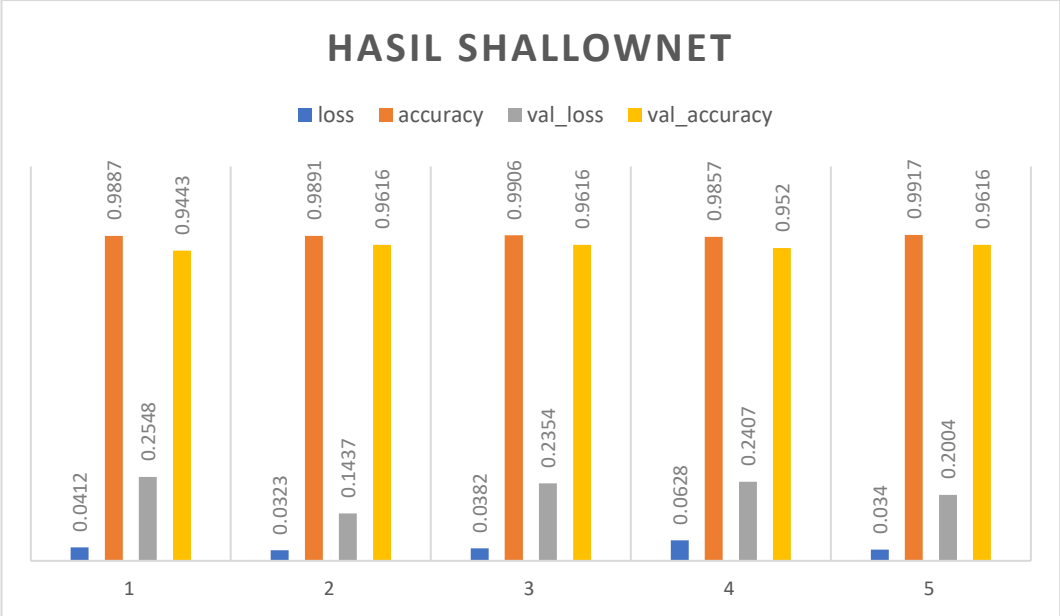
dengan perhitungan *degrees of freedom*

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1 - 1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2 - 1}\left(\frac{s_2^2}{n_2}\right)^2} \tag{3.6}$$

Bab IV
Hasil dan Pembahasan

4.1 ShallowNet

ShallowNet dengan *input* 64×64, epoch sebanyak 20, *optimizer* RMSprop, dan *activation* ReLU. Hasil yang didapat dari ShallowNet adalah 0.0417 untuk rata-rata *loss*, 0.98916 untuk rata-rata *accuracy*, 0.215 untuk rata-rata *val_loss*, dan 0.95622 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.1 dan hasil *Confusion Matrix* pada Tabel 4.1.



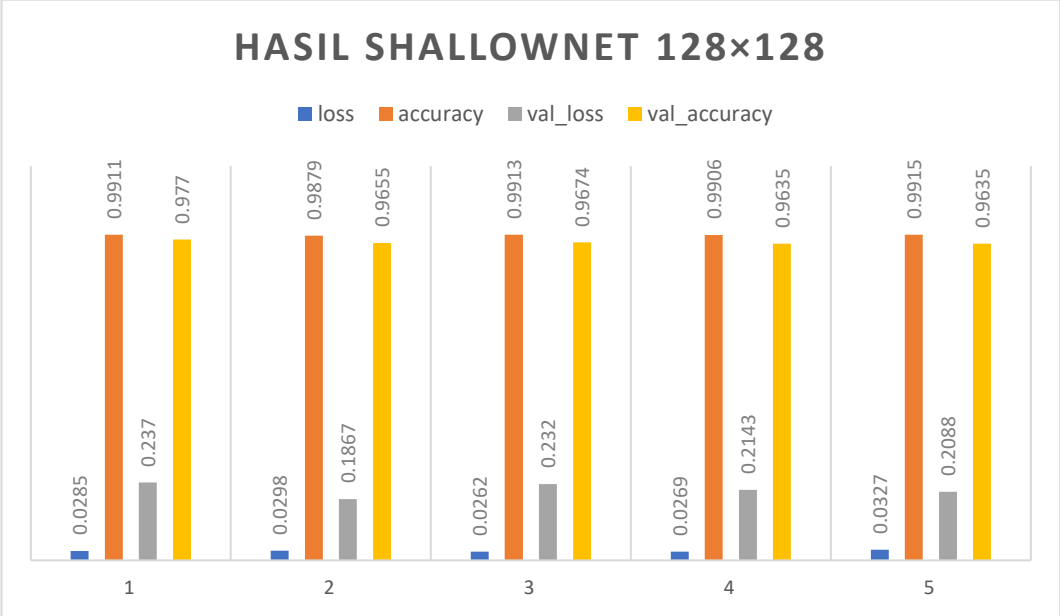
Gambar 4.1 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 1

Tabel 4.1 Hasil *Confusion Matrix* Model 1

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	382	13	7	119
2	388	7	22	104
3	367	18	2	134
4	376	4	21	120
5	371	9	11	130

4.2 ShallowNet 128×128

ShallowNet dengan *input* 128×128, epoch sebanyak 20, *optimizer* RMSprop, dan *activation* ReLU. Hasil yang didapat dari ShallowNet 128×128 adalah 0.02882 untuk rata-rata *loss*, 0.99048 untuk rata-rata *accuracy*, 0.2158 untuk rata-rata *val_loss*, dan 0.96738 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.2 dan hasil *Confusion Matrix* pada Tabel 4.2.



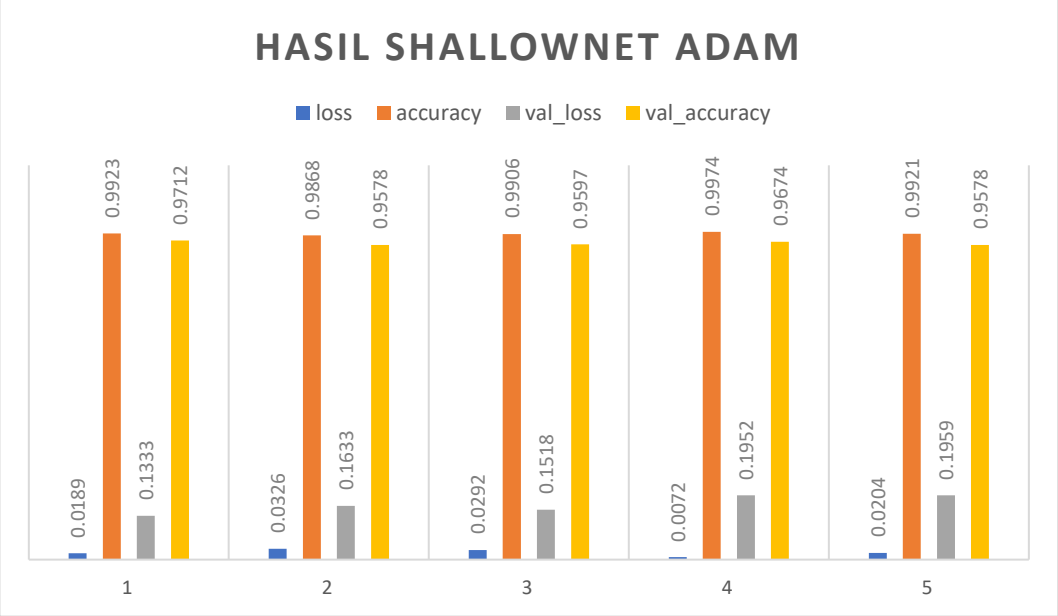
Gambar 4.2 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 2

Tabel 4.2 Hasil *Confusion Matrix* Model 2

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	378	9	3	131
2	374	13	5	129
3	374	13	4	130
4	371	16	3	131
5	375	12	7	127

4.3 ShallowNet Adam

ShallowNet dengan *input* 64×64, epoch sebanyak 20, *optimizer* Adam, dan *activation* ReLU. Model ini sama dengan model pada subbab 4.1, hanya beda pada *optimizer* saja. Hasil yang didapat dari ShallowNet Adam adalah 0.02166 untuk rata-rata *loss*, 0.99184 untuk rata-rata *accuracy*, 0.1679 untuk rata-rata *val_loss*, dan 0.96278 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.3 dan hasil *Confusion Matrix* pada Tabel 4.3.



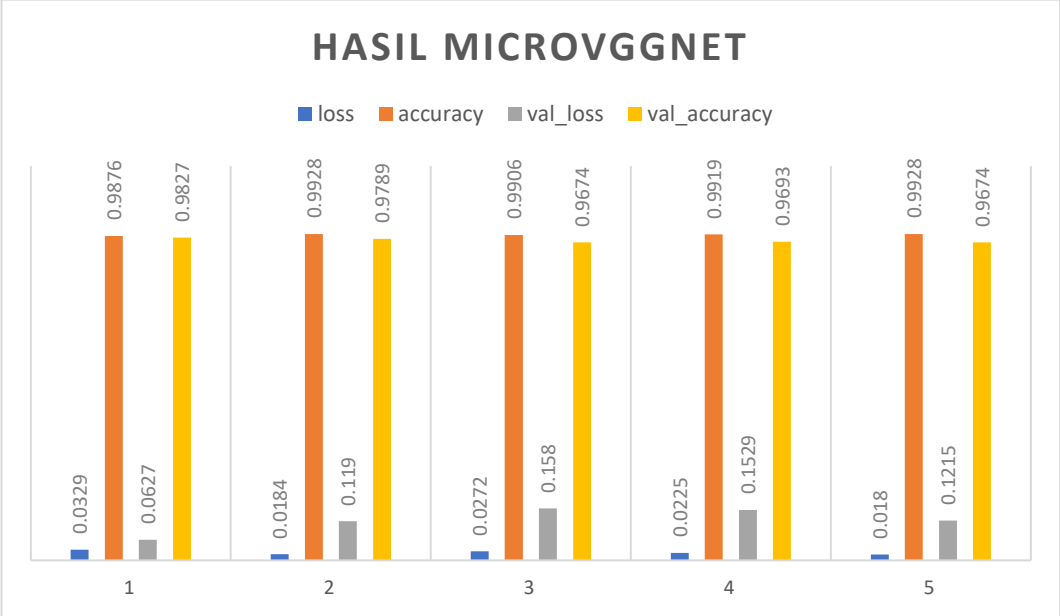
Gambar 4.3 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 3

Tabel 4.3 Hasil *Confusion Matrix* Model 3

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	371	7	8	135
2	366	12	10	133
3	360	18	3	140
4	373	5	12	131
5	369	9	13	130

4.4 MicroVGGNet

MicroVGGNet dengan *input* 64×64, epoch sebanyak 20, *optimizer* Adam, dan *activation* ReLU. Hasil yang didapat dari MicroVGGNet adalah 0.0238 untuk rata-rata *loss*, 0.99114 untuk rata-rata *accuracy*, 0.12282 untuk rata-rata *val_loss*, dan 0.97314 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.4 dan hasil *Confusion Matrix* pada Tabel 4.4.



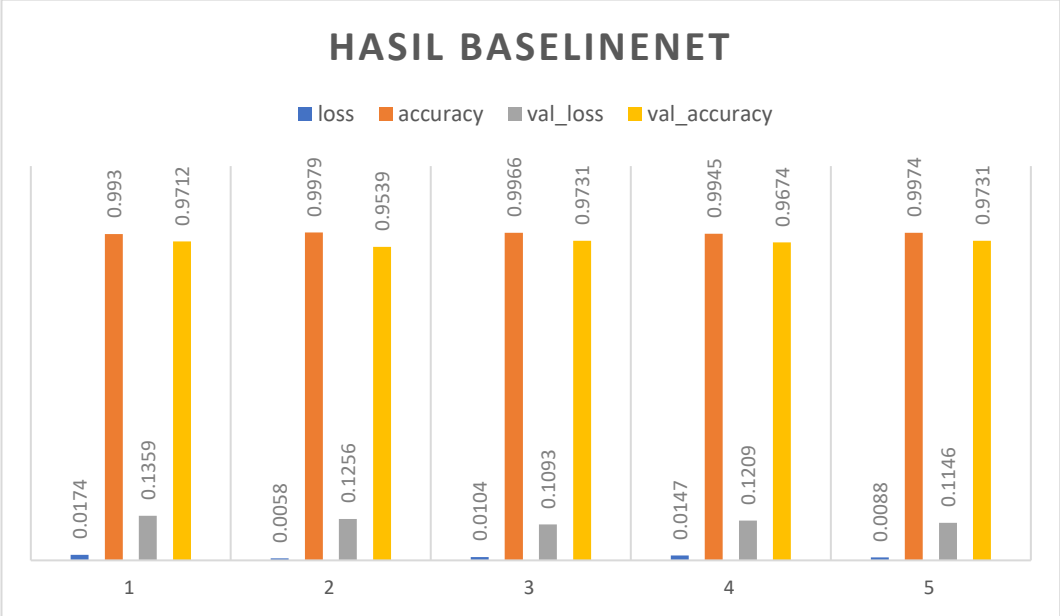
Gambar 4.4 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 4

Tabel 4.4 Hasil *Confusion Matrix* Model 4

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	381	2	7	131
2	388	7	4	122
3	389	6	11	115
4	363	11	5	142
5	367	7	10	137

4.5 BaselineNet

BaselineNet dengan *input* 64×64, epoch sebanyak 20, *optimizer* Adam, dan *activation* ReLU. Hasil yang didapat dari BaselineNet adalah 0.01142 untuk rata-rata *loss*, 0.99588 untuk rata-rata *accuracy*, 0.12126 untuk rata-rata *val_loss*, dan 0.96774 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.5 dan hasil *Confusion Matrix* pada Tabel 4.5.



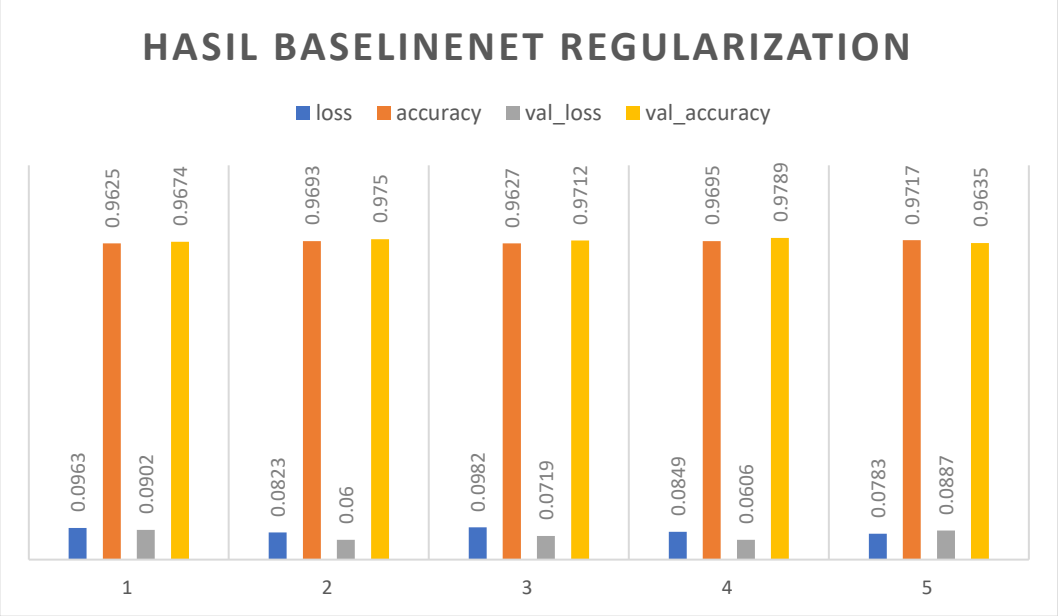
Gambar 4.5 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 5

Tabel 4.5 Hasil *Confusion Matrix* Model 5

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	381	7	8	125
2	373	15	9	124
3	379	9	5	128
4	374	14	3	130
5	378	10	4	129

4.6 BaselineNet Regularization

BaselineNet dengan *input* 64×64, epoch sebanyak 20, *optimizer* Adam, *activation* ReLU, dan *regularization* berupa Image Augmentation + Dropout. Model ini sama dengan model pada subbab 4.5, hanya ditambah dengan *regularization.* Hasil yang didapat dari BaselineNet Regularization adalah 0.088 untuk rata-rata *loss*, 0.96714 untuk rata-rata *accuracy*, 0.07428 untuk rata-rata *val_loss*, dan 0.9712 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.6 dan hasil *Confusion Matrix* pada Tabel 4.6.



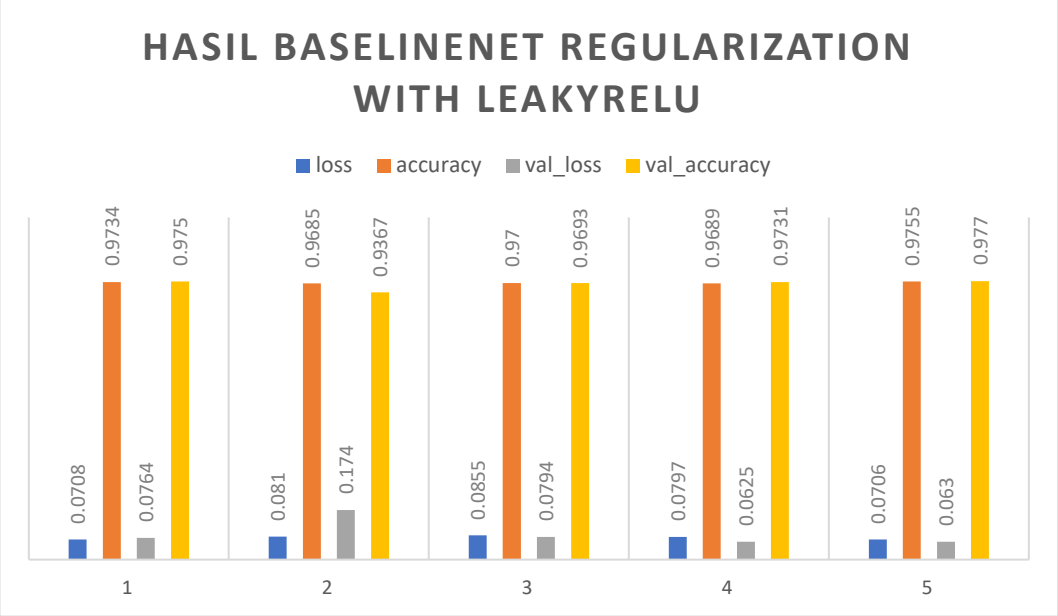
Gambar 4.6 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 6

Tabel 4.6 Hasil *Confusion Matrix* Model 6

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	370	16	1	134
2	374	12	1	134
3	373	13	2	133
4	377	9	2	133
5	369	17	2	133

4.7 BaselineNet Regularization with LeakyReLU

BaselineNet dengan *input* 64×64, epoch sebanyak 20, *optimizer* Adam, *activation* LeakyReLU, dan *regularization* berupa Image Augmentation + Dropout. Hasil yang didapat dari BaselineNet Regularization with LeakyReLU adalah 0.07752 untuk rata-rata *loss*, 0.97126 untuk rata-rata *accuracy*, 0.09106 untuk rata-rata *val_loss*, dan 0.96622 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.7 dan hasil *Confusion Matrix* pada Tabel 4.7.



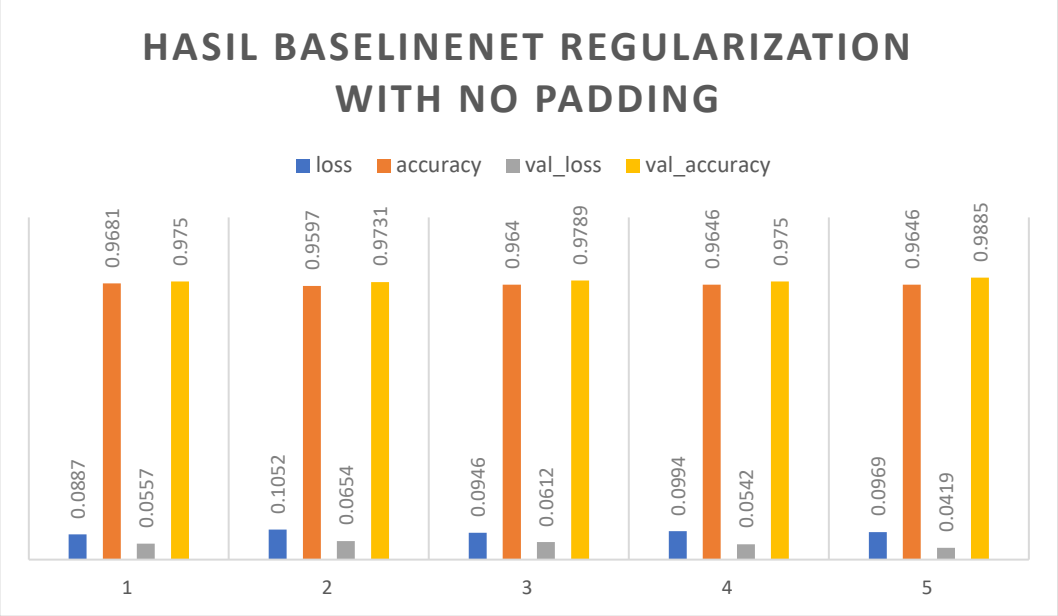
Gambar 4.7 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 7

Tabel 4.7 Hasil *Confusion Matrix* Model 7

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	382	10	3	126
2	359	33	0	129
3	378	14	2	127
4	384	8	6	123
5	387	5	7	122

4.8 BaselineNet Regularization with No Padding

BaselineNet tanpa penambahan padding dengan *input* 64×64, epoch sebanyak 20, *optimizer* Adam, *activation* ReLU, dan *regularization* berupa Image Augmentation + Dropout. Hasil yang didapat dari BaselineNet Regularization with No Padding adalah 0.09696 untuk rata-rata *loss*, 0.9642 untuk rata-rata *accuracy*, 0.05568 untuk rata-rata *val_loss*, dan 0.9781 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.8 dan hasil *Confusion Matrix* pada Tabel 4.8.



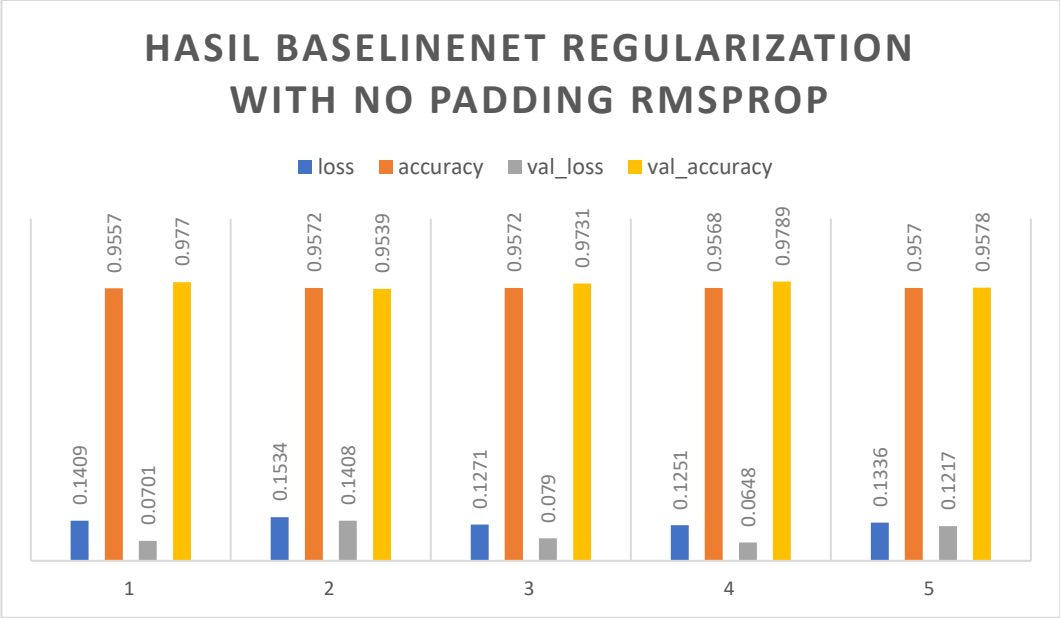
Gambar 4.8 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 8

Tabel 4.8 Hasil *Confusion Matrix* Model 8

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	382	11	2	126
2	381	12	2	126
3	389	4	7	121
4	382	11	2	126
5	390	3	3	125

4.9 BaselineNet Regularization with No Padding RMSprop

BaselineNet tanpa penambahan padding dengan *input* 64×64, epoch sebanyak 20, *optimizer* RMSprop, *activation* ReLU, dan *regularization* berupa Image Augmentation + Dropout. Model ini sama dengan model pada subbab 4.8, hanya berbeda pada *optimizer*. Hasil yang didapat dari BaselineNet Regularization with No Padding RMSprop adalah 0.13602 untuk rata-rata *loss*, 0.9567 untuk rata-rata *accuracy*, 0.09528 untuk rata-rata *val_loss*, dan 0.96814 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.9 dan hasil *Confusion Matrix* pada Tabel 4.9.



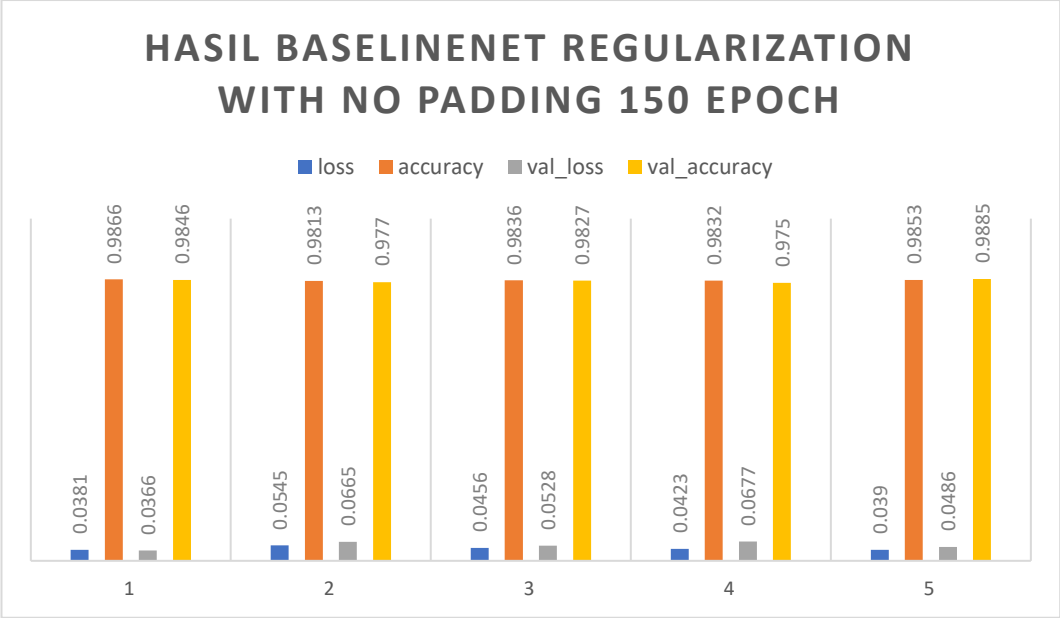
Gambar 4.9 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 9

Tabel 4.9 Hasil *Confusion Matrix* Model 9

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	385	8	4	124
2	370	23	1	127
3	380	13	1	127
4	386	7	4	124
5	371	22	0	128

4.10 BaselineNet Regularization with No Padding 150 Epoch

BaselineNet tanpa penambahan padding dengan *input* 64×64, epoch sebanyak 150, *optimizer* Adam, *activation* ReLU, dan *regularization* berupa Image Augmentation + Dropout. Model ini sama dengan model pada subbab 4.8, hanya berbeda pada banyak epoch. Hasil yang didapat dari BaselineNet Regularization with No Padding 150 Epoch adalah 0.0439 untuk rata-rata *loss*, 0.984 untuk rata-rata *accuracy*, 0.05444 untuk rata-rata *val_loss*, dan 0.98156 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.10 dan hasil *Confusion Matrix* pada Tabel 4.10.



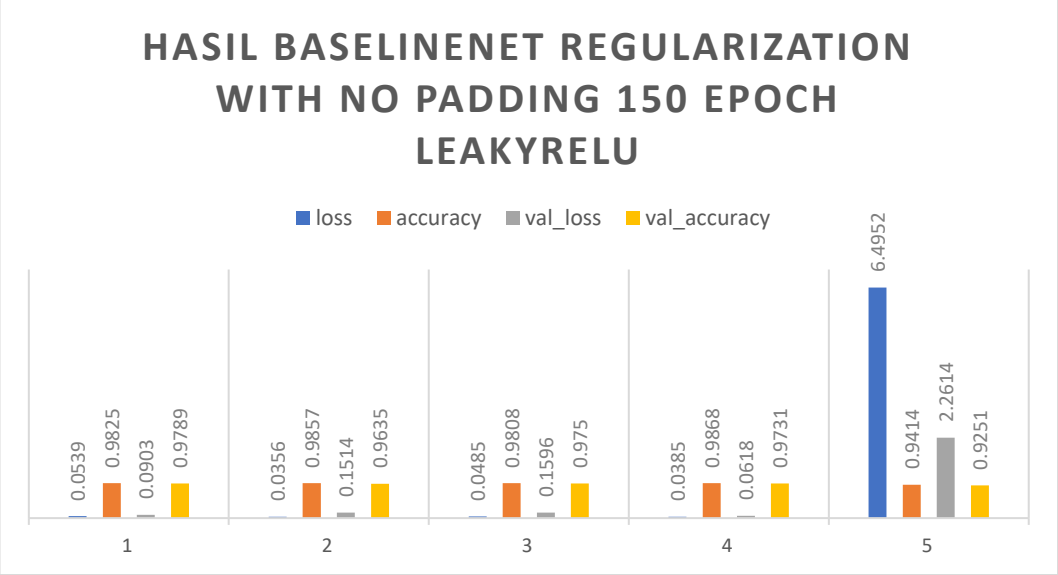
Gambar 4.10 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 10

Tabel 4.10 Hasil *Confusion Matrix* Model 10

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	390	3	5	123
2	383	10	2	126
3	386	7	2	126
4	384	9	4	124
5	389	3	3	126

4.11 BaselineNet Regularization with No Padding 150 Epoch LeakyReLU

BaselineNet tanpa penambahan padding dengan *input* 64×64, epoch sebanyak 150, *optimizer* Adam, *activation* LeakyReLU, dan *regularization* berupa Image Augmentation + Dropout. Hasil yang didapat dari BaselineNet Regularization with No Padding 150 Epoch LeakyReLU adalah 1.39842 untuk rata-rata *loss*, 0.97544 untuk rata-rata *accuracy*, 0.5449 untuk rata-rata *val_loss*, dan 0.96312 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.11 dan hasil *Confusion Matrix* pada Tabel 4.11.



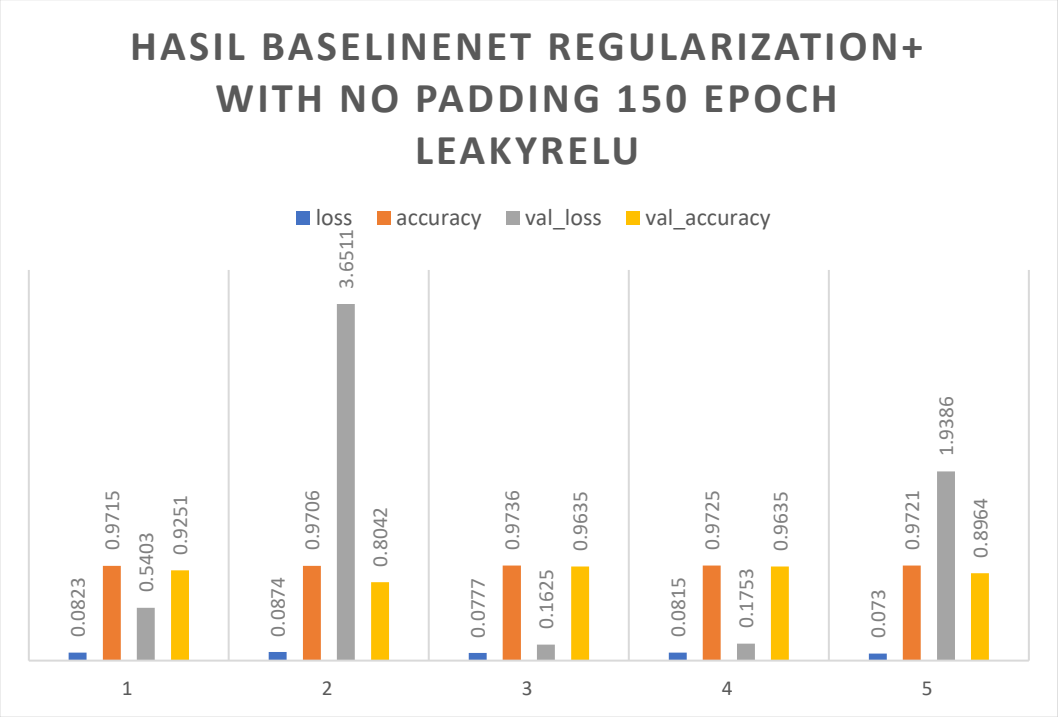
Gambar 4.11 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 11

Tabel 4.11 Hasil *Confusion Matrix* Model 11

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	374	10	1	136
2	381	4	15	121
3	372	13	0	136
4	376	9	5	131
5	347	36	3	135

4.12 BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU

BaselineNet tanpa penambahan padding dengan *input* 64×64, epoch sebanyak 150, *optimizer* Adam, *activation* LeakyReLU, dan *regularization* berupa Image Augmentation + Dropout. Model ini sama dengan model pada subbab 4.11, hanya ada menambahan *regularization*. Hasil yang didapat dari BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU adalah 0.08038 untuk rata-rata *loss*, 0.97266 untuk rata-rata *accuracy*, 1.196316 untuk rata-rata *val_loss*, dan 0.91054 untuk rata-rata *val_accuracy*. Hasil *loss*, *accuracy*, *val_loss*, dan *val_accuracy* dapat dilihat pada Gambar 4.12 dan hasil *Confusion Matrix* pada Tabel 4.12.



Gambar 4.12 Hasil *loss*, *accuracy*, *val_loss*, dan *val accuracy* model 12

Tabel 4.12 Hasil *Confusion Matrix* Model 12

Percobaan	*True Positive*	*False Positive*	*False Negative*	*True Negative*
1	391	2	37	91
2	392	1	101	27
3	386	7	12	116
4	385	8	11	117
5	390	3	51	77

4.13 Perbandingan Arsitektur Model

4.13.1 Grafik Train Loss

Dari hasil rata-rata *loss* yang sudah dipaparkan sebelumnya model yang menghasilkan nilai rata-rata *loss* terendah adalah model BaselineNet dengan nilai 0.01142, sedangkan model yang menghasilkan nilai rata-rata *loss* tertinggi adalah model BaselineNet Regularization with No Padding 150 Epoch LeakyReLU dengan nilai 1.39842.

Gambar 4.13 Hasil *train loss* 12 model

Keterangan:

Model 1: ShallowNet

Model 2: ShallowNet 128×128

Model 3: ShallowNet Adam

Model 4: MicroVGGNet

Model 5: BaselineNet

Model 6: BaselineNet Regularization

Model 7: BaselineNet Regularization with LeakyReLU

Model 8: BaselineNet Regularization with No Padding

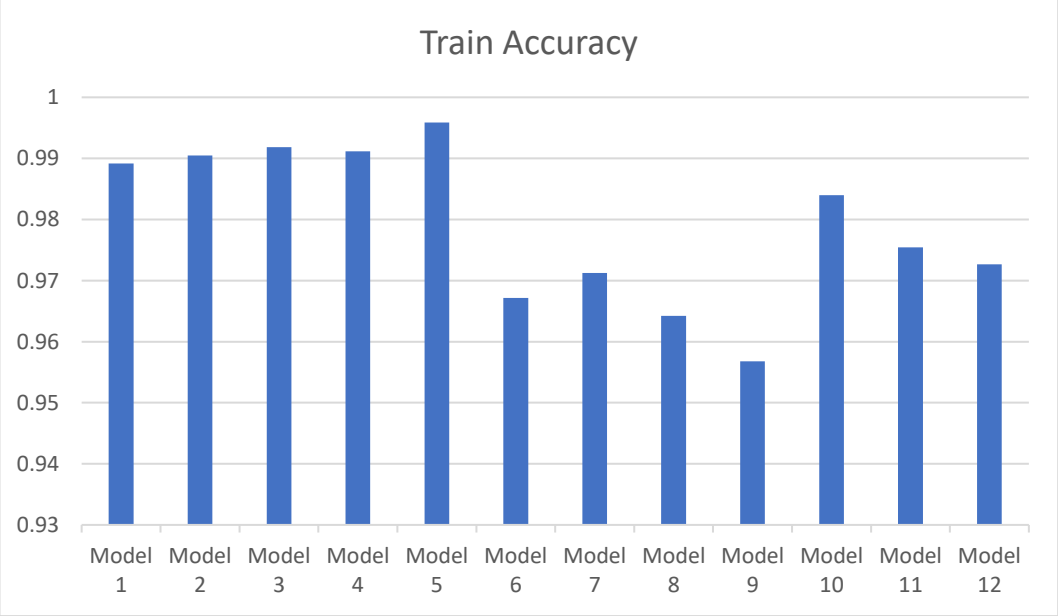
Model 9: BaselineNet Regularization with No Padding RMSprop

Model 10: BaselineNet Regularization with No Padding 150 Epoch

Model 11: BaselineNet Regularization with No Padding 150 Epoch LeakyReLU
Model 12: BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU

4.13.2 Grafik Train Accuracy

Dari hasil rata-rata *accuracy* yang sudah dipaparkan sebelumnya model yang menghasilkan nilai rata-rata *accuracy* terendah adalah model BaselineNet Regularization with No Padding RMSprop dengan nilai 0.95678, sedangkan model yang menghasilkan nilai rata-rata *accuracy* tertinggi adalah model BaselineNet dengan nilai 0.99588.



Gambar 4.14 Hasil *train accuracy* 12 model

Keterangan:
Model 1: ShallowNet
Model 2: ShallowNet 128×128
Model 3: ShallowNet Adam
Model 4: MicroVGGNet
Model 5: BaselineNet
Model 6: BaselineNet Regularization
Model 7: BaselineNet Regularization with LeakyReLU
Model 8: BaselineNet Regularization with No Padding

Model 9: BaselineNet Regularization with No Padding RMSprop

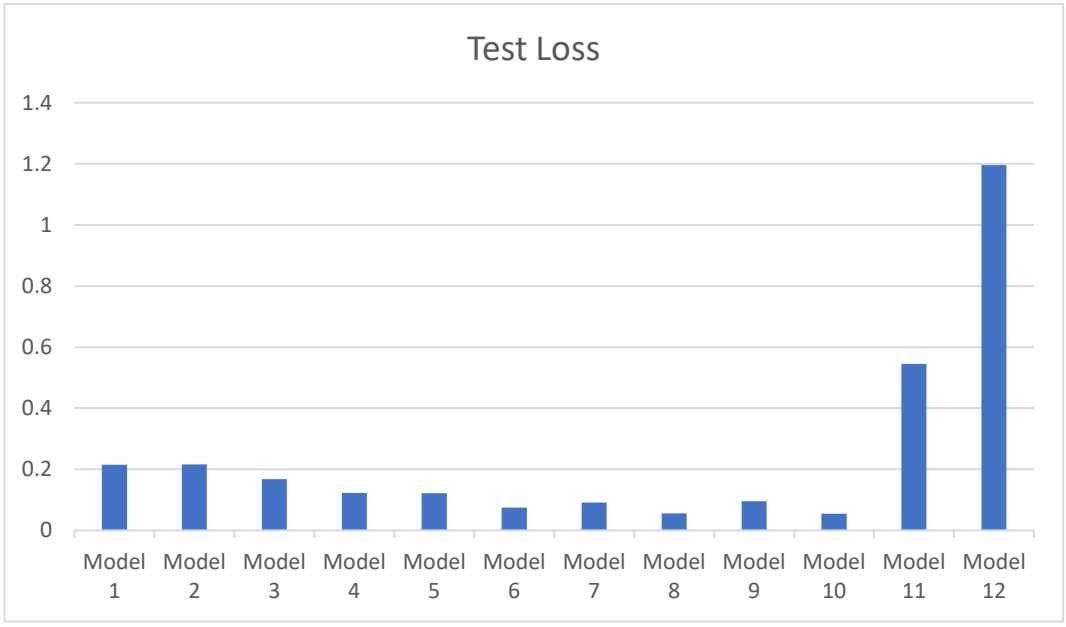
Model 10: BaselineNet Regularization with No Padding 150 Epoch

Model 11: BaselineNet Regularization with No Padding 150 Epoch LeakyReLU

Model 12: BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU

4.13.3 Grafik Test Loss

Dari hasil rata-rata *val_loss* yang sudah dipaparkan sebelumnya model yang menghasilkan nilai rata-rata *val_loss* terendah adalah model BaselineNet Regularization with No Padding 150 Epoch dengan nilai 0.05444, sedangkan model yang menghasilkan nilai rata-rata *val_loss* tertinggi adalah model BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU dengan nilai 1.196316.



Gambar 4.15 Hasil *test loss* 12 model

Keterangan:

Model 1: ShallowNet

Model 2: ShallowNet 128×128

Model 3: ShallowNet Adam

Model 4: MicroVGGNet

Model 5: BaselineNet

Model 6: BaselineNet Regularization

Model 7: BaselineNet Regularization with LeakyReLU

Model 8: BaselineNet Regularization with No Padding

Model 9: BaselineNet Regularization with No Padding RMSprop

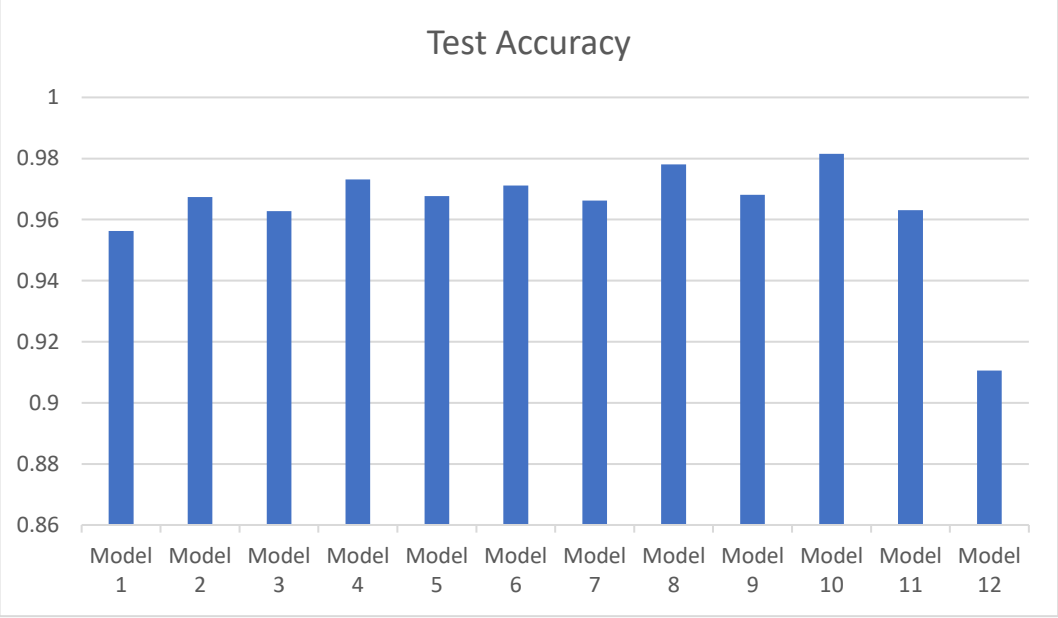
Model 10: BaselineNet Regularization with No Padding 150 Epoch

Model 11: BaselineNet Regularization with No Padding 150 Epoch LeakyReLU

Model 12: BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU

4.13.4 Grafik Test Accuracy

Dari hasil rata-rata *val_accuracy* yang sudah dipaparkan sebelumnya model yang menghasilkan nilai rata-rata *val_accuracy* terendah adalah model BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU dengan nilai 0.91054, sedangkan model yang menghasilkan nilai rata-rata *val_accuracy* tertinggi adalah model BaselineNet Regularization with No Padding 150 Epoch dengan nilai 0.98156.



Gambar 4.16 Hasil *test accuracy* 12 model

Keterangan:

Model 1: ShallowNet

Model 2: ShallowNet 128×128

Model 3: ShallowNet Adam

Model 4: MicroVGGNet

Model 5: BaselineNet

Model 6: BaselineNet Regularization

Model 7: BaselineNet Regularization with LeakyReLU

Model 8: BaselineNet Regularization with No Padding

Model 9: BaselineNet Regularization with No Padding RMSprop

Model 10: BaselineNet Regularization with No Padding 150 Epoch

Model 11: BaselineNet Regularization with No Padding 150 Epoch LeakyReLU

Model 12: BaselineNet Regularization+ with No Padding 150 Epoch LeakyReLU

4.13.5 Perbandingan Keseluruhan

Dari model yang ada dibandingkan dengan menghitung rata-rata *loss*, rata-rata *accuracy*, rata-rata *val_loss*, dan rata-rata *val_accuracy* dari 5 percobaan

Tabel 4.13 Perbandingan Hasil 12 model

Model	*average loss*	*average accuracy*	*average val_loss*	*average val_accuracy*
ShallowNet	0.04170	0.98916	0.215000	0.95622
ShallowNet 128×128	0.02882	0.99048	0.215800	0.96738
ShallowNet Adam	0.02166	0.99184	0.167900	0.96278
MicroVGGNet	0.02380	0.99114	0.122820	0.97314
BaselineNet	0.01142	0.99588	0.121260	0.96774
BaselineNet Regularization	0.08800	0.96714	0.074280	0.97120
BaselineNet Regularization with LeakyReLU	0.07752	0.97126	0.091060	0.96622
BaselineNet Regularization with No Padding	0.09696	0.96420	0.055680	0.97810
BaselineNet Regularization with No Padding RMSprop	0.13602	0.95678	0.095280	0.96814
BaselineNet Regularization with No Padding 150 Epoch	0.04390	0.98400	0.054440	0.98156
BaselinNet Regularization with No Padding 150 Epoch LeakyReLU	1.39842	0.97544	0.544900	0.96312
BaselinNet Regularization+ with No Padding 150 Epoch LeakyReLU	0.08038	0.97266	1.196316	0.91054

Dari Tabel 4.13 dapat disimpulkan bahwa rata-rata *loss* terkecil dan rata-rata *accuracy* terbesar diperoleh dari model BaselineNet. Rata-rata *val_loss* terkecil dan *val_accuracy* terbesar diperoleh dari model BaselineNet Regularization with No Padding 150 Epoch. Pada Tabel 4.13 juga, jika kedua model tersebut dibandingkan dengan yang lain maka rata-rata *loss* dan rata-rata *accuracy* dari BaselineNet Regularization with No Padding 150 Epoch cukup bagus dan rata-rata *val_loss* dan rata-rata *val_accuracy* dari model BaselineNet juga cukup bagus. Sehingga dapat disimpulkan dari keduabelas model yang ada hasil yang terbaik diperoleh model BaselineNet dan BaselineNet Regularization with No Padding 150 Epoch.

4.14 Hasil T-Test

Pengujian T-Test dilakukan untuk membandingkan perbedaan pada model yang menggunakan regularisasi dengan model yang tidak menggunakan regularisasi. Model yang dibandingkan adalah BaselineNet dengan BaselineNet Regularization. Hipotesis awal adalah H_0 = kedua model sama dan H_1 = kedua model tidak sama.

Tabel 4.14 Perbedaan Nilai Model Regularisasi dan Tanpa Regularisasi

	Rata-Rata		
	BaselineNet Tanpa Regularisasi	BaselineNet Regularisasi	Perbedaan Rata-Rata
loss	0.01142	0.08800	0.07658
accuracy	0.99588	0.96714	0.02874
val_loss	0.12126	0.07428	0.04698
val_accuracy	0.96774	0.97120	0.00346

4.14.1 T-Test Train

Tabel 4.15 Hasil T-Test *Loss Train*

			loss	
			Equal variances assumed	Equal variances not assumed
Levene's Test for Equality of Variances	F		5.181	
	Sig.		.052	
t-test for Equality of Means	t		-17.229	-17.229
	df		8	6.066
	Sig. (2-tailed)		.000	.000
	Mean Difference		-.07658	-.07658
	Std. Error Difference		.0044448	.0044448
	95% Confidence Interval of the Difference	Lower	-.0868298	-.0874275
		Upper	-.0663302	-.0657325

Tabel 4.16 Hasil T-Test *Accuracy Train*

			accuracy	
			Equal variances assumed	Equal variances not assumed
Levene's Test for Equality of Variances	F		8.349	
	Sig.		.020	
t-test for Equality of Means	t		13.595	13.595
	df		8	5.793
	Sig. (2-tailed)		.000	.000
	Mean Difference		.02874	.02874
	Std. Error Difference		.002114	.002114
	95% Confidence Interval of the Difference	Lower	.0238651	.0235221
		Upper	.0336149	.0339579

Tabel 4.15 dan

Tabel 4.16 menunjukkan nilai Sig. (2-*tailed*) 0,000 pada *loss* dan *accuracy* sehingga pada *loss* dan *accuracy* berhasil tolak H_0. Pada *loss* dan *accuracy* kedua model memiliki nilai yang berbeda.

4.14.2 T-Test Test

Tabel 4.17 Hasil T-Test *Loss Test*

			loss	
			Equal variances assumed	Equal variances not assumed
Levene's Test for Equality of Variances	F		1.625	
	Sig.		.238	
t-test for Equality of Means	t		5.875	5.875
	df		8	7.162
	Sig. (2-tailed)		.000	.001
	Mean Difference		.04698	.04968
	Std. Error Difference		.0079962	.0079962
	95% Confidence Interval of the Difference	Lower	.0285406	.0281583
		Upper	.0654194	.0658017

Tabel 4.18 Hasil T-Test *Accuracy Test*

			accuracy	
			Equal variances assumed	Equal variances not assumed
Levene's Test for Equality of Variances	F		.162	
	Sig.		.698	
t-test for Equality of Means	t		-.766	-.766
	df		8	7.426
	Sig. (2-tailed)		.466	.468
	Mean Difference		-.00346	-.00346
	Std. Error Difference		.0045197	.0045197
	95% Confidence Interval of the Difference	Lower	-.0138824	-.0140244
		Upper	.0069624	.0071044

Tabel 4.17 dan Tabel 4.18 menunjukkan nilai Sig. (2-*tailed*) 0,000 pada *val_loss* dan 0,466 pada *val_accuracy* sehingga pada *val_loss* berhasil tolak H_0 dan *val_accuracy* gagal tolak H_0. Jadi, kedua model memiliki nilai *val_loss* yang berbeda dan nilai *val_accuracy* yang sama.

Bab V
Penutup

5.1 Simpulan

Dari hasil dari uji T (T-Test) pada bab sebelumnya, yaitu membandingkan model yang menggunakan regularisasi (BaselineNet *Regularization*) dengan model yang tidak menggunakan regularisasi (BaselineNet) hasil yang didapat adalah sebagai berikut. Perbedaan rata-rata *loss* pada kedua model sebesar 0.07658 dan nilai yang lebih baik atau lebih kecil diperoleh model dengan tanpa regularisasi. Perbedaan rata-rata *accuracy* pada kedua model sebesar 0.02874 dan nilai yang lebih baik atau lebih besar diperoleh model dengan tanpa regularisasi. Perbedaan rata-rata *val_loss* pada kedua model sebesar 0.04698 dan nilai yang lebih baik atau lebih kecil diperoleh model dengan regularisasi. Perbedaan rata-rata *val_accuracy* pada kedua model sebesar 0.00346 dan nilai yang lebih baik atau lebih besar diperoleh model dengan regularisasi.

Pada hasil membandingkan kedua model di atas, dapat disimpulkan bahwa regularisasi memberikan pengaruh signifikan pada *loss*, *accuracy*, dan *val_loss*, sedangkan pada *val_accuracy* tidak memberikan pengaruh. Dari kedua model tersebut dapat disimpulkan model BaselineNet (tanpa regularisasi) lebih baik dari pada model BaselineNet Regularisasi dikarenakan pada *loss* dan *accuracy* terdapat perbedaan yang signifikan dan hasil yang lebih baik diperoleh model BaselineNet (tanpa regularisasi).

5.2 Saran

Saran untuk penelitian selanjutnya, peneliti bisa menerapkan regularisasi yang berbeda seperti L1 regularization atau L2 regularization, menggunakan dataset yang berbeda, atau menambah arsitektur model lainnya untuk digunakan sebagai pembanding.

DAFTAR PUSTAKA

Adam, R. 2019, 'Contoh Perhitungan Algoritma Backpropagation - Structilmy', diakses pada 26 November 2020, <https://structilmy.com/2019/07/contoh-perhitungan-algoritma-backpropagation/>

Anggriawan, F. 2016, 'Apakah Pneumonia Dapat Menular?', diakses pada 8 Agustus 2020, <https://lifestyle.okezone.com/read/2016/09/13/481/1488504/apakah-pneumonia-dapat-menular>

Baron, M. 2014, 'Probability And Statistics For Computer Scientists', 2nd ed. Boca Raton: CRC Press.

Chen, J. 2020, 'Neural Network Definition', diakses pada 6 Desember 2020 <https://www.investopedia.com/terms/n/neuralnetwork.asp>

Dickson, B. 2020, 'What is artificial narrow intelligence (Narrow AI)? – TechTalks', diakses pada 6 Desember 2020 <https://bdtechtalks.com/2020/04/09/what-is-narrow-artificial-intelligence-ani/>

Fahriz, A. 2019, 'Mengenal Jenis Pembelajaran Mesin Supervised Learning dan Unsupervised Learning', diakses pada 7 September 2020 <https://medium.com/kelompok1/mengenal-jenis-pembelajaran-mesin-supervised-learning-dan-unsupervised-learning-c588881e8ef5>

Gylberth, R. 2018, 'Apa itu regularisasi dalam pembelajaran mesin? - Qoura', diakses pada 8 September 2020 <https://id.quora.com/Apa-itu-regularisasi-dalam-pembelajaran-mesin>

Kusaribe, H. 2020, 'Mengenal Lebih Dekat Dengan Uji T sebagai Metode Pengujian Statistika | Metode Penelitian', diakses pada 19 Januari 2021 <https://metodepenelitian.id/apa-itu-uji-t/>

Nagpal, A. 2017, 'L1 and L2 Regularization Methods. Machine Learning | by Anuja Nagpal | Towards Data Science', diakses pada 27 November 2020 <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>

Pane, M. D. C. 2020, 'Pneumonia', diakses pada 7 Agustus 2020 <https://www.alodokter.com/pneumonia>

Pratama, A. N. 2018, 'Hari Ini dalam Sejarah, Rontgen Temukan Teknologi Sinar-X', diakses pada 9 Agustus 2020 <https://internasional.kompas.com/read/2018/11/08/11252491/hari-ini-dalam-sejarah-rontgen-temukan-teknologi-sinar-x?page=all#page2>

PT Dewaweb. 2018, *Kecerdasan Buatan: Perkembangan dan Dampak*, diakses pada 7 September 2020 <https://www.dewaweb.com/blog/kecerdasan-buatan/>

Schirrmeister, R. T. et al., 2017. Deep Learning with Convolutional Neural Network for EEG Decoding And Visualization. *Human Brain Mapping*, **38**(11).

Sena, S. 2017, 'Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN) | by Samuel Sena | Medium', diakses pada 27 November 2020 <https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94>

Sidiq, R. F. 2020, 'Belajar Program Python Pemula', diakses pada 7 September 2020 <https://tugasmahasiswa.com/belajar-program-python-pemul>

Simonyan, K. dan Zisserman, A. 2015, *Very Deep Convolutional Networks For Large-Scale Image Recognition.* San Diego, International Conference on Learning Representations.

Singh, A. 2018, 'Artificial Neural Network | Types | Feed Forward | Feedback | Structure | Perceptron | Machine Learning | Application | Tech Blog', diakses pada 24 November 2020 <https://msatechnosoft.in/blog/artificial-neural-network-types-feed-forward-feedback-structure-perceptron-machine-learning-applications/>

Widiputra, H. D. 2016, 'Artificial Neural Network', diakses pada 4 September 2020 <https://dosen.perbanas.id/artificial-neural-network/>

Yi, S., Xiaogang, W. dan Xiaoou, T. 2016,. *Sparsifying Neural Network Connections for Face Recognition.* Las Vegas, Computer Vision and Pattern Recognition.