Klasifikasi Alzheimer Empat Kelas Berdasarkan Data MRI Otak Menggunakan Model DenseNet169, ResNet50, dan VGG16

Nama: Pandega Abyan Zumarsyah
NIM : 18/424977/TK/46672

Fakultas Teknik
Universitas Gadjah Mada
2021

Intisari

Penggunaan *deep learning* untuk analisis citra medis makin meningkat, terutama untuk citra MRI otak. *Deep learning* mampu melakukan klasifikasi pada citra tersebut sehingga membantu diagnosis untuk suatu penyakit tertentu, termasuk Alzheimer.

Ada berbagai penelitian yang menggunakan *deep learning* untuk klasifikasi Alzheimer. Dalam pengujian ini, dilakukan klasifikasi Alzheimer empat kelas, yaitu *Moderate Demented*, *Mild Demented*, *Very Mild Demented*, dan *Nondemented* atau normal. Data untuk pengujian ini didapatkan dari kaggle. Data tersebut sudah cukup banyak tetapi sangat tidak seimbang sehingga perlu augmentasi, terutama untuk kelas dengan data yang masih sangat sedikit.

Terdapat tiga model yang diuji dan dibandingkan, yaitu DenseNet169, ResNet50, dan VGG16. Pengujian menunjukan bahwa ketiga model sudah memberikan performa yang cukup baik dalam hal akurasi dan nilai AUC dengan nilai akurasi dari ketiganya berturut-turut sebesar 97,16%, 94,67%, 98,89%. Secara umum, model VGG16 memberikan performa yang terbaik dari berbagai sisi. Namun, dibanding dua model lainnya, waktu *training* model VGG16 tiga kali lebih lama sehingga dapat menjadi pertimbangan tersendiri.

I. PENDAHULUAN

A. Latar Belakang

Dalam beberapa waktu terakhir, penggunaan *deep learning* untuk analisis citra medis makin meningkat, terutama untuk citra MRI otak. Bahkan, untuk periode 2014 – 2019, publikasi dengan kata kunci "deep learning" dan "MRI" meningkat secara eksponensial [1].

MRI atau *magnetic resonance imaging* merupakan salah satu teknik observasi tubuh yang cukup populer karena noninvasif. MRI otak sendiri merupakan cara yang sangat baik untuk observasi status dan struktur otak serta mendeteksi adanya kelainan pada otak. [1] Umumnya, interpretasi dan analisis citra medis seperti MRI dilakukan oleh ahli yang berpengalaman. Namun, itu membutuhkan waktu serta usaha yang tidak sedikit. Interpretasi citra MRI secara otomatis melalui *deep learning* menjadi solusi untuk masalah tersebut. [2] Berkaitan dengan hal ini, *deep learning* dapat digunakan untuk membantu diagnosis Alzheimer.

Alzheimer merupakan kelainan neurologis pada otak yang menyebabkan kerusakan permanen pada sel otak. Itu dimulai dengan deteriorasi sel otak yang sedikit demi sedikit mengarah pada demensia. Bagian korteks dari pasien makin mengerut dan terdapat kerusakan besar pada bagian *hippocampus*. Bagian itu berperan penting dalam proses berpikir dan mengingat. Karena kerusakan itu, pasiennya pun tidak bisa menjalankan aktivitas sehari-hari secara normal. [3] Jika dibiarkan, kerusakannya akan menjalar ke bagian otak yang lebih vital sehingga bisa mengarah pada kematian.

Mengingat penyakit ini sangat berbahaya, bahkan menjadi penyebab kematian nomor enam di Amerika Serikat, diagnosis dini tentu sangat diperlukan. [4] Diagnosis dini ini penting untuk memberikan perawatan yang sesuai sebelum kondisinya makin parah. [5] Namun, itu tidaklah mudah karena membutuhkan ahli yang berpengalaman dan waktu serta usaha yang tidak sedikit. *Deep learning* dapat menjadi solusi untuk masalah itu.

Dalam pengujian ini, beberapa model *deep learning* digunakan untuk melakukan klasifikasi pada citra MRI yang berkaitan dengan penyakit Alzheimer. Klasifikasinya dilakukan untuk empat kelas: (*Moderate Demented*, *Mild Demented*, *Very Mild Demented*, dan *Nondemented* atau normal). Model *deep learning* yang digunakan ada tiga: DenseNet169, ResNet50, dan VGG16.

Dataset untuk pengujian ini didapatkan dari kaggle [6]. Di kaggle itu, sebenarnya sudah ada banyak kode pengujian untuk dataset yang sama dengan berbagai metode. Sebagian kode yang ada masih belum melakukan augmentasi data dan hanya fokus kepada satu model. Kami pun menjadikan beberapa kode sebagai referensi untuk membuat kode yang harapannya bisa lebih baik lagi. Di antaranya dengan meningkatkan augmentasi dan pengolahan data di awal serta dengan membandingkan beberapa model sekaligus. Dengan semua itu, diharapkan hasil dari pengujian ini dapat berkontribusi dalam riset di bidang *deep learning* untuk keperluan medis atau pun berbagai bidang lainnya.

B. Tujuan

1. Menggunakan berbagai model *deep learning* untuk melakukan klasifikasi citra
2. Menghasilkan model berkualitas tinggi untuk klasifikasi Alzheimer empat kelas berdasarkan citra MRI
3. Membandingkan berbagai model dan metode dalam *deep learning* untuk klasifikasi citra

II. KAJIAN PUSTAKA

A. Proses Klasifikasi dan Parameternya

Sederhananya, proses klasifikasi citra merupakan pemberian label ke suatu citra berdasarkan fitur yang ada dalam citra itu. Dalam klasifikasi citra, inputnya berupa citra sementara outputnya berupa keterangan citra tersebut terklasifikasikan ke mana. [7]

Proses klasifikasi citra ini berkaitan dapat sangat membantu diagnosis dalam medis. Itu karena sebagian diagnosis pada dasarnya hanyalah memberikan label berdasarkan citra medis yang didapatkan. Dalam hal ini, inputnya berupa citra medis sementara outputnya berupa keterangan apakah ada kelainan atau tidak. [7]

Dalam *data science*, terdapat beberapa parameter penting yang digunakan juga dalam analisis performa suatu proses klasifikasi. Parameter itu berhubungan dengan istilah-istilah *true positive* (TP), *false positive* (FP), *true negative* (TN), dan *false negative* (FN).

Dalam klasifikasi, akurasi berhubungan dengan rasio prediksi yang benar terhadap total:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Akurasi adalah parameter ukur yang baik, namun tidak bisa berdiri sendiri. Ada parameter presisi yang merupakan rasio TP terhadap total prediksi positif: [8]

$$Precision = \frac{TP}{TP + FP}$$

Ada pula recall yang merupakan rasio TP terhadap total yang benar-benar positif: [8]

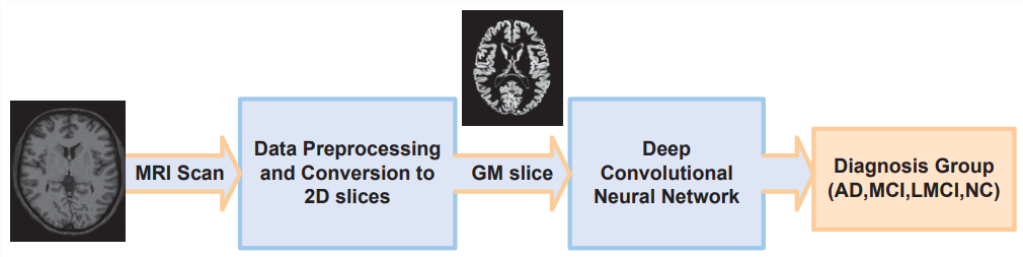
$$Recall = \frac{TP}{TP + FN}$$

Kemudian ada *loss function* yang berhubungan dengan *error*/ perbedaan antara output model dengan yang diharapkan. Ini digunakan di berbagai masalah optimisasi dengan tujuan meminimalkan *loss* nya. Dengan *loss* yang minimal, diharapkan akurasinya bisa maksimal. [9]

Yang terakhir yang digunakan di sini adalah AUC (*Area Under the Curve*). Ini menunjukan seberapa baik modelnya dalam memisahkan setiap kelas. Jika AUC mendekati 0, berarti sangat buruk, bahkan justru berkebalikan dengan yang diharapkan. Jika AUC = 0,5 berarti tidak ada pemisahan kelas yang jelas. Jika AUC mendekati 1, berarti makin baik. [10]

B. Klasifikasi Alzheimer Empat Kelas (Berhubungan dengan MCI)

Dalam suatu penelitian, *deep learning* digunakan untuk klasifikasi Alzheimer dan MCI (*mild cognitive impairment*) menjadi empat kelas: Alzheimer (AD), MCI, late MCI (LMCI), dan orang normal (NC). Ilustrasi alurnya dapat dilihat pada Gambar 1. [3]

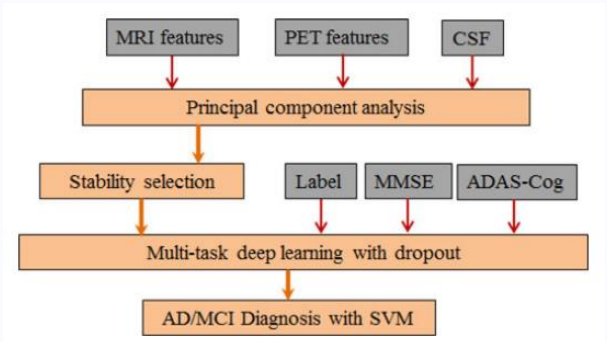


Gambar 1: Alur dalam Klasifikasi Alzheimer Empat Kelas [3]

Bagian *preprocessing* diperlukan karena citra MRI awalnya berbentuk 3D Nifti. Maka, dilakukanlah *slicing* untuk mendapat citra-citra aksial. Irisan yang kurang berguna lalu dibuang. Hasil dari ini adalah irisan-irisan *gray matter* (GM) dari citra otak. Kemudian, irisan GM masuk ke jaringan neural dengan arsitektur CNN. Model yang diuji di sini adalah Googlenet, Resnet-18, dan Resnet-152. [3]

Dataset yang dipakai dalam penelitian itu didapatkan dari *Alzheimer Disease Neuroimaging Initivative* (ADNI) dengan total 355 volume dan 149 subjek. Hasilnya, ketiga model menunjukan performa yang lebih baik daripada penelitian-penelitian serupa mengenai klasifikasi multikelas untuk Alzheimer. Secara keseluruhan, model Googlenet, Resnet-18, dan Resnet-152 berturut-turut memberikan akurasi 99%, 98%, dan 98%. Selain akurasi, parameter seperti presisi, sensitivitas, dan spesifisitas juga dipakai. Semuanya menunjukan angka yang tinggi, mendekati 100%. [3]

Dalam suatu penelitian lain, digunakan dataset dengan jenis yang sedikit berbeda, yaitu data pasien AD, MCI, MCI yang kemudian berkembang menjadi AD (MCI.C), MCI yang kemudian tidak berkembang menjadi AD (MCI.NC), dan normal (NC). Klasifikasi dilakukan empat kali: AD versus NC, MCI versus NC, AD versus MCI, dan MCI.C versus MCI.NC. Ilustrasi alurnya dapat dilihat pada Gambar 2. [4] Perlu diperhatikan bahwa dataset yang digunakan di sini relatif lebih sedikit daripada penelitian sebelumnya.



Gambar 2: Alur dalam Klasifikasi Alzheimer melalui *Deep Learning* dengan *Dropout* [4]

Selain data MRI, digunakan pula data PET dan CSF. *Principal component analysis* merupakan transformasi linear ortogonal yang mengubah fitur-fitur menjadi variabel-variabel yang *linearly uncorrelated*. Ini sering digunakan untuk mengurangi dimensi dari data. [4]

Langkah yang sangat penting dalam penelitian itu adalah dipakainya *dropout* dalam *deep learning*. *Dropout* dilakukan untuk meningkatkan kemampuan generalisasi ketika data yang ada tidak terlalu banyak. Itu dilakukan dengan mengeluarkan beberapa unit secara acak ketika *training*. Dengan begitu, model yang dibuat tidak hanya sesuai untuk kasus yang mirip data *training*, tetapi bisa lebih umum. [4]

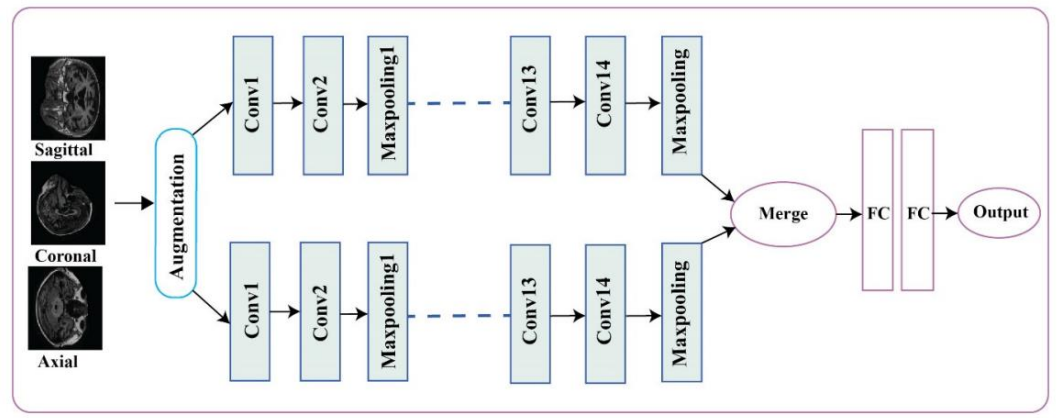
Hasil dari penelitian itu adalah akurasi sebesar: 91% untuk AD versus NC, 77% untuk MCI versus NC, 70% untuk AD vs MCI, dan 57% untuk MCI.C versus MCI.NC. Penelitian itu juga mendapatkan kesimpulan bahwa teknik *dropout* sangat berguna untuk meningkatan performa, dengan kenaikan akurasi rata-rata sebesar 5,9%. [4]

C. Klasifikasi Alzheimer Empat Kelas (Berhubungan dengan Demensia)

Sebelumnya adalah penelitian mengenai klasifikasi Alzheimer empat kelas, namun yang berhubungan dengan kasus MCI. Dalam pengujian yang dilakukan, Alzheimer empat kelasnya berhubungan dnegan demensia. Maka, penelitian tentang itu perlu ditinjau.

Ada suatu penelitian dengan empat kelas yang sama dengan pengujian (*Moderate Demented*, *Mild Demented*, *Very Mild Demented*, dan *Nondemented* atau normal), namun dengan sumber data yang berbeda. Penelitian itu menggunakan dataset dari OASIS dengan total citra 382 buah. Terdapat *preprocessing* yang meliputi proses *enhancement* citra dan augmentasi data. Setelah dilakukan *preprocessing*, banyak datanya menjadi 3820 buah. [5]

Model yang digunakan adalah Siamese CNN (SCNN) dengan memodifikasi model VGG16 [5]. Ilustrasi untuk model itu dapat dilihat pada Gambar 3.



Gambar 3: Model SCNN untuk Klasifikasi Alzheimer Empat Kelas [5]

Di sini, dua model VGG16 yang telah dimodifikasi bekerja secara paralel. Tiap modelnya memiliki 14 layer konvolusi, layer *max-pooling*, *normalization*, dan *Gaussian noise*. Konfigurasi paralel ini diharapkan dapat mengekstrak lebih banyak fitur yang kemudian disatukan. Ada tiga normalisasi yang diuji: *batch normalization, switch normalization,* dan *group normalization*. [5]

Hasilnya, akurasi yang didapatkan untuk 20 epochs adalah sebesar 99,05%. *Batch normalization* memberikan hasil yang lebih baik dibanding dua normalisasi lain yang diuji. [5] Hasil akhir di penelitian ini cenderung lebih tinggi daripada beberapa penelitian serupa yang telah dilakukan. Hasil dari penelitian lain yang cukup serupa itu dapat dilihat pada Gambar 4.

Paper	Method	Dataset	Accuracy
Islam et al. [53]	ResNet, CNN	OASIS (MRI)	93.18%
Hosseini et al. [54]	3D-DSA-CNN	ADNI (MRI)	97.60%
Evign et al. [55]	3D-CNN	ADNI (MRI)	98.01%
Farooq et al. [56]	GoogLeNet	OASIS (MRI)	98.88%
Khan et al. [40]	VGG	ADNI (MRI)	99.36%

Gambar 4: Beberapa Hasil Penelitian Klasifikasi Alzheimer Empat Kelas [5]

Berbagai pengujian klasifikasi Alzheimer yang berhubungan dengan demensia juga bisa didapatkan di kaggle. Pengujian-pengujian itu sudah terhubung langsung dengan dataset di kaggle yang juga akan kami gunakan [6]. Meski datanya sangat tidak seimbang, beberapa pengujian yang ada tidak melakukan augmentasi untuk menyeimbangkan datanya. Selain itu, sebagian pengujian itu juga hanya fokus pada satu model dengan hasil yang tidak terlalu bagus. Padahal, dataset yang ada di kaggle cukup banyak, dengan total sekitar 4000 citra. Bagaimana pun juga, berbagai pengujian itu tetap bisa dijadikan sebagai referensi.

Di antara pengujian yang hasilnya sangat baik sehingga kami jadikan referensi utama adalah pengujian oleh Bryce Smith dan koleganya [11]. Pengujian ini menggunakan model VGG16 dengan sedikit layer tambahan. Hasilnya, akurasi yang didapatkan untuk 50 epochs adalah sebesar 99,52%. Padahal, pengujian ini menggunakan dataset yang sangat tidak seimbang dan belum diseimbangkan.

III. METODOLOGI

A. Alat dan Bahan

Dataset yang digunakan dalam pengujian ini didapatkan dari kaggle [6]. Dataset itu memuat sekitar 4000 citra yang terdiri atas empat kelas (*Moderate Demented*, *Mild Demented*, *Very Mild Demented*, dan *Nondemented* atau normal). Namun, datanya sangat tidak seimbang. Kelas *nondemented* memuat lebih dari 2000 citra sementara kelas *moderate demented* memuat kurang dari 100 citra. Karenanya, perlu ada augmentasi untuk kelas yang datanya masih sedikit.

Pada dataset itu, sudah ada pembagian "training" dan "test". Namun, menurut [11], pembagian yang ada itu kurang uniform. Ketika kami mencoba menguji menggunakan pembagian itu, hasilnya juga kurang baik. Karenanya, perlu dilakukan pembagian ulang.

Pengujian dilakukan di Google Colab agar dapat memanfaatkan sumber daya dari Google, seperti RAM, Disk, dan bahkan GPU. Adanya GPU sangatlah penting karena dapat meningkatkan kecepatan ketika *training* puluhan kali lipat dibanding CPU.

Ada berbagai fungsi dari modul-modul python yang digunakan dalam pengujian ini. Yang utama tentu fungsi-fungsi dari modul tensorflow. Di antaranya adalah fungsi ImageDataGenerator() dan fungsi untuk optimizer. Dari tensorflow, diimpor pula beberapa layer seperti Dense, Flatten, dan Dropout. Model DenseNet169, ResNet50, dan VGG16 yang digunakan dalam pengujian juga diimpor dari tensorflow. Selain itu, dari modul lain, ada fungsi-fungsi yang diimpor untuk mengelola file dan menampilkan hasil akhir. Berbagai modul itu memang sangat memudahkan proses pengujian ini.

B. Metode Umum

Pengujian ini memanfaatkan berbagai referensi yang ada untuk membuat pengujian yang lebih baik. Referensi itu dapat berupa kode dari orang lain atau konsep-konsep seputar *deep learning*. Sebagaimana telah disampaikan, referensi utamanya adalah pengujian oleh Bryce Smith dan koleganya [11]. Selain itu, tentu ada banyak referensi lainnya.

Model yang digunakan oleh Smith adalah VGG16. Namun, kami mencoba menggunakan model lain sebagai pembanding, yaitu DenseNet169 dan ResNet50. Smith tidak melakukan augmentasi data untuk menyeimbangkan dataset nya. Namun, di sini, kami akan menyeimbangkan dataset nya.

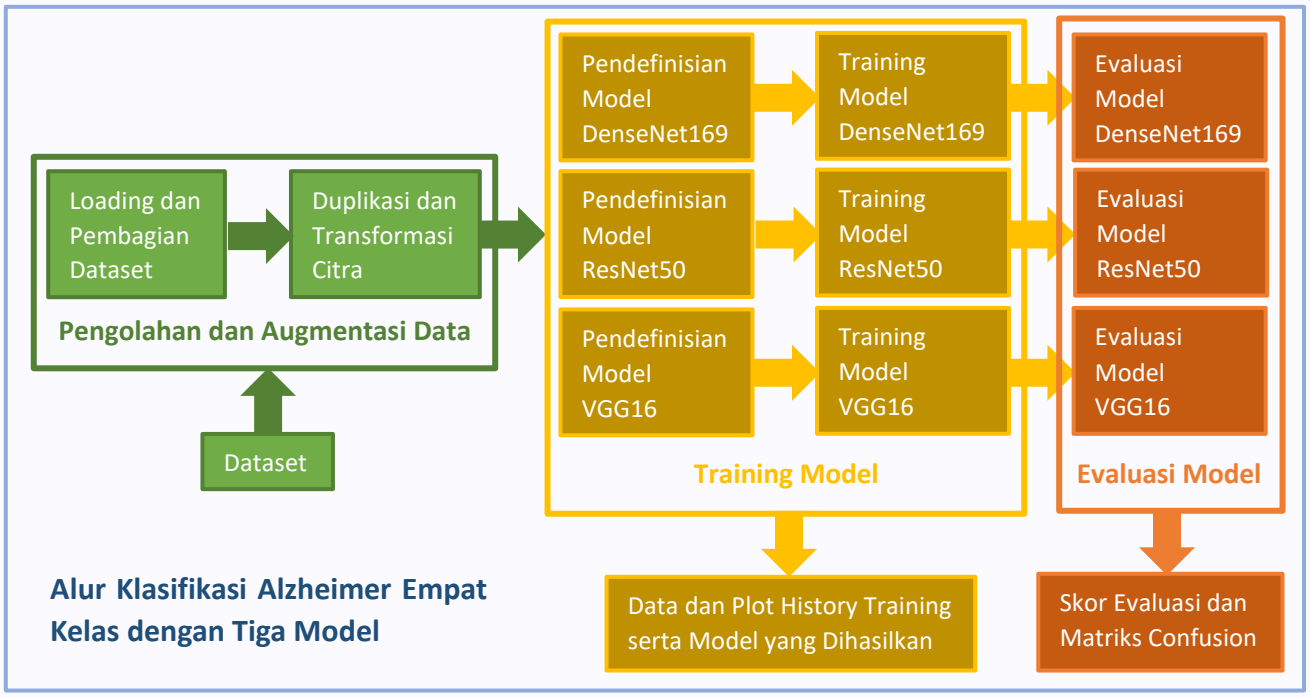
Secara umum, alur dalam pengujiannya dapat dilihat pada Gambar 5. Awalnya, terdapat loading dan pembagian ulang dataset. Data dibagi menjadi "train", "val", dan "test" dengan rasio 70:15:15. Setelah itu, terdapat proses augmentasi data untuk menyeimbangkan dataset nya.

Kemudian, terdapat pendefinisian untuk modelnya, meliputi impor model dari tensorflow serta penambahan layer dan optimizer. Kami sudah menguji beberapa kombinasi layer tambahan sehingga layer-layer yang digunakan di hasil akhir memang sudah yang cukup baik.

Setelah itu, data yang sudah diolah tadi masuk ke *training*. Output dari *training* ini adalah data dan plot *training history* serta model akhir yang didapatkan. Selanjutnya, terdapat evaluasi yang outputnya adalah skor dan matriks *confusion*.

Pengujian di Google Colab ini dihubungkan dengan Google Drive. File dataset sejak awal sudah diletakan di Google Drive. Berbagai output juga dijadikan file dan kemudian disimpan di Google Drive.

Dalam pengujian ini, digunakan pula dataset kecil untuk *trial-error* yang membutuhkan waktu singkat. Dataset kecil itu diambil dari dataset utamanya. Agar lebih cepat, *trial-error* juga menggunakan batch dan epoch yang rendah. Bagaimana pun juga, *trial-error* ini penting untuk menguji berbagai parameter sehingga hasil akhirnya dapat lebih optimal.



Gambar 5: Alur Klasifikasi yang Digunakan

C. Langkah Pengujian

Pembahasan di sini mencakup hal-hal yang sifatnya lebih teknis dibanding sebelumnya, berupa berbagai hal yang dilakukan ketika pengujian.

Dalam pengujian, langkah awalnya tentu memasukan datanya. Karena dataset nya ada di Google Drive, maka perlu *mounting* terlebih dahulu. Dataset nya dimasukan ke Colab lalu diekstrak. Dataset ini perlu dimasukan ke Colab karena pengolahan secara langsung bisa jauh lebih cepat dibanding pengolahan melalui Google Drive. Hal itu telah kami buktikan dan sesuai dengan referensi. Selanjutnya, diimpor lah beberapa fungsi dari berbagai modul, utamanya dari modul tensorflow.

Setelah berbagai inisialisasi itu, terdapat pendefinisian awal. Ini mencakup pendefinisian nilai parameter-parameter penting yang digunakan dalam pengujian, seperti rasio pembagian data, parameter untuk *Image Transformation*, banyaknya epochs dan ukuran batch, serta parameter untuk optimizer. Terdapat pula pendefinisian fungsi untuk memuat bagian kode yang cukup panjang dan sering digunakan. Dengan mendefinisikan kode itu sebagai fungsi, penggunaannya dapat lebih mudah. Fungsi yang didefinisikan adalah fungsi untuk plot *training history* dan menyimpan datanya, fungsi untuk evaluasi model, serta fungsi untuk membuat matriks *confusion* dan menyimpannya.

Kemudian, dilakukan pembagian ulang pada dataset nya secara acak menggunakan fungsi tertentu. Selain itu, terdapat proses duplikasi file citra agar banyak file nya mencapai angka tertentu. File-file citra yang ada diduplikasi secara urut sampai bisa mencukupi. Jika belum cukup, maka proses diulang dari awal. Proses duplikasi ini hanya dilakukan untuk data "training" yang membutuhkan keseimbangan kelas.

Perlu diperhatikan bahwa pembagian dan duplikasi ini dilakukan sebelum *Image Transformation*. Jika datanya ditransform terlebih dahulu baru kemudian dibagi, akan ada citra yang sangat mirip pada data train dan test sehingga testing nya tidak adil.

Langkah selanjutnya adalah *Image Loading and Transformation*. Awalnya, terdapat pendefinisian untuk *load* dan *transform* citra melalui ImageDataGenerator. Citra masuk ke generator untuk ditransformasi secara acak sehingga tidak ada lagi citra hasil duplikasi yang sama persis. Dengan begitu, kelasnya bisa lebih seimbang dengan data yang tetap beragam.

Proses *Image Transformation* ini dilakukan sebelum *training*, tidak bersamaan dengan *training*. Ini karena data yang sama akan digunakan untuk beberapa kali *training* dengan model berbeda. Untuk banyak data sekitar 10 ribu, prosesnya membutuhkan sekitar satu menit.

Bagian selanjutnya adalah pengujian dengan model DenseNet169. Sebelumnya, perlu impor model awal dari tensorflow. Model yang diimpor sudah memiliki *pretrained weights* untuk pengolahan citra. Selanjutnya, terdapat penambahan beberapa layer. Layer apa saja yang perlu ditambahkan merupakan hasil dari beberapa kali pengujian dan kajian pustaka. Ringkasan dari modelnya dapat dilihat pada Tabel 1. Terlihat bahwa layer terakhirnya adalah layer Dense dengan empat output. Empat output ini berhubungan dengan empat kelas yang diklasifikasi.

Tabel 1: Ringkasan Model DenseNet169 yang Digunakan

Layer (type)	Output Shape	Param #
densenet169 (Functional)	(None, 5, 6, 1664)	12642880
flatten_1 (Flatten)	(None, 49920)	0
batch_normalization_3 (Batch	(None, 49920)	199680
dense_3 (Dense)	(None, 1024)	51119104
batch normalization 4 (Batch	(None, 1024)	4096
activation_2 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense 4 (Dense)	(None, 1024)	1049600
batch_normalization_5 (Batch	(None, 1024)	4096
activation_3 (Activation)	(None, 1024)	0
dropout 3 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 4)	4100

Total params: 65,023,556
Trainable params: 52,276,740
Non-trainable params: 12,746,816

Selanjutnya, model itu di-*compile* dengan optimizernya. Dalam pengujian ini, ketiga model menggunakan optimizer yang sama, yaitu SGD dengan *learning rate* sebesar 0,0001 dan *momentum* sebesar 0,9. Saat *compiling*, didefinisikan pula *loss* dan *metrics* yang dipakai. Dalam pengujian ini, ketiga model menggunakan *loss* bernama "categorical_crossentropy" dengan empat buah *metrics*: akurasi, presisi, *recall*, dan AUC (*Area Under Curve*).

Setelah didefinisikan, dijalankanlah *training.* Untuk tiap epoch, dengan ukuran batch 20, waktu yang dibutuhkan sekitar 20 – 25 detik. Model akhir yang dihasilkan disimpan lalu dipindahkan ke Google Drive. Ada pula plot *history training* yang gambar beserta datanya disimpan juga.

Kemudian, terdapat proses evaluasi terhadap modelnya. Evaluasi dilakukan dengan menguji model itu dengan data "training", "val", dan "test". Pengujian dengan data "training" umumnya menunjukan hasil yang bagus karena modelnya sudah familiar. Maka, yang krusial adalah pengujian dengan data "test". Selain itu, dibuat pula matriks *confusion* untuk melihat secara jelas berapa prediksi yang tepat dan kurang tepat.

Model kedua yang diuji adalah ResNet50. Konfigurasi untuk model ini pada dasarnya sama seperti sebelumnya, bahkan sangat mirip. Sebenarnya, dari referensi yang kami dapatkan, ada banyak penambahan layer. Namun, setelah beberapa kali percobaan, kami simpulkan bahwa layer yang terlalu banyak itu justru menurunkan performa. Ketika layer tambahannya dibuat sama seperti DenseNet169 sebelumnya, performanya justru lebih baik. Ringkasan dari model ini dapat dilihat pada Tabel 2. Model ini kemudian juga menjalani *training* dan evaluasi seperti sebelumnya.

Tabel 2: Ringkasan Model ResNet50 yang Digunakan

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 6, 7, 2048)	23587712
flatten_2 (Flatten)	(None, 86016)	0
batch_normalization_6 (Batch	(None, 86016)	344064
dense_6 (Dense)	(None, 1024)	88081408
batch_normalization_7 (Batch	(None, 1024)	4096
activation_4 (Activation)	(None, 1024)	0
dropout_4 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 1024)	1049600
batch_normalization_8 (Batch	(None, 1024)	4096
activation_5 (Activation)	(None, 1024)	0
dropout_5 (Dropout)	(None, 1024)	0
dense_8 (Dense)	(None, 4)	4100

Total params: 113,075,076
Trainable params: 89,311,236
Non-trainable params: 23,763,840

Model ketiganya adalah VGG16, yang digunakan di referensi utama. Di model ini, penambahan layernya tidak sebanyak sebelumnya. Layer-layer yang ditambahkan sebenarnya hanya mengikuti referensi utama karena performanya sudah baik. Namun, dengan mengurangi bagian tertentu, performanya justru bisa menjadi sedikit lebih baik. Ringkasan dari model ini dapat dilihat pada Tabel 3. Dapat dilihat bahwa layer tambahannya hanya ada empat: Flatten, Dense, Dropout, dan Dense lagi sebagai output akhir. Model ini kemudian juga menjalani *training* dan evaluasi seperti sebelumnya. Waktu *training* yang dibutuhkan oleh model ini relatif lebih lama dari dua lainnya.

Tabel 3: Ringkasan Model VGG16 yang Digunakan

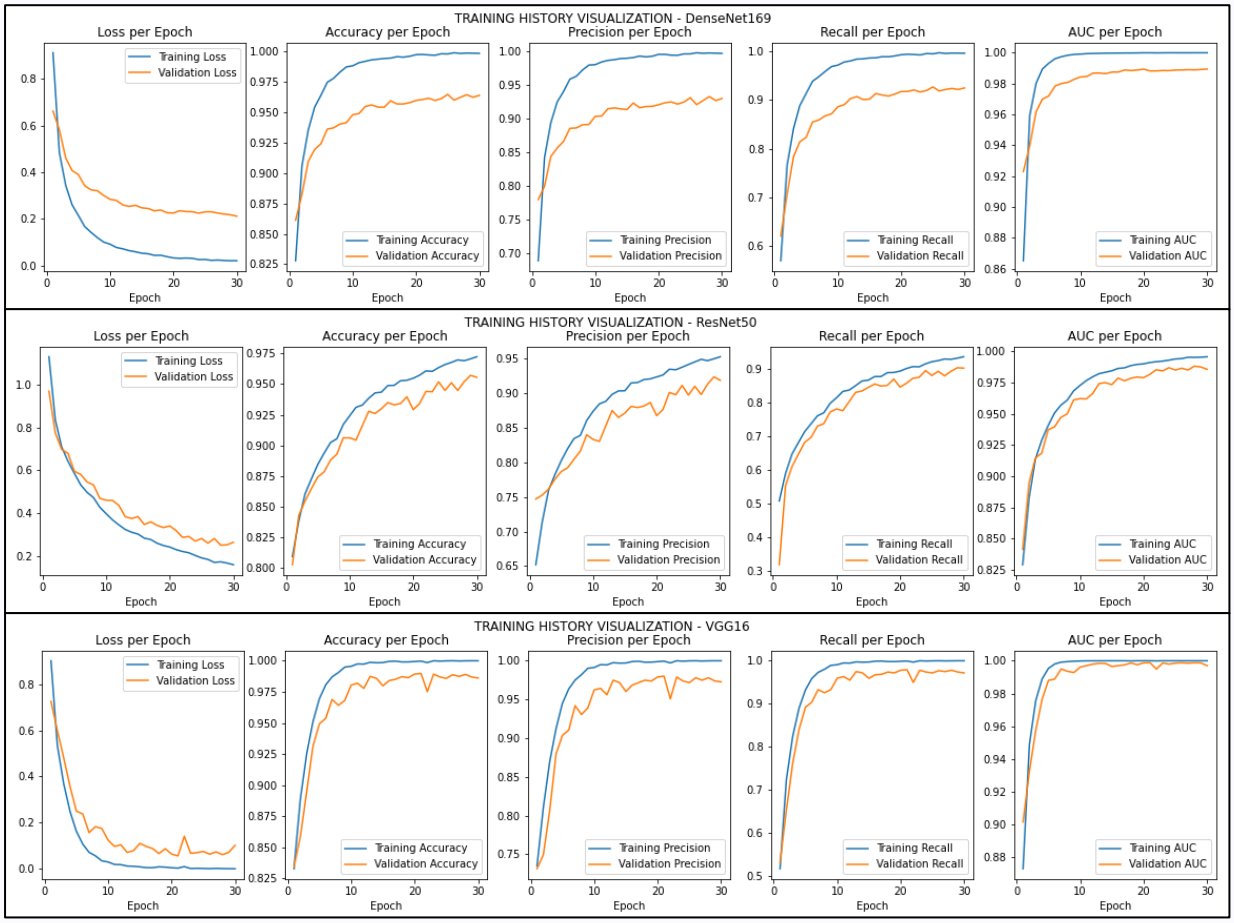
Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 512)	14714688
flatten_3 (Flatten)	(None, 512)	0
dense_9 (Dense)	(None, 1024)	525312
dropout_6 (Dropout)	(None, 1024)	0
dense_10 (Dense)	(None, 4)	4100

Total params: 15,244,100
Trainable params: 15,244,100
Non-trainable params: 0

IV. HASIL DAN ANALISIS

A. Training History

Training history menunjukan performa dari model ketika menjalani *training*. Performa itu diukur dengan berbagai *metrics* yang telah ditetapkan sebelumnya. Yang diukur adalah performa model ketika diuji dengan data "training" dan data "val". Untuk ketiga model, kurva *training history* nya dapat dilihat pada Gambar 6.



Gambar 6: Kurva *Training History* untuk Tiga Model yang Digunakan

Dari Gambar 6, terlihat bahwa model ResNet50 cenderung lambat untuk berubah dibanding dua model lainnya. Itu artinya, untuk epoch yang rendah, model ini belum memberikan performa yang cukup baik.

Di sisi lain, meski model DenseNet169 cukup cepat dalam berubah, terdapat perbedaan yang signifikan antara pengujian untuk "training" dan pengujian untuk "val". Ini menjadi kekurangan tersendiri yang tidak ada pada dua model lainnya.

Model VGG16 memberikan perubahan yang cepat dengan perbedaan yang tipis antara pengujian untuk "training" dan pengujian untuk "val". Terlihat bahwa hasilnya sudah cukup baik sejak epoch 10. Ini berarti, model ini sudah cukup baik bahkan untuk epoch yang rendah.

Namun, model VGG16 ini bukan tanpa kelemahan. Ketika *training*, dengan ukuran *batch* 20, model ini membutuhkan waktu sekitar satu menit untuk setiap epoch nya. Dibanding dua lainnya, proses *training* VGG16 membutuhkan waktu hampir tiga kali lebih lama. Ini tentu bisa menjadi pertimbangan ketika *training* dengan epoch yang jauh lebih banyak.

Bagaimana pun juga, pada akhirnya, ketiga model memberikan hasil yang cukup baik. Akurasi, presisi, *recall*, dan AUC nya secara umum sudah di atas 90%. Analisis untuk hasil akhirnya akan lebih jelas pada pembahasan selanjutnya.

B. Perbandingan Hasil Akhir

Tabel 4: Skor Akhir Evaluasi Ketiga Model

Model	**DenseNet169**			**ResNet50**			**VGG16**		
Result	**train**	**val**	**test**	**train**	**val**	**test**	**train**	**val**	**test**
Loss	0.00043	0.211389	0.16512	0.042155	0.26285	0.29496	0.00027	0.101679	0.064413
Accuracy	100%	96,3651%	97,1562%	99,5898%	95,5544%	94,6742%	100%	98,6140%	98,8883%
Precision	100%	92,9548%	94,589%	99,3063%	91,8085%	89,9265%	100%	97,2775%	97,8261%
Recall	100%	92,4686	94,0021%	99,0513%	90,272%	88,6246%	100%	97,1757%	97,7249%
AUC	100%	98,9396	99,4723%	99,9874%	98,5515%	98,3077%	100%	99,6993%	99,8189%

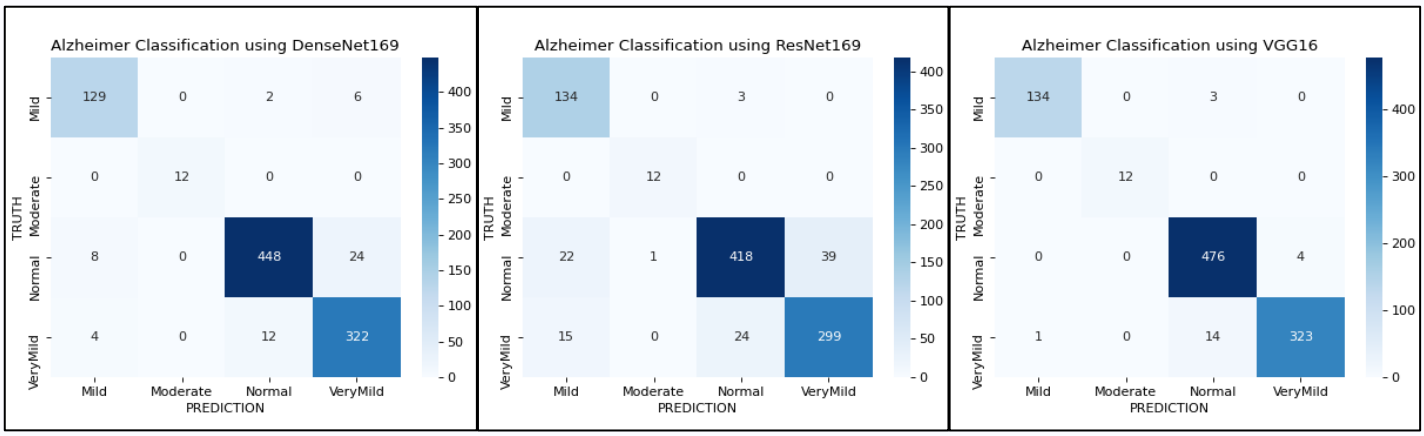
Skor dari evaluasi tiap model sudah disusun pada Tabel 4. Terlihat bahwa dari segi akurasi dan AUC, ketiga model sudah menunjukan performa yang sangat bagus. Untuk tiga parameter ukur lainnya, VGG16 memberikan hasil terbaik diikuti DenseNet169 kemudian ResNet50. Yang jelas terlihat kurang baik adalah pada *loss* nya. Nilai *loss* dari model DenseNet169 dan ResNet50 masih cukup tinggi. Selain itu, presisi dan *recall* pada model ResNet50 juga masih di bawah 90%.

Dari evaluasi menggunakan data "training", sebenarnya semuanya sudah menunjukan hasil yang sangat bagus, dengan nilai 99% sampai 100%. *Loss* nya juga sudah sangat rendah. Ini wajar saja mengingat dengan data itulah modelnya berlatih. Maka, modelnya pasti sudah cukup familiar dengan data itu dan bisa melakukan klasifikasi dengan baik. Untuk data "val" dan "test", modelnya belum familiar, bahkan belum pernah bertemu sekalipun dengan data "test". Maka, wajar jika performanya bisa jauh lebih rendah dibanding ketika menggunakan data "training".

Ketika meninjau hasil akhir ini, perlu diingat kembali bahwa dibanding dua lainnya, proses training VGG16 membutuhkan waktu hampir tiga kali lebih lama. Seandainya epoch untuk DenseNet169 dan ResNet50 dibuat tiga kalinya sehingga waktu proses trainingnya seperti VGG16, mungkin saja keduanya memberikan performa akhir yang tidak jauh berbeda dengan VGG16.

C. Matriks Confusion

Dua hasil sebelumnya sebenarnya sudah cukup untuk menunjukan performa dari modelnya. Namun, untuk lebih memahami bagaimana model menghadapi data yang ada, matriks *confusion* dapat digunakan. Matriks *confusion* untuk ketiga model dapat dilihat pada Gambar 7.



Gambar 7: Matriks *Confusion* untuk Tiga Model yang Digunakan

Matriks *confusion* ini menunjukan bagaimana model mengklasifikasikan data yang ada kemudian membandingkannya dengan label yang sebenarnya. Makin tinggi nilai dari bagian diagonal dibanding bagian lainnya, itu berarti modelnya makin baik. Itu karena bagian diagonal menunjukan keadaan saat prediksi sesuai dengan realita.

Dari sini, kita dapat melihat bahwa model ResNet50 melakukan kesalahan yang paling banyak, diikuti DenseNet169 kemudian VGG16 menjadi yang terbaik. Ini sama seperti pada pembahasan sebelumnya. Namun, ada juga hal yang tidak didapatkan dari pembahasan sebelumnya.

Dari sini, kita bisa melihat bahwa dari 480 citra yang sebenarnya normal, model ResNet salah memprediksinya sebagai *Mild* sebanyak 22 kali, sebagai *Moderate* sebanyak satu kali, serta sebagai *Very Mild* sebanyak 39 kali. Informasi seperti ini bisa menjadi cukup praktis. Perlu diperhatikan bahwa data untuk *Moderate* sejak awal memang sudah sedikit.

Kita dapat menyimpulkan bahwa beberapa kesalahan yang sering terjadi adalah: Normal dianggap *Very Mild*, Normal dianggap *Mild*, *Very Mild* dianggap *Mild*, dan *Very Mild* dianggap Normal. Ini tentu menjadi informasi yang penting bagi pengguna karena dia bisa mengetahui kapan modelnya dimungkinkan mengalami kesalahan. Ini juga bisa menjadi bahan evaluasi sehingga perbaikan modelnya dapat fokus pada kasus-kasus tertentu. Informasi seperti itu tidak didapatkan pada dua pembahasan sebelumnya namun bisa disajikan secara elegan melalui matriks *confusion*.

V. KESIMPULAN

Berbagai model *deep learning* dapat diterapkan untuk klasifikasi Alzheimer empat kelas berdasarkan citra MRI otak. Pengujian menunjukan bahwa model DenseNet169, ResNet50, dan VGG16 sudah memberikan performa yang cukup baik dalam hal akurasi dan nilai AUC. Akurasi dari ketiganya berturut-turut sebesar 97,16%, 94,67%, 98,89%.

Secara umum, model VGG16 memberikan performa yang terbaik dari berbagai sisi. Model ini juga bisa mencapai performa tinggi untuk nilai epoch yang rendah. Namun, dibanding dua model lainnya, waktu *training* model VGG16 tiga kali lebih lama. Ini tentu menjadi pertimbangan tersendiri ketika *training* dengan epoch yang banyak.

Dari matriks *confusion* yang dihasilkan, dapat disimpulkan bahwa beberapa kesalahan yang sering dilakukan model adalah: Normal dianggap *Very Mild*, Normal dianggap *Mild*, *Very Mild* dianggap *Mild*, dan *Very Mild* dianggap Normal. Ini tentu menjadi informasi yang penting bagi pengguna model atau pun pihak yang ingin mengembangkan model itu lebih jauh.

VI. DAFTAR PUSTAKA

[1] X.-M. Z. Xingzhong Zhao, "Deep learning of brain magnetic resonance images: A brief review," *Methods - Elsevier,* 2020.

[2] Y. P. M. L. Z. C. L. T. C. L. J. W. Jin Liu, "Applications of Deep Learning to MRI Images: A Survey," *Big Data Mining and Analytics,* vol. 1, pp. 1-18, 2018.

[3] S. M. A. M. A. S. R. Ammarah Farooq, "A Deep CNN based Multi-class Classification of Alzheimer's Disease using MRI," *Instrumentation and Measurement Society - IEEE,* 2017.

[4] L. T. K.-H. T. S. J. D. S. a. J. L. Feng Li, "A Robust Deep Model for Improved Classification," *Biomedical and Health Informatics,* vol. 19, 2015.

[5] M. M. M. B. Y. S. Atif Mehmood, "A Deep Siamese Convolution Neural Network for Multi-Class Classification of Alzheimer Disease," *Brain Sciences - MDPI,* p. 84, 2020.

[6] S. Dubey, "Alzheimer's Dataset (4 class of Images)," 2020. [Online]. Available: https://www.kaggle.com/tourist55/alzheimers-dataset-4-class-of-images. [Diakses 3 April 2021].

[7] "Image Classification," Science Direct, [Online]. Available: https://www.sciencedirect.com/topics/computer-science/image-classification. [Diakses 15 March 2021].

[8] K. P. Sung, "Accuracy, Precision, Recall or F1?," Towards Data Science, 15 March 2018. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. [Diakses 15 March 2021].

[9] DeepAI, "Loss Function," DeepAI, [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/loss-function. [Diakses 3 April 2021].

[10] S. Narkhede, "Understanding AUC - ROC Curve," towards data science, 26 June 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. [Diakses 3 April 2021].

[11] Z. B. D. C. Bryce Smith, "Smith_Burns_Cosmas_99%testaccuracy," Kaggle, 4 June 2020. [Online]. Available: https://www.kaggle.com/brycesmith/smith-burns-cosmas-99-testaccuracy/comments. [Diakses 3 April 2021].