PEMBELAJARAN MENDALAM UNTUK DETEKSI PENYAKIT KATARAK MELALUI FUNDUS MATA

LAPORAN TUGAS

Thirafi Najwan Kurniatama
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Indonesia
13520157

Michella Chandra
Faculty of Mathematics and Natural Sciences
Institut Teknologi Bandung, Indonesia
10118011

Abstrak

Pada zaman ini, teknologi informasi dan komputasi diimplementasikan dalam berbagai bidang, termasuk kesehatan. Selain untuk mengembangkan metode pengobatan, teknologi juga banyak digunakan untuk membantu pendeteksian penyakit. Salah satu aplikasinya adalah pendeteksian penyakit lewat analisis portret fundus mata, dimana pasien yang mengidap penyakit tertentu akan memiliki karakteristik fundus mata berbeda. Metode pendeteksian biasanya dilakukan secara analisis visual dibantu dengan model pembelajaran mendalam. Dalam tugas ini akan digunakan 3 jenis arsitektur model yaitu *Multi Layer Perceptron* (MLP), *Convolutional Neural Network* (CNN), dan *Recurrent Neural Network* (RNN). Data berupa gambar 2 dimensi yaitu portret fundus mata yang diberikan label dan dibagi menjadi training dan testing. Data training kemudian akan diproses untuk melatih masing-masing model, dan model akan menghasilkan prediksi label dari data testing yang akan diukur akurasinya.

Keywords CNN · RNN · MLP · deteksi penyakit mata

1 Pendahuluan

Implementasi teknologi informasi dan komputasi pada bidang kesehatan bukanlah hal yang asing untuk ditemukan pada era modern ini. Dalam bidang ini, teknologi digunakan untuk berbagai keperluan, mulai dari administrasi, pengadaan obat / farmasi, penyimpanan data pasien, hingga penelitian [1]. Selain mempercepat proses pengobatan secara umum, teknologi juga dapat menambah faktor keamanan dan kenyamanan bagi pasien yang sedang diobati. Sampai saat ini pun, teknologi di bidang medis masih aktif dikembangkan untuk mempercepat proses penanganan dan pengobatan dari berbagai penyakit.

Tidak hanya dalam hal administrasi dan pengembangan metode pengobatan, teknologi informasi dan komputasi juga dapat digunakan dalam hal pendeteksian penyakit. Salah satu contoh implementasi yang akan dibahas dalam tugas ini adalah analisis portret mata. Beberapa penyakit tertentu bisa menyebabkan tampilan visual mata yang berbeda, seperti perubahan warna, corak-corak tertentu di bagian-bagian tertentu, dan lainnya. Berdasarkan tampilan visualnya, mata akan diklasifikasikan ke dalam jenis-jenis penyakit yang mungkin diderita pemiliknya.

Untuk membantu proses klasifikasi ini, biasanya digunakan model yang dibangun berdasarkan prinsip *deep learning* atau pembelajaran mendalam. Beberapa arsitektur tertentu sangatlah membantu untuk digunakan dalam menganalisis gambar, khususnya yang jumlahnya sangat besar. Melalui pembelajaran mendalam, diharapkan komputer mampu menghasilkan informasi baru dengan mempelajari data yang diproses dalam model tertentu. Pembelajaran mendalam juga terbukti memiliki performa lebih tinggi jika dibandingkan dengan model komputasi klasik lain seperti SVM.

Tujuan dari tugas ini adalah untuk membuat model yang bisa mendeteksi penyakit katarak berdasarkan tampilan visual mata kiri atau kanan pasien. Adapun data yang digunakan berupa 600 foto fundus mata kiri dan kanan dari pasien, dimana masing-masing gambar diberikan label normal dan karakter dari dokter mengenai penyakit yang diderita. Adapun label ini adalah Normal (N) dan Katarak (C). Foto-foto ini juga diambil dengan kamera dari berbagai merek sehingga resolusi fotonya berbeda.

Dalam melakukan analisis foto pada tugas ini, akan digunakan tiga buah arsitektur model, yaitu *Multi Layer Perceptron* (MLP), *Convolutional Neural Network* (CNN), dan *Recurrent Neural Network* (RNN). *Multi Layer Perceptron* adalah kelas *feedforward artificial neural network* yang *fully connected*. Sesuai namanya, MLP adalah jenis ANN yang memiliki banyak layer, dan merupakan salah satu bentuk paling sederhana dan mendasar dari model berbasis pembelajaran mendalam. MLP setidaknya memiliki setidaknya tiga buah layer, yaitu input, hidden layer, dan output. MLP menggunakan teknik *backpropagation* untuk memperbaharui parameternya setiap kali model dilatih.

Arsitektur kedua adalah *Convolutional Neural Network* (CNN). CNN merupakan model yang biasa dipakai untuk menangani data yang bersifat lokalitas spasial, sehingga sangat berguna untuk model yang inputntya berupa gambar. CNN mendapatkan input berupa matriks 2 dimensi, dimana lokasi setiap data dalam matriks tersebut bersifat penting karena akan mempengaruhi penghitungan nilai data yang bertetangga dengannya. Secara umum, CNN terdiri dari 2 tipe *layer* atau lapisan. Pertama adalah lapisan konvolusional dimana proses konvolusi (penghitungan yang lebih menekankan ke lokalitas spasial) terjadi dan hasilnya diteruskan ke lapisan selanjutnya. Lapisan kedua adalah lapisan *pooling* yang mengurangi dimensi *feature*. Pada lapisan *pooling* terakhir, data akan menjadi vektor satu dimensi yang terhubung dengan MLP.

Arsitektur terakhir adalah *Recurrent Neural Network* (RNN). RNN adalah jenis arsitektur pembelajaran mendalam yang digunakan khususnya untuk data yang sifatnya berurutan atau bersambung. RNN bekerja seiring waktu berjalan, dimana untuk setiap time step input akan dimasukkan dan model dilatih untuk menghasilkan output jika diperlukan. Kemudian, informasi pada hidden state akan diteruskan ke time step berikutnya (berfungsi sebagai memori). Pada setiap time step, proses dan parameter yang sama akan digunakan secara berulang. Dalam kasus ini, RNN digunakan karena sifatnya yang tidak membuang informasi begitu saja dalam pembelajarannya, namun menyimpannya ke dalam memori terlebih dahulu. Sifat ini akan berguna ketika pelatihan model untuk klasifikasi dilakukan sehingga prediksi bisa bekerja sesuai dengan ingatan yang tersimpan.

2 Metodologi

Metode penelitian terbagi menjadi 3 bagian berbeda untuk MLP, CNN dan RNN. Perhatikan bahwa metodologi ini dibuat berdasarkan sesuai dengan apa yang tertulis di Notebook yang dikumpulkan.

Untuk setiap arsitektur, data diekstrak terlebih dahulu. Pada Notebook yang dikumpulkan, tahap ini dicantumkan di bagian ***Load and Cleansing Data***.

Data diambil dari website `https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k`. Data input berupa foto fundus mata kiri dan kanan pasien, dimana untuk setiap entrinya terdapat informasi umur, jenis kelamin (secara biner dan tertulis), status penyakit (secara biner) dan label berupa angka yang berkorespondensi dengan kondisi penyakit yang diderita. Output untuk tugas ini adalah klasifikasi fundus mata normal atau katarak.

2.1 *Data Preprocessing*

Dari data tersebut, akan dipilih masing-masing 300 sampel secara acak untuk mata katarak dan mata normal. Sampel acak ini nantinya akan dipisah menjadi 480 data untuk melatih model dan 120 data untuk menguji model. Data yang digunakan untuk prediksi adalah gambar fundus kiri atau fundus kanan pasien.

Dimensi gambar awal yang digunakan adalah *preprocessed image* yang sudah disediakan dalam *dataset*. Gambar tersebut berukuran 512x512. Namun, saat proses pelatihan model, dimensi gambar akan menyesuaikan dimensi optimal masing-masing model. Hal ini dilakukan dengan mengubah dimensi gambar menggunakan teknik *bilinear interpolation* serta memotong gambar di tengah sesuai dengan dimensi yang diinginkan. Terakhir, piksel tiap gambar akan dinormalisasi dengan rata-rata dan standar deviasi 0.5.

2.2 *Multi Layer Perceptron* (MLP)

Pada tugas ini, model MLP yang dibangun memiliki 5 buah layer, yaitu layer input berukuran 128x128x3 = 49152 (128x128 pixel dan 3 channel) neuron, 3 buah *hidden layer fully connected* yang masing-masing berukuran 128, 64 dan 32 dengan fungsi aktivasi ReLU, dan layer output berukuran 1 neuron dengan fungsi aktivasi Sigmoid. Digunakan juga optimisasi NAdam dan *Loss Function* Binary Cross Entropy.

2.2.1 *Forward Propagation*

Pada *input layer* terdapat 49152 neuron yang berisi gambar setiap pixel foto fundus kiri atau kanan mata pasien. Melalui *forward propagation*, data input $\mathbf{X}$ akan menjadi *neuro transmitter* $\mathbf{Z}^{(1)}$

$$\mathbf{Z}^{(1)} = \mathbf{X}W^{(0)} + \mathbf{b}^{(0)}$$

yang akan diproses menggunakan fungsi aktivasi f **ReLU** menjadi

$$\mathbf{A}^{(1)} = f(\mathbf{Z}^{(1)}).$$

Output $\mathbf{A}^{(1)}$ dari *hidden layer* pertama akan menjadi input bagi *hidden layer* kedua melalui *neuro transmitter* $\mathbf{Z}^{(2)}$

$$\mathbf{Z}^{(2)} = \mathbf{A}^{(1)}W^{(1)} + \mathbf{b}^{(1)}$$

yang menjadi input bagi $\mathbf{A}^{(2)}$ dari *hidden layer* kedua, diproses oleh fungsi aktivasi f ReLU

$$\mathbf{A}^{(2)} = f(\mathbf{Z}^{(2)}).$$

Output tersebut akan menjadi input bagi *hidden layer* ketiga dengan proses seperti sebelumnya

$$\mathbf{Z}^{(3)} = \mathbf{A}^{(2)}W^{(2)} + \mathbf{b}^{(2)}$$

dan diproses oleh fungsi aktivasi f ReLU menjadi

$$\mathbf{A}^{(3)} = f(\mathbf{Z}^{(3)}).$$

$\mathbf{A}^{(3)}$ dari *hidden layer* ketiga menjadi input bagi *neuro transmitter* $\mathbf{Z}^{(4)}$

$$\mathbf{Z}^{(4)} = \mathbf{A}^{(3)}W^{(3)} + \mathbf{b}^{(3)}$$

yang akan menjadi input untuk menentukan $\mathbf{A}^{(4)} = \hat{\mathbf{Y}}$ pada output layer. Seperti sebelumnya, $\mathbf{A}^{(4)}$ diproses menggunakan fungsi aktivasi, namun kali ini menggunakan fungsi S **Sigmoid**.

$$\mathbf{A}^{(4)} = \hat{\mathbf{Y}} = S(\mathbf{Z}^{(4)}).$$

Fungsi aktivasi Sigmoid sendiri berbentuk sebagai berikut

$$s(x) = \frac{1}{1 + e^{-x}}$$

dengan $t = 1, 2, \cdots, T$ dan $j = 1, 2$.

2.2.2 *Backward Propagation*

Backward propagation dilakukan untuk meminimumkan *loss function* E dimana

$$E = \sum_{j=1}^{8} E_j$$

dimana fungsi kerugian E_j untuk kelas output ke j berupa *binary cross entropy* dengan formula

$$E_j = -w_n[y_n \cdot logx_n + (1 - y_n) \cdot log(1 - x_n)])$$

dengan $j = 1, 2$.

Untuk meminimukan nilai *loss function*, perlu dicari nilai parameter bobot sehingga nilai prediksi sedekat mungkin dengan aktualnya. Pembaharuan nilai bobot sendiri dilakukan dengan cara sebagai berikut

$$\mathbf{W}^{(2),(n+1)} = \mathbf{W}^{(2),(n)} - \alpha * \frac{\partial \mathbf{E}}{\partial \mathbf{W}^{(2)}}$$

$$\mathbf{W}^{(1),(n+1)} = \mathbf{W}^{(1),(n)} - \alpha * \frac{\partial \mathbf{E}}{\partial \mathbf{W}^{(1)}}$$

$$\mathbf{W}^{(0),(n+1)} = \mathbf{W}^{(0),(n)} - \alpha * \frac{\partial \mathbf{E}}{\partial \mathbf{W}^{(0)}}$$

2.3 *Convolutional Neural Network* (CNN)

Pada Model CNN ini akan digunakan arsitektur VGG11 yang menerima *input* berupa gambar dengan dimensi 224x224x3. Gambar akan diproses melalui lima lapisan VGG Block dengan ReLU sebagai proses konvolusi lalu dilanjutkan melalui tiga *fully connected* layer dengan dua lapisan *dropout* sebagai proses penentuan keputusan.

Kelima lapisan VGG Block yang digunakan memiliki dimensi konvolusi 64, 128, 256, 512, dan 512 serta jumlah lapisan konvolusi 1, 1, 2, 2, dan 2 secara berturut-turut. Lalu, lapisan *fully connected* yang digunakan memiliki dimensi berturut 512*7*7, 4096, dan 1 dengan masing masing memiliki lapisan *droput* dengan probabilitas 0.5.

Model akan dilatih menggunakan metode *gradient descent* dengan proses *forward* dan *backward* propagation serta loss function yang sama dengan model sebelumnya. Model ini akan menggunakan optimisasi SGD dengan *learning rate* 0.001, *momentum* 0.9, dan *weight_decay* 0.0005.

2.3.1 *Convolutional Layer*

Proses paling penting dari model CNN adalah lapisan konvolusinya. Lapisan ini mencoba mengekstrak relasi lokalitas spasial yang biasanya terdapat pada data yang berupa citra atau gambar. Lapisan ini bekerja dengan memanfaatkan prinsip konvolusi diskrit dengan formula sebagai berikut

$$[H]_{i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum c[V]_{a,b,c,d}[X]_{i+a,j+b,c}$$

dengan Δ merepresentasikan derajat lokalitas dan a, b, c, secara berturut merepresentasikan ruang tinggi, panjang, dan *channel*.

2.3.2 *VGG Block*

VGG Block adalah sekumpulan lapisan konvolusi dengan aktivasi non-linear dan diakhiri dengan lapisan *pooling*, biasanya digunakan *maximum pooling*. VGG Block merekomendasikan lapisan konvolusi dengan kernel $3x3$ dan padding 1 untuk menjaga dimensi awal. Lapisan *maximum pooling* yang direkomendasikan memiliki kernel $2x2$ dengan stride 2 untuk mengurangi dimensi input menjadi setengah awalnya.

2.4 *Recurrent Neural Network* (RNN)

Model arsitektur RNN yang digunakan pada tugas ini berupa model ***Long Short-Term Memory*** (**LSTM**). Salah satu hal prominen dari model ini adalah ***memory cell*** atau **sel** yang ukurannya sama dengan *hidden state* model, yang sesuai namanya berfungsi sebagai pengingat kejadian yang sudah terjadi namun dalam situasi yang terkontrol (bisa dikendalikan apa dan kapan memori dimasukkan atau tidak). Dalam mengontrol *memory cell* digunakan dua buah pintu. Pintu pertama dibutuhkan untuk membaca entri hasil sel, yang akan disebut *output gate*. Pintu kedua dibutuhkan untuk membaca entri yang akan dimasukkan ke dalam sel, disebut *input gate*, dan pintu terakhir berfungsi untuk menghapus dan mengulang isi dari sel yang disebut *forget gate*.

Misalkan h adalah hidden unit, ukuran batch adalah n, dan jumpah input d. Maka, untuk input $\mathbf{X}_t \in \mathbb{R}^{n\times d}$ dan *hidden state* $\mathbf{H}_{t-1} \in \mathbb{R}^{n\times h}$, *input gate* $\mathbf{I}_t \in \mathbb{R}^{n\times h}$, *forget gate* $\mathbf{F}_t \in \mathbb{R}^{n\times h}$ dan *output gate* $\mathbf{O}_t \in \mathbb{R}^{n\times h}$ dihitung sebagai berikut:

$$\mathbf{I}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xi} + \mathbf{H}_{t-1}\mathbf{W}_{hi} + \mathbf{b}_i),$$
$$\mathbf{F}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xf} + \mathbf{H}_{t-1}\mathbf{W}_{hf} + \mathbf{b}_f),$$
$$\mathbf{O}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xo} + \mathbf{H}_{t-1}\mathbf{W}_{ho} + \mathbf{b}_o),$$

dengan $\mathbf{W}_{xi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{xo} \in \mathbb{R}^{d\times x}$ dan $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho} \in \mathbb{R}^{h\times h}$ adalah parameter berat dan $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^{1\times h}$ adalah parameter bias.

Kemudian, definisikan *candidate memory cell* $\hat{\mathbf{C}}_t \in \mathbb{R}^{ntimesh}$ sebagai kandidat sel memori, yang komputasinya mirip dengan ketiga pimtu sebelumnya namun menggunakan fungsi $\tanh$. Formula penghitungannya pada waktu ke t adalah sebagai berikut:

$$\hat{\mathbf{C}}_t = \tanh(\mathbf{X}_t\mathbf{W}_{xc} + \mathbf{H}_{t-1}\mathbf{W}_{hc} + \mathbf{b}_c),$$

dengan $\mathbf{W}_{xc} \in \mathbb{R}^{d\times h}$ dan $\mathbf{W}_{hc} \in \mathbb{R}^{h\times h}$ adalah parameter berat dan $\mathbf{b}_c \in \mathbb{R}^{1\times h}$ adalah parameter bias.

Sesuai dengan yang disebutkan sebelumnya, *input gate* $\mathbf{I}_t$ mengendalikan berapa banyak data yang diambil melalui $\hat{\mathbf{C}}_t$ dan *forget gate* $\mathbf{F}_t$ menentukan berapa banyak isi sel memori lama $\mathbf{C}_{t-1} \in \mathbb{R}^{n \times h}$ diambil. Proses ini bisa dituliskan sebagai berikut:

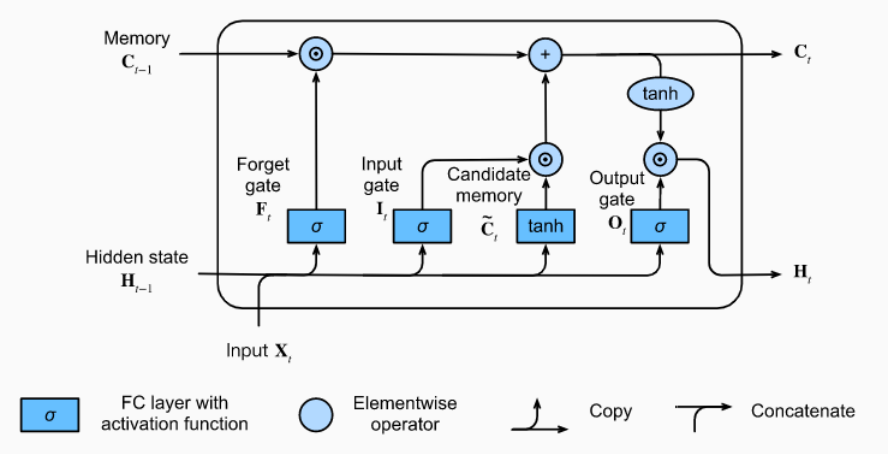
$$\mathbf{C}_t = \mathbf{F}_t \bigodot C_{t-1} + \mathbf{I}_t \bigodot \hat{\mathbf{C}}_t.$$

Contoh penggunaannya, jika *forget gate* kurang lebih 1 dan *input gate* kurang lebih 0, maka sel memori $\mathbf{C}_{t-1}$ akan disimpan seiring waktu dan diteruskan ke iterasi waktu berikutnya.

Dalam mendefinisikan *hidden state* $\mathbf{H}_t \in \mathbb{R}^{n \times h}$, digunakan *output gate*. Digunakan fungsi $\tanh$ pada sel memori sebagai berikut:

$$\mathbf{H}_t = \mathbf{O}_t \bigodot \tanh(\mathbf{C}_t).$$

Proses LSTM dalam bentuk grafik bisa digambarkan dalam diagram sebagai berikut.



Gambar 1: Skema Umum LSTM (sumber: d2l.ai)

Pada model yang digunakan dalam paper ini, akan digunakan tiga LSTM instance dengan *input dimension* 224, *hidden dimension* 32, serta *layer dimension* 2 untuk memproses masing-masing *channel* dari gambar. Ketiga output tersebut akan digabungkan dan diproses menggunakan satu *dropout layer* dengan probabilitas 0.5 dan satu *fully connected layer* dengan 96 neuron.

Proses *feeding* LSTM pada tiap langkahnya akan memberikan satu kolom piksel (224x1) sebanyak 224 kali. Keluaran akan ditentukan dari *hidden states* pada langkah terakhir.

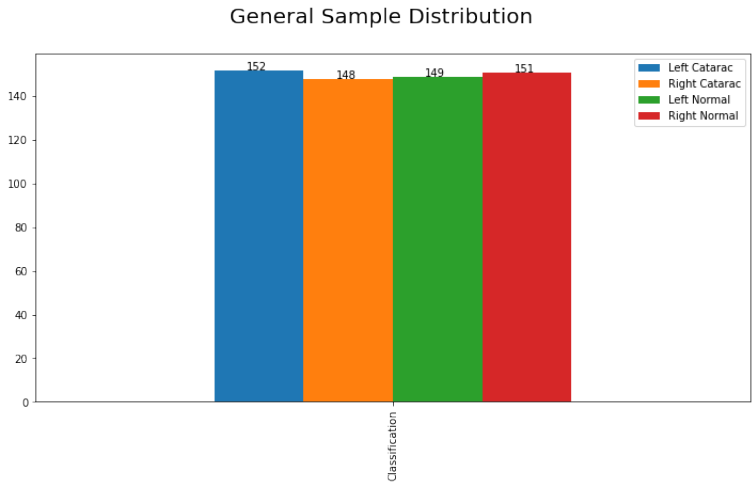
3 Hasil

Berikut dilampirkan distribusi data, hasil training yaitu tabel perbandingan antara label aktual dan prediksi, nilai total akurasi, serta nilai evaluasi *Recall*, *Precision*, dan *F1 Score* untuk masing-masing model. Disertakan juga grafik loss function untuk setiap model.

3.1 *Distribusi Data*

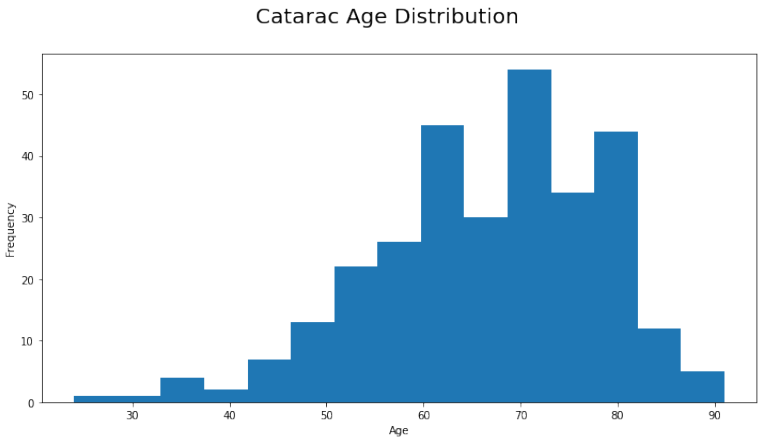
Berikut adalah statistik berupa *bar chart* dan histogram relevan mengenai data yang digunakan

3.1.1 *Persebaran Klasifikasi*

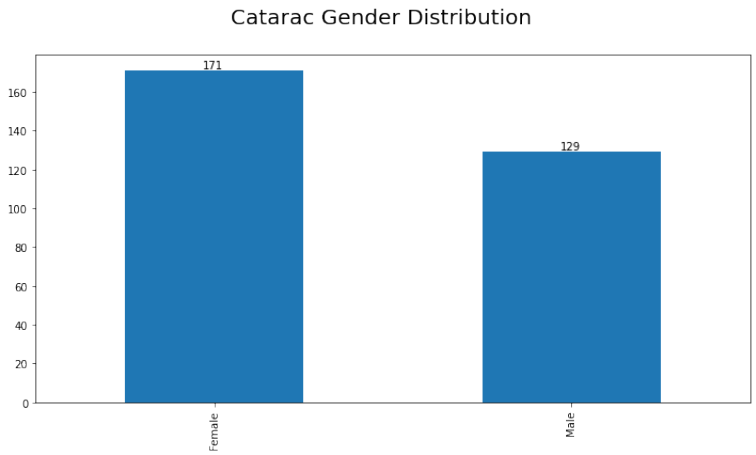


Gambar 2: Persebaran data umum

3.1.2 *Mata Katarak*

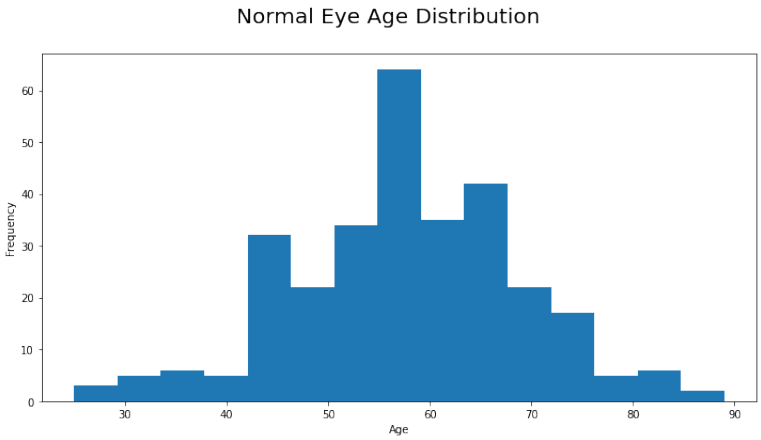


Gambar 3: Persebaran umur pasien katarak

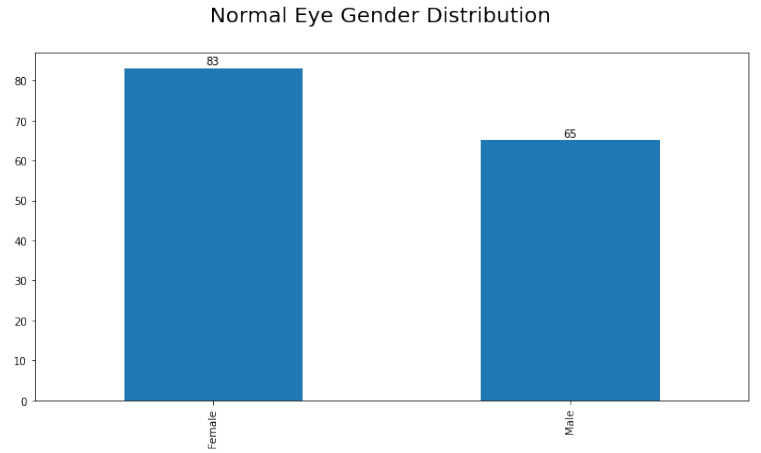


Gambar 4: Persebaran gender pasien katarak

3.1.3 *Mata Normal*



Gambar 5: Persebaran umur pasien normal



Gambar 6: Persebaran umur pasien normal

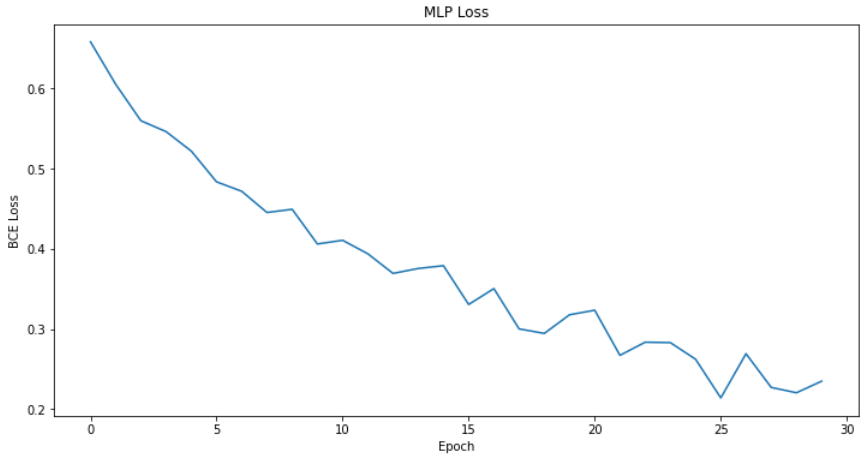
3.2 *Multi Layer Perceptron* (MLP)

		Predicted	
		Normal	Katarak
Actual	Normal	45	12
	Katarak	15	48

Tabel 1: Matriks hasil prediksi label MLP

Accuracy: 0.7750		Labels	
		Normal	Katarak
Evaluation	Recall	0.789474	0.761905
	Precision	0.750000	0.800000
	F1 Score	0.769231	0.780488

Tabel 2: Matriks evaluasi hasil prediksi MLP



Gambar 7: *Loss Function* MLP

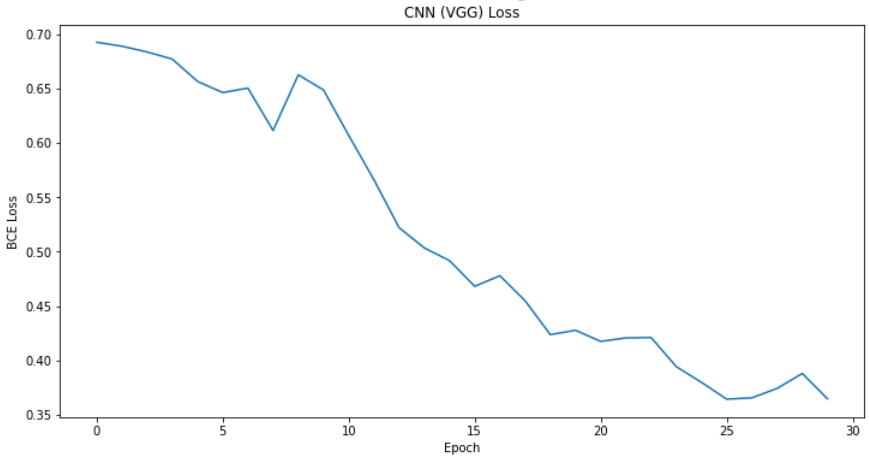
3.3 *Convolutional Neural Network* (CNN)

		Predicted	
		Normal	Katarak
Actual	Normal	40	17
	Katarak	5	58

Tabel 3: Matriks hasil prediksi label CNN

Accuracy: 0.8167		Labels	
		Normal	Katarak
Evaluation	Recall	0.701754	0.920635
	Precision	0.888889	0.773333
	F1 Score	0.784314	0.840580

Tabel 4: Matriks evaluasi hasil prediksi CNN



Gambar 8: *Loss Function* CNN

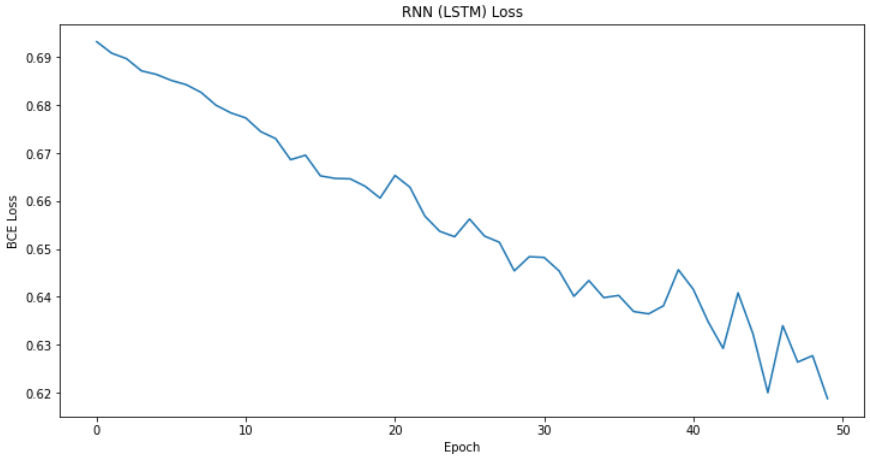
3.4 *Recurrent Neural Network* (RNN)

		Predicted	
		Normal	Katarak
Actual	Normal	23	34
	Katarak	27	36

Tabel 5: Matriks hasil prediksi label RNN

Accuracy: 0.4917		Labels	
		Normal	Katarak
Evaluation	Recall	0.403509	0.571429
	Precision	0.460000	0.514286
	F1 Score	0.429907	0.541353

Tabel 6: Matriks evaluasi hasil prediksi RNN



Gambar 9: *Loss Function* RNN

4 Diskusi

Dari hasil yang tertera pada bagian sebelumnya, terlihat model CNN dengan arsitektur VGG11 memiliki tingkat akurasi yang lebih tinggi dibandingkan model yang lain. Hal ini wajar mengingat CNN memang memiliki performa yang baik untuk mengambil relasi spasial. Hasil dari model ini masih bisa terus dikembangkan dengan penambahan dataset dan *epoch* pada training. Hal ini tidak dilakukan pada percobaan kali ini dikarenakan keterbatasan waktu dan sumber daya.

Model MLP memiliki tingkat akurasi yang hampir setinggi CNN. Hal ini dikarenakan seluruh piksel tetap dijadikan *consideration* untuk hasil akhirnya. Namun, karena itu jugalah, model MLP ini lebih rawan untuk mem-*pickup* piksel yang tidak relevan dibanding dengan CNN yang mengandalkan konvolusi dan *spatial downsampling*. Hasil dari model ini tidak bisa lagi dikembangkan dengan menambahkan *epoch* karena akan terjadi *overfitting*. Model ini mungkin bisa dikembangkan dengan *fine-tuning hyperparameter* lebih jauh.

Model terakhir dan yang paling buruk adalah LSTM. Hal ini dikarenakan LSTM memang tidak difokuskan untuk pemrosesan gambar dan data yang digunakan bukan berupa data sekuensial. Metode yang digunakan pada pemrosesan LSTM di paper ini juga tidak terlalu cocok karena model mencari hubungan temporal per kolom piksel yang dimasukkan, yang mana tentunya hubungan temporal ini sangat *negligible*. Model ini sepertinya sudah tidak dapat dikembangkan karena model ini memang tidak cocok dengan data yang diberikan.

5 Kontribusi

Thirafi Najwan Kurniatama, 13520157, berperan terutama untuk membuat kode di Notebook. Sebagai anak Informatika yang sudah lebih terbiasa dengan *Machine Learning* dan Python, Thirafi berinisiatif untuk membuat model di Notebook. Beliau juga mengajarkan anggota lain tentang istilah-istilah dan kode yang mungkin kurang familiar untuk ditemukan.

Michella Chandra, 10118011, berperan terutama untuk membuat laporan. Kode mayoritas dibuat oleh Thirafi agar tidak menimbulkan ketidaksesuaian *syntax* atau *variabel*, sehingga Michella sebagai orang yang lebih berpengalaman dalam menggunakan LaTeX lebih berkontribusi dalam pembuatan laporan.

Keduanya memiliki kontribusi kurang lebih sama di pembuatan video.

References

[1] J. Fadil Ahmad, B. Diana Teknologi Informasi Kesehatan I Aplikasi Komputer Dasar. (2018). Pusat Pendidikan Sumber Daya Manusia Kesehatan.

[2] A. Canziani, A. Paszke, E. Culurciello. An analysis of deep neural network models for practical applications. arXiv preprint, arXiv: 1605.07678.

[3] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. Nature 521. (2015), pages 436-444.