



TESIS - SS235401

**ESTIMASI PARAMETER PADA MODEL HIGHLY
MULTIVARIATE SPATIO-TEMPORAL LOG
GAUSSIAN COX PROCESS MENGGUNAKAN
PROBABILISTIC DEEP LEARNING**

**EKKY RINO FAJAR SAKTI
NRP. 6003221012**

**Dosen Pembimbing:
Dr. Achmad Choiruddin, S.Si, M.Sc
Dr. Dra. Kartika Fithriasari, M.Si**

**Program Magister
Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya
2024**



TESIS - SS235401

**ESTIMASI PARAMETER PADA MODEL HIGHLY
MULTIVARIATE SPATIO-TEMPORAL LOG
GAUSSIAN COX PROCESS MENGGUNAKAN
PROBABILISTIC DEEP LEARNING**

**EKKY RINO FAJAR SAKTI
NRP 6003221012**

**Dosen Pembimbing
Dr. Achmad Choiruddin, S.Si, M.Sc
Dr. Dra. Kartika Fithriasari, M.Si**

**Program Magister
Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
2024**

(Halaman ini sengaja dikosongkan)



TESIS - SS235401

**PARAMETER ESTIMATION OF HIGHLY
MULTIVARIATE SPATIO-TEMPORAL LOG
GAUSSIAN COX PROCESS USING PROBABILISTIC
DEEP LEARNING**

**EKKY RINO FAJAR SAKTI
NRP 6003221012**

**Supervisors
Dr. Achmad Choiruddin, S.Si, M.Sc
Dr. Dra. Kartika Fithriasari, M.Si**

**Program of Master
Departement of Statistics
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
2024**

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Statistika (M.Stat)
Di
Institut Teknologi Sepuluh Nopember
Oleh:
EKKY RINO FAJAR SAKTI
NRP: 6003221012

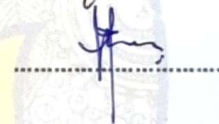
Tanggal Ujian: 8 Juli 2024
Periode Wisuda: September 2024

Disetujui Oleh:
Pembimbing:

1 Dr. Achmad Choiruddin, S.Si., M.Sc.
NIP. 1991201911101



2 Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002



Penguji:

1 Dr. Sutikno, S.Si., M.Si.
NIP. 19710313 199702 1 001



2 Dr. Hidayatul Khusna, S.Si.
NIP. 1994202012032



Kepala Departemen Statistika
Fakultas Sains dan Analitika Data



Dr. Dra. Kartika Fithriasari, M.Si
NIP. 19691212 199303 2 002

(Halaman ini sengaja dikosongkan)

***ESTIMASI PARAMETER PADA MODEL HIGHLY MULTIVARIATE
SPATIO-TEMPORAL LOG GAUSSIAN COX PROCESS MENGGUNAKAN
PROBABILISTIC DEEP LEARNING***

Oleh : Ekky Rino Fajar Sakti
Pembimbing : Dr. Achmad Choiruddin, S.Si, M.Sc
Co-Pembimbing : Dr. Dra. Kartika Fithriasari, M.Si

ABSTRAK

Data *multivariate spatio-temporal point pattern* semakin mudah dijumpai akhir-akhir ini karena pesatnya perkembangan teknologi dalam pengambilan data, seperti lokasi banyak spesies pohon pada sebuah hutan tropis yang diobservasi pada banyak waktu dan lokasi beberapa tipe kriminal di suatu daerah pada rentang periode waktu tertentu. Salah satu analisis statistika yang penting dilakukan terhadap data *multivariate spatio-temporal point pattern* adalah estimasi parameter untuk mengetahui pola persebaran titik dan hubungan antar tipe *multivariate* dan waktu. Namun demikian, penerapan estimasi parameter menggunakan metode berbasis *likelihood* sulit dilakukan karena tingginya kompleksitas model dan jumlah parameter yang harus diestimasi. Salah satu alternatif ketika metode parametrik sulit digunakan adalah *deep learning*. Walaupun *deep learning* telah terbukti mampu untuk memodelkan data berukuran besar dengan akurasi yang baik, hanya sedikit penelitian yang menerapkan *deep learning* untuk menyelesaikan permasalahan *multivariate spatio-temporal point pattern*. Oleh sebab itu, penelitian ini bertujuan untuk menerapkan *deep learning* sebagai *estimator* parameter untuk model *multivariate spatio-temporal* Log Gaussian Cox Process (LGCP). Penelitian ini menggunakan konsep *probabilistic deep learning* sehingga setiap parameter yang diestimasi memiliki distribusi yang sesuai dengan asumsinya. Model *probabilistic deep learning* dievaluasi pada studi simulasi. Pada studi terapan, model *probabilistic deep learning* digunakan sebagai *estimator* parameter untuk *dataset* Barro Colorado Island (BCI). *Dataset* ini berisi data *point pattern* dari ratusan spesies pohon yang diobservasi pada beberapa waktu. Hasil analisis pada studi simulasi menunjukkan bahwa *probabilistic deep learning* memiliki waktu pelatihan 68% lebih singkat, nilai MSE yang lebih kecil hingga 69%, serta nilai RMSE yang lebih kecil pada parameter $\hat{\alpha}$ sebesar 33%, parameter $\hat{\phi}$ sebesar 27%, dan parameter $\hat{\psi}$ sebesar 46% daripada penelitian sebelumnya. Pada studi terapan, *probabilistic deep learning* memiliki waktu pelatihan yang lebih singkat hingga 10 kali lipat dan nilai MSE yang lebih kecil hingga 50% daripada penelitian sebelumnya.

Kata kunci : LGCP, *Multivariate Point Pattern*, *Neural Network*, *Probabilistic Deep Learning*

(Halaman ini sengaja dikosongkan)

PARAMETER ESTIMATION OF HIGHLY MULTIVARIATE SPATIO-TEMPORAL LOG GAUSSIAN COX PROCESS USING PROBABILISTIC DEEP LEARNING

Name : Ekky Rino Fajar Sakti
Supervisor : Dr. Achmad Choiruddin, S.Si, M.Sc
Co- supervisor : Dr. Dra. Kartika Fithriasari, M.Si

ABSTRACT

Spatio-temporal multivariate point pattern data has become increasingly common lately due to the advancement of technology for massive data collection, for example location of trees species within a forest observed at several timestamps and location of several criminal type within a region occurred on some period of times. It is very important to carry out statistical analysis of multivariate point pattern data so it can be useful for solving complex problems that often arise in the real world. One statistical analysis that can be carried out is parameter estimation to determine the distributional pattern of points inside multivariate point pattern data. However, performing parameter estimation using a likelihood-based method on multivariate spatio-temporal point pattern cases is very challenging due to the curse of dimensionality which increase the model complexity and the number of parameters that must be estimated. Deep learning serves as an alternative parameter estimator when conventional parametric methods is difficult to applied. It has demonstrated its ability to accurately model huge datasets exhibiting highly nonlinear patterns. Despite its potential, only a few studies have employed deep learning to tackle multivariate spatio-temporal point pattern problems. Hence, this research aims to implement deep learning as a parameter estimator for the multivariate spatio-temporal Log Gaussian Cox Process (LGCP) model. This study employs the concept of probabilistic deep learning, ensuring each estimated parameter follows a certain distribution that aligns with its assumption. The probabilistic deep learning model is evaluated in a simulation study. In the application study, probabilistic deep learning model is employed as parameter estimator for the Barro Colorado Island (BCI) dataset. This dataset contains point pattern data for hundreds of tree species observed at various time points. The analysis in simulation study shows that probabilistic deep learning achieve 68% faster training time, lower MSE value by up to 69%, and lower RMSE value on $\hat{\alpha}$ by 33%, $\hat{\phi}$ by 27%, and $\hat{\psi}$ by 46% over previous approach. Meanwhile, the analysis of application study demonstrate that our model has up to 10 times faster training time and lower MSE value by up to 50% over previous approach, demonstrating our model effectiveness to handle highly multivariate spatio-temporal point pattern data.

Keywords: *LGCP, Multivariate Point Pattern, Neural Network, Probabilistic Deep Learning*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala limpahan Rahmat dan hidayatnya sehingga penulis dapat menyelesaikan penulisan buku tesis yang berjudul **“Estimasi Parameter pada Model *Highly Multivariate Spatio-Temporal Log Gaussian Cox Process* menggunakan Probabilistic Deep Learning”**. Tesis ini disusun sebagai salah satu syarat untuk memperoleh gelar magister pada program studi Magister Statistika, Departemen Statistika, Institut Teknologi Sepuluh Nopember, Surabaya.

Selesainya penyusunan tesis ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh sebab itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Kedua orang tua penulis dan pendamping terkasih penulis yang tanpa lelah memberikan semangat dan doa kepada penulis dalam menyelesaikan studi magister ini.
2. Bapak Dr. Achmad Choiruddin, S.Si., M.Sc. selaku Dosen Pembimbing I, Ibu Dr. Dra. Kartika Fithriasari, M.Si. selaku Dosen Pembimbing II, dan Ibu Widhianingsih Tintrim Dwi Ary, S.Si., M.Stat., Ph.D. yang senantiasa membimbing, memberikan arahan, dan petunjuk dalam penyelesaian tesis ini.
3. Bapak Dr. Sutikno, S.Si., M.Si. selaku Dosen Penguji I dan Ibu Dr. Hidayatul Khusna, S.Si. selaku Dosen Penguji II atas segala masukan dan saran untuk kebaikan tesis ini.
4. Ibu Dr. Dra. Kartika Fithriasari, M.Si. selaku Kepala Departemen Statistika ITS, Ibu Santi Wulan Purnami, M.Si., Ph.D. selaku Sekretaris Departemen I (Bidang Akademik, Kemahasiswaan, Penelitian dan Pengabdian Kepada Masyarakat), Ibu Prof. Dr. Vita Ratnasari, M.Si. selaku Sekretaris Departemen II (Bidang Sumber Daya Keuangan, Sumber Daya Manusia, dan Sarana Prasarana), serta Bapak Dr.rer.pol. Dedy Dwi Prastyo, S.Si., M.Si. selaku Kaprodi Pascasarjana Statistika ITS yang telah memfasilitasi sarana dan prasarana selama kegiatan perkuliahan dan penyusunan tesis penulis.

5. Ibu Prof. Dr. Vita Ratnasari, S.Si., M.Si. selaku dosen wali penulis yang memberikan arahan dan masukan terkait rencana studi selama penulis menempuh pendidikan magister.
6. Seluruh Bapak/Ibu dosen pengajar Statistika ITS yang telah membagikan ilmu dan pengalaman kepada penulis selama masa perkuliahan.
7. Teman-teman seperjuangan mahasiswa S2 Statistika angkatan Ganjil 2022 dan rekan-rekan kerja di PT. SML Surabaya atas segala bantuan dan kebersamaan selama penulis menjalani perkuliahan.
8. Semua pihak lainnya yang telah membantu penulis dalam penyusunan tesis ini namun tidak dapat penulis sebutkan satu persatu.

Mengingat adanya keterbatasan penulis dalam menyusun dan menyelesaikan tesis ini, maka penulis sangat terbuka terhadap kritik, saran, dan masukan membangun dari semua pihak demi kebaikan tesis ini. Penulis berharap tesis ini dapat memberikan manfaat dan menambah wawasan keilmuan pembaca.

Malang, April 2024

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN TESIS.....	v
ABSTRAK.....	vii
ABSTRACT.....	ix
KATA PENGANTAR	xi
DAFTAR ISI.....	xiii
DAFTAR TABEL.....	xv
DAFTAR GAMBAR	xvii
DAFTAR NOTASI.....	xxi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	6
1.3 Tujuan Penelitian	6
1.4 Manfaat Penelitian	6
1.5 Batasan Penelitian	6
BAB 2 TINJAUAN PUSTAKA	9
2.1 Multivariate Spatio-Temporal Point Process	9
2.2 Multivariate Spatio-Temporal log-Gaussian Cox Process.....	12
2.3 Interpretasi Model.....	16
2.3.1 Hierarchical Clustering	16
2.3.2 Korelasi Antar Spesies	18
2.4 Neural Network.....	19
2.4.1 RMSProp.....	21
2.4.2 Fungsi Aktivasi	26
2.5 Deep Learning.....	29
2.5.1 Dropout pada Deep Learning	32
2.5.2 Variational Autoencoder	34
2.6 Probabilistic Deep Learning	35
2.7 Mean Squared Error.....	42
BAB 3 METODOLOGI PENELITIAN.....	45
3.1 Desain Pengembangan Model.....	45
3.1.1 Diskretisasi.....	46
3.1.2 Multivariate LGCP menggunakan Neural Network.....	48

3.1.3	Pembagian Dataset.....	54
3.2	Studi Simulasi.....	54
3.2.1	Sumber Data dan Variabel Penelitian.....	55
3.2.2	Struktur Data.....	58
3.2.3	Proses Pelatihan.....	59
3.2.4	Tahapan Analisis pada Studi Simulasi	60
3.3	Studi Terapan.....	61
3.3.1	Sumber Data dan Variabel Penelitian.....	61
3.3.2	Struktur Data.....	64
3.3.3	Proses Pelatihan.....	66
3.3.4	Interpretasi Model.....	66
3.3.5	Tahapan Analisis pada Studi Terapan	67
BAB 4	HASIL DAN PEMBAHASAN.....	69
4.1	Arsitektur Model	69
4.1.1	Model untuk Studi Simulasi	70
4.1.2	Model untuk Studi Terapan	71
4.2	Studi Simulasi.....	82
4.2.1	Pembangkitan <i>Point Pattern</i>	82
4.2.2	Diskretisasi <i>Point Pattern</i>	83
4.2.3	Pembagian Dataset dan Pelatihan Model	84
4.2.4	Hasil Evaluasi Model.....	85
4.3	Studi Terapan	89
4.3.1	Pengumpulan Dataset	89
4.3.2	Diskretisasi <i>Point Pattern</i> dan Variabel Covariate.....	90
4.3.3	Pembagian Dataset dan Pelatihan Model	91
4.3.4	Hasil Evaluasi Model.....	93
4.3.5	Interpretasi Model.....	97
BAB 5	KESIMPULAN DAN SARAN.....	111
5.1	Kesimpulan.....	111
5.2	Saran.....	112
	DAFTAR PUSTAKA.....	115
	LAMPIRAN... ..	119
	BIOGRAFI PENULIS.....	161

DAFTAR TABEL

Tabel 3.1	Struktur Data Bangkitan pada Studi Simulasi	58
Tabel 3.2	Struktur Data <i>Point Pattern</i> Setelah Proses Diskretisasi pada Data Simulasi.....	59
Tabel 3.3	Penjelasan tentang Variabel <i>Covariate</i>	63
Tabel 3.4	Struktur Data <i>Point Pattern</i> pada BCI Dataset.....	64
Tabel 3.5	Struktur Data <i>Point Pattern</i> Setelah Proses Diskretisasi pada <i>Dataset BCI</i>	65
Tabel 3.6	Struktur Data Variabel <i>Covariate</i> Setelah Proses Diskretisasi pada <i>Dataset BCI</i>	66
Tabel 4.1	Arsitektur Model pada Studi Terapan	71
Tabel 4.2	Contoh Struktur Data Variabel <i>Dummy</i>	81
Tabel 4.3	Perbandingan Waktu Pelatihan dan MSE antara Model Mateu & Jalilian (2022) dan <i>Probabilistic Deep Learning</i> dari 200 Iterasi	86
Tabel 4.4	Perbandingan RMSE pada Metode Choiruddin dkk (2020), Mateu & Jalilian (2022) dan <i>Probabilistic Deep Learning</i> dari 200 Iterasi	87
Tabel 4.5	Perbandingan Waktu Estimasi pada Choiruddin dkk (2020), Mateu & Jalilian (2022), dan <i>Probabilistic Deep Learning</i>	88
Tabel 4.6	Struktur Data Pencatatan Nilai MSE dan Waktu Pelatihan pada Setiap Iterasi.....	94
Tabel 4.7	Perbandingan Waktu Pelatihan dan MSE antara Model Mateu & Jalilian (2022) dan <i>Probabilistic Deep Learning</i> dari 100 Iterasi dengan Jumlah Spesies 25 dan Waktu Observasi 4.....	94
Tabel 4.8	Perbandingan Waktu Pelatihan dan MSE antara Model Mateu & Jalilian (2022) dan <i>Probabilistic Deep Learning</i> dari 5 Iterasi dengan Jumlah Spesies 100 dan Waktu Observasi 8...	96

Tabel 4.9	Hasil Estimasi Λ untuk 25 Spesies dengan 4 Waktu Observasi	98
Tabel 4.10	Hasil Estimasi μ untuk 25 Spesies dengan 4 Waktu Observasi	99
Tabel 4.11	Hasil Estimasi σ untuk 25 Spesies dengan 4 Waktu Observasi	101
Tabel 4.12	Hasil Estimasi PV untuk 25 Spesies dengan 4 Waktu Observasi.....	102
Tabel 4.13	Distribusi (Dalam Persen) Estimasi Nilai Korelasi Antar 25 Spesies dengan Waktu Observasi 4	104
Tabel 4.14	Distribusi Estimasi Λ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8	104
Tabel 4.15	Distribusi Estimasi μ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8	105
Tabel 4.16	Distribusi Jarak Euclid (Dalam Persen) antara 100 Spesies ...	106
Tabel 4.17	Distribusi Estimasi σ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8	107
Tabel 4.18	Distribusi Estimasi PV (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8	108
Tabel 4.19	Distribusi (Dalam Persen) Estimasi Nilai Korelasi Antar 100 Spesies dengan Waktu Observasi 8	109

DAFTAR GAMBAR

Gambar 1.1	Peta Penelitian	5
Gambar 2.1	Contoh 10 Realisasi <i>Stationary Point Process</i> dengan Nilai Intensitas yang Serupa	10
Gambar 2.2	Contoh <i>Multivariate Spatio-Temporal Point Pattern</i> yang Merepresentasikan Lokasi Pohon pada 4 Spesies yang Diobservasi pada 8 Waktu di Suatu Hutan (Condit dkk, 2019)	11
Gambar 2.3	Ilustrasi Cox process. (a) Realisasi <i>Random Intensity Function</i> $\Lambda(u)$. (b) Realisasi <i>Poisson point process</i> berdasarkan $\Lambda(u)$. (c) Realisasi titik berdasarkan $\Lambda(u)$. (Baddeley dkk, 2015)	13
Gambar 2.4	Perbandingan Struktur antara (a) <i>Biological Neural Network</i> dan (b) <i>Artificial Neural Network</i> (Aggarwal, 2018)	19
Gambar 2.5	Contoh <i>Neural Network</i> dengan <i>Hidden Layer</i> (Aggarwal, 2018).....	21
Gambar 2.6	Contoh Arsitektur <i>Neural Network</i>	22
Gambar 2.7	Fungsi Aktivasi <i>Sigmoid</i>	27
Gambar 2.8	Fungsi Aktivasi <i>Softplus</i>	27
Gambar 2.9	Fungsi Aktivasi ReLU	28
Gambar 2.10	Fungsi Aktivasi <i>Tanh</i>	29
Gambar 2.11	Fungsi Eksponensial	29
Gambar 2.12	Contoh Arsitektur <i>Deep Learning</i> (Tama dkk, 2023)	30
Gambar 2.13	Perbandingan Performa <i>Deep Learning</i> dan Metode <i>Machine Learning</i> Lain Berdasarkan Jumlah Data (Aggarwal, 2018)..	31
Gambar 2.14	Perbandingan Hasil Pemodelan antara Model <i>Underfit</i> (a), <i>Good Fit</i> (b), dan Model <i>Overfit</i> (c) (Ghojogh & Crowley, 2019).....	33
Gambar 2.15	Arsitektur <i>Variational Autoencoder</i>	34

Gambar 2.16 Contoh Arsitektur <i>Probabilistic Deep Learning</i> (Chang, 2021)	36
Gambar 2.17 Contoh Penerapan <i>Probabilistic Deep Learning</i> (Dürr dkk, 2020)	36
Gambar 2.18 Penerapan <i>Backpropagation</i> pada <i>Probabilistic Deep Learning</i>	37
Gambar 3.1 Ilustrasi <i>Probabilistic Deep Learning</i> untuk <i>Spatio-Temporal</i> LGCP	45
Gambar 3.2 Ilustrasi Diskretisasi pada (a) <i>Point Pattern</i> , (b) Variabel <i>Covariate</i> . Hasil diskretisasi digunakan sebagai <i>Input</i> pada Model Deep Learning.	47
Gambar 3.3 Model <i>Multivariate</i> LGCP menggunakan <i>Neural Network</i>	50
Gambar 3.4 Ilustrasi <i>Probabilistic Deep Learning</i> pada Studi Simulasi	60
Gambar 3.5 Diagram Alir Tahapan Analisis pada Studi Simulasi	61
Gambar 3.6 Contoh <i>Point Pattern</i> dari 4 Spesies Pohon	62
Gambar 3.7 Variabel <i>Covariate</i> yang Tersedia pada BCI Dataset	62
Gambar 3.8 Diagram Alir Tahapan Analisis pada Studi Terapan	67
Gambar 4.1 Langkah Pemilihan Arsitektur Model	70
Gambar 4.2 Realisasi 6 <i>Point Pattern</i> Bangkitan pada Model <i>Multivariate</i> LGCP untuk Tipe Pertama	82
Gambar 4.3 Realisasi <i>Point Pattern</i> Bangkitan pada Model <i>Multivariate</i> LGCP untuk Semua Tipe	83
Gambar 4.4 Plot Hasil Diskretisasi pada Data Bangkitan	83
Gambar 4.5 Grafik MSE Pelatihan Model pada Studi Simulasi	84
Gambar 4.6 Perbandingan Hasil Estimasi $\Lambda_{ij}(s)$ pada Salah Satu Iterasi Model di Studi Simulasi	87
Gambar 4.7 Contoh Plot Hasil Diskretisasi pada 2 Spesies dan 2 Waktu Observasi	90

Gambar 4.8 Grafik MSE Pelatihan Model dengan Jumlah Spesies 25 dan Waktu Observasi 4 pada Studi Terapan	92
Gambar 4.9 Grafik MSE Pelatihan Model dengan Jumlah Spesies 100 dan Waktu Observasi 8 pada Studi Terapan	92
Gambar 4.10 Hasil Estimasi $\Lambda_{ij}(s, t)$ dengan Jumlah Spesies 25 dan Waktu Observasi 4 pada Salah Satu Iterasi	95
Gambar 4.11 Hasil Estimasi $\Lambda_{ij}(s, t)$ dengan Jumlah Spesies 100 dan Waktu Observasi 8 pada Salah Satu Iterasi	97
Gambar 4.12 <i>Heatmap Plot</i> yang Menunjukkan Jarak Euclid Antar 25 Spesies	100
Gambar 4.13 <i>Heatmap Plot</i> yang Menunjukkan Korelasi Antar 25 Spesies dengan 4 Waktu Observasi pada Tahun 1995	103

(Halaman ini sengaja dikosongkan)

DAFTAR NOTASI

a	:	Jumlah <i>latent field</i>
α	:	Salah satu parameter pada model <i>multivariate</i> LGCP dan <i>multivariate spatio-temporal</i> LGCP
b'	:	Ukuran <i>batch</i> pada pelatihan <i>deep learning</i>
d	:	Dimensi suatu ruang
x	:	<i>Spatial point pattern</i>
X	:	<i>Spatial point process</i>
\check{X}	:	<i>Multivariate point process</i>
$N(\cdot)$:	Fungsi yang menyatakan jumlah titik pada <i>point pattern</i>
u	:	Vektor titik koordinat
v	:	Vektor titik koordinat
A	:	Suatu bidang berdimensi $\mathbb{R}^2 \times \mathbb{R}^2$
T	:	Jumlah waktu observasi
m	:	Jumlah tipe <i>multivariate</i>
t	:	Indeks waktu observasi, $t = 1, 2, \dots, T$
s	:	Indeks tipe <i>multivariate</i> , $s = 1, 2, \dots, m$
l	:	Indeks <i>latent field</i> , $l = 1, 2, \dots, a$
$x^{(s)}$:	<i>Spatial point pattern</i> untuk tipe <i>multivariate</i> ke- s
$x^{(s,t)}$:	<i>Spatial point pattern</i> untuk tipe <i>multivariate</i> ke- s dan waktu observasi ke- t
$X^{(s)}$:	<i>Spatial point process</i> untuk tipe <i>multivariate</i> ke- s
$X^{(s,t)}$:	<i>Spatial point process</i> untuk tipe <i>multivariate</i> ke- s dan waktu observasi ke- t
$\lambda(u)$:	Nilai intensitas pada titik koordinat u (peluang)
$o(\cdot)$:	<i>Pair correlation function</i>
z	:	Variabel <i>covariate</i>
p	:	Jumlah variabel <i>covariate</i>
c	:	Indeks variabel <i>covariate</i> , $c = 1, 2, \dots, p$

$\Lambda(u)$:	Nilai intensitas yang bersifat random pada koordinat u
$\Psi(u)$:	<i>Gaussian random field</i> pada koordinat u untuk kasus <i>univariate</i> LGCP
$\mathbf{F}^{(s)}(u)$:	<i>Gaussian random field</i> pada koordinat u yang disebabkan karena faktor eksternal yang tidak dapat dijelaskan (<i>unobservable factor</i>)
$\mathbf{K}^{(s)}(u)$:	<i>Gaussian random field</i> pada koordinat u yang disebabkan karena faktor internal setiap tipe <i>multivariate</i>
d_1 dan d_2	:	Ukuran diskretisasi $d_1 \times d_2$
i	:	Indeks <i>grid</i> , $i = 1, 2, \dots, d_1$
j	:	Indeks <i>grid</i> , $j = 1, 2, \dots, d_2$
B_{ij}	:	<i>Grid</i> ke- i dan ke- j
W	:	<i>Observation window</i>
$\mathbf{E}^{(l)}(u)$:	<i>Gaussian random field</i> pada koordinat u dan <i>latent field</i> ke- l hasil dekomposisi dari $\mathbf{F}^{(s)}(u)$
$\mathbf{E}_{ij}^{(l)}$:	<i>Gaussian random field</i> pada <i>grid</i> B_{ij} dan <i>latent field</i> ke- l hasil dekomposisi dari $\mathbf{F}_{ij}^{(s)}$
$\mathbf{E}_{ij}^{(l,t)}$:	<i>Gaussian random field</i> pada <i>grid</i> B_{ij} , <i>latent field</i> ke- l , dan waktu observasi ke- t hasil dekomposisi dari $\mathbf{F}_{ij}^{(s,t)}$
$\mathbf{V}^{(s)}(u)$:	<i>Gaussian random field</i> pada koordinat u dan tipe <i>multivariate</i> ke- s hasil dekomposisi dari $\mathbf{K}^{(s)}(u)$
$\mathbf{V}_{ij}^{(s)}$:	<i>Gaussian random field</i> pada koordinat u dan tipe <i>multivariate</i> ke- s hasil dekomposisi dari $\mathbf{K}_{ij}^{(s)}$
$\mathbf{V}_{ij}^{(s,t)}$:	<i>Gaussian random field</i> pada koordinat u , tipe <i>multivariate</i> ke- s , dan waktu observasi ke- t hasil dekomposisi dari $\mathbf{K}_{ij}^{(s,t)}$
$\mu^{(s)}(u)$:	Salah satu parameter <i>multivariate</i> LGCP pada koordinat u dan tipe <i>multivariate</i> ke- s
$\mu_{ij}^{(s)}$:	Salah satu parameter <i>multivariate</i> LGCP pada <i>grid</i> B_{ij} dan tipe <i>multivariate</i> ke- s
$\mu^{(s,t)}(u)$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP pada koordinat u , tipe <i>multivariate</i> ke- s , dan waktu observasi ke- t

$\mu_{ij}^{(s,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP pada <i>grid</i> B_{ij} , tipe <i>multivariate</i> ke- s , dan waktu observasi ke- t
$\sigma^{(s)}$:	Salah satu parameter <i>multivariate</i> LGCP pada tipe <i>multivariate</i> ke- s
$\sigma^{(s,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP pada tipe <i>multivariate</i> ke- s dan waktu observasi ke- t
$\phi^{(l)}$:	Salah satu parameter <i>multivariate</i> LGCP untuk <i>latent field</i> ke- l
$\psi^{(s)}$:	Salah satu parameter <i>multivariate</i> LGCP untuk tipe <i>multivariate</i> ke- s
$\phi^{(l,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP untuk <i>latent field</i> ke- l dan waktu observasi ke- t
$\varphi^{(l,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP untuk <i>latent field</i> ke- l dan waktu observasi ke- t
$\rho^{(l,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP untuk <i>latent field</i> ke- l dan waktu observasi ke- t
$\psi^{(s,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP untuk tipe <i>multivariate</i> ke- s dan waktu observasi ke- t
$\zeta^{(s,t)}$:	Salah satu parameter <i>multivariate spatio-temporal</i> LGCP untuk tipe <i>multivariate</i> ke- s dan waktu observasi ke- t
Σ_E	:	Fungsi covariance untuk <i>Gaussian random field</i> E
Σ_V	:	Fungsi covariance untuk <i>Gaussian random field</i> V
A	:	<i>Lower triangular matrix</i> untuk Σ_E
B	:	<i>Lower triangular matrix</i> untuk Σ_V
w	:	Matriks bobot pada <i>neural network</i> atau <i>deep learning</i>
b	:	Bias pada <i>neural network</i> atau <i>deep learning</i>
\hat{h}	:	Output <i>hidden layer</i> pada <i>neural network</i> atau <i>deep learning</i>
\mathcal{X}	:	Input data untuk <i>neural network</i> atau <i>deep learning</i>
$f_{\text{sig}}(\cdot)$:	Fungsi aktivasi sigmoid
$f_{\text{exp}}(\cdot)$:	Fungsi aktivasi eksponensial
$f_{\text{spplus}}(\cdot)$:	Fungsi aktivasi softplus
$f_{\text{tanh}}(\cdot)$:	Fungsi aktivasi tanh

$f_{\text{relu}}(\cdot)$: Fungsi aktivasi relu
$\mathcal{L}(\cdot)$: Jumlah <i>neuron</i> pada suatu <i>hidden layer</i>
$n(\cdot)$: Urutan <i>hidden layer</i> pada suatu <i>deep learning</i>
$\boldsymbol{\theta}_{sim}$: Sekumpulan parameter <i>multivariate</i> LGCP, $\boldsymbol{\theta}_{bci} = \{\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\phi}, \boldsymbol{\sigma}, \boldsymbol{\psi}\}$
$\boldsymbol{\theta}_{bci}$: Sekumpulan parameter <i>multivariate spatio-temporal</i> LGCP, $\boldsymbol{\theta}_{bci} = \{\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\phi}, \boldsymbol{\varphi}, \boldsymbol{\rho}, \boldsymbol{\sigma}, \boldsymbol{\psi}, \boldsymbol{\zeta}\}$
\mathcal{D}_{sim}	: Point pattern bangkitan untuk studi simulasi
$\tilde{\mathcal{D}}_{sim}$: Point pattern bangkitan \mathcal{D}_{sim} yang telah didiskretisasi
\mathcal{D}_{bci}	: Point pattern dataset BCI untuk studi terapan
$\tilde{\mathcal{D}}_{bci}$: Point pattern \mathcal{D}_{bci} yang telah didiskretisasi
\mathcal{Z}_{bci}	: Variabel <i>covariate</i> dataset BCI untuk studi terapan
$\tilde{\mathcal{Z}}_{bci}$: Variabel <i>covariate</i> \mathcal{Z}_{bci} yang telah didiskretisasi
\mathcal{T}_{bci}	: Waktu observasi pada dataset BCI untuk studi terapan
\mathbf{D}	: Matriks yang berisi variabel <i>dummy</i> untuk \mathcal{T}_{bci}

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Spatio-temporal point process merupakan salah satu model yang digunakan untuk melakukan analisis statistik terhadap sekumpulan titik (*point*) yang tersebar secara acak pada suatu bidang spasial dan pada interval waktu (Baddeley, 2007). Titik-titik tersebut merepresentasikan koordinat lokasi dan waktu suatu kejadian atau obyek yang terjadi pada sebuah bidang, misalnya lokasi setiap pohon pada suatu hutan yang diobservasi pada beberapa waktu dan lokasi terjadinya gempa bumi selama beberapa tahun. Data sebaran titik tersebut disebut sebagai data *spatio-temporal point pattern*.

Perkembangan teknologi yang sangat pesat akhir-akhir ini menyebabkan pengambilan data menjadi lebih mudah. Oleh sebab itu, data-data berukuran besar menjadi lebih sering ditemukan, contohnya adalah pengambilan data geografi menggunakan satelit yang diambil secara berkala pada setiap interval waktu tertentu (Choiruddin, 2021). Dalam bidang ekologi, pengambilan data secara berkala pada banyak variabel juga sering dilakukan sehingga volume datanya menjadi sangat besar. Data ekologi berukuran masif tersebut dapat direpresentasikan sebagai *point pattern* yang terdiri dari banyak tipe dan repetisi waktu. Salah satu contoh *dataset* ekologi berukuran besar adalah *dataset* Barro Colorado Island (BCI) yang berisi koordinat lokasi ratusan spesies pohon, yang direpresentasikan sebagai tipe *multivariate*, pada area hutan hujan tropis seluas 50 hektar di Pulau Barro Colorado, Panama (Condit, 2019). Lokasi setiap spesies pohon diambil setiap 5 tahun sekali sejak tahun 1985 hingga 2015 sehingga *dataset* ini juga mengandung unsur waktu (*temporal*). Analisis pada data *spatio-temporal point pattern* yang memiliki banyak tipe *multivariate* dan waktu menjadi penting dilakukan karena pola persebaran titik serta hubungannya antar spesies dan waktu dapat diketahui dengan baik. Meskipun memiliki kebermanfaatan yang signifikan, analisis pada data *multivariate spatio-temporal point pattern* merupakan pekerjaan yang tidak mudah karena tingginya dimensi data yang digunakan sehingga model yang dikembangkan menjadi sangat kompleks dan jumlah parameter yang

diestimasi juga semakin banyak. Penelitian-penelitian terdahulu telah dilakukan untuk menganalisis data *multivariate point pattern* menggunakan model *multivariate LGCP*. Choiruddin dkk (2020) memperkenalkan *least square estimator* yang diregularisasi untuk melakukan estimasi parameter pada 86 spesies pohon di hutan hujan tropis. Penelitian lainnya oleh Husain & Choiruddin (2021) bertujuan untuk mengubah struktur model intensitas sehingga proses estimasi parameter menjadi semakin efisien. Penelitian tersebut diterapkan pada 9 spesies pohon yang dipengaruhi oleh 11 variabel *covariate* pada suatu hutan. Penelitian lainnya bertujuan untuk melakukan estimasi fungsi intensitas menggunakan metode *quasi-likelihood* (Guan dkk, 2015) dan estimasi *second-order properties* menggunakan metode *conditional composite likelihood* (Hessellund dkk, 2022) yang memiliki komputasi lebih efisien daripada metode *likelihood* standar. Kedua penelitian tersebut menerapkan *estimator* untuk menganalisis kurang dari 10 spesies pohon. Penelitian-penelitian tersebut menerapkan *estimator* parametrik pada kasus *multivariate* berdimensi kecil hingga puluhan tipe *multivariate* tanpa mengikutsertakan unsur *temporal*. Sejauh pemahaman peneliti, hingga saat ini belum ada penelitian yang menerapkan metode parametrik pada ratusan tipe *multivariate* sekaligus dengan unsur *temporal*. Hal tersebut menunjukkan bahwa penerapan metode parametrik pada kasus *highly multivariate* sangat sulit dilakukan karena semakin tinggi dimensi tipe *multivariate* dan waktu, maka jumlah parameter yang harus diestimasi menjadi semakin besar yang mengakibatkan pendekatan parametrik menjadi semakin rumit dan beban komputasinya meningkat secara signifikan (Choiruddin dkk, 2020; Rajala dkk, 2018; Mateu & Jalilian, 2022). Oleh sebab itu, penelitian ini bertujuan untuk mengembangkan *estimator* yang lebih fleksibel dan memiliki komputasi yang lebih efisien sehingga dapat digunakan pada kasus *multivariate* yang berdimensi sangat tinggi.

Salah satu alternatif *estimator* parameter ketika metode *estimator* parametrik tidak dapat dipakai adalah *neural network* (Lenzi dkk, 2023). *Neural network* merupakan sebuah metode yang terinspirasi dari mekanisme dan proses pembelajaran yang terjadi pada otak makhluk hidup (Aggarwal, 2018). Bentuk *neural network* yang paling sederhana adalah *single layer perceptron*. Pada model ini, *neural network* hanya dapat dipakai untuk melakukan klasifikasi secara linear.

Beberapa tahun kemudian, penelitian lain menyebutkan bahwa *neural network* dengan dua atau lebih *hidden layer* memiliki performa yang lebih baik daripada *single layer perceptron* karena model ini dapat memodelkan data secara nonlinier (Singh & Banerjee, 2019). Dari perkembangan tersebut, dapat disimpulkan bahwa jumlah *hidden layer* pada *neural network* dapat ditambahkan untuk memodelkan data yang sangat kompleks dan derajat nonlinier yang tinggi. Konsep tersebut kemudian dinamakan sebagai *deep learning*. Secara formal, *deep learning* merupakan sebuah model yang terdiri dari sekumpulan *layer-layer* sederhana dan nonlinier (LeCun, 2015). *Deep learning* termasuk ke dalam kelompok *representation learning* yang berarti *deep learning* dapat secara otomatis menemukan fitur atau representasi yang tepat berdasarkan data input. Dengan demikian, sebuah model *deep learning* dapat memodelkan data yang sangat kompleks tanpa perlu intervensi dari manusia. Fleksibilitas ini memungkinkan *deep learning* untuk diterapkan pada berbagai macam data berukuran besar dan kompleks seperti pengolahan gambar. Penelitian-penelitian terdahulu telah membuktikan bahwa *deep learning* berhasil mendapatkan performa yang sangat baik pada kasus klasifikasi gambar. Dalam bidang kesehatan, salah satu penelitian mengembangkan model *deep learning* untuk melakukan diagnosa kanker kulit berdasarkan gambar luka pada kulit. Penelitian ini membuktikan bahwa model *deep learning* berhasil menyamai akurasi diagnosa yang dilakukan oleh ahli dermatologi (Esteva dkk, 2017). Penelitian-penelitian lain membuktikan bahwa *deep learning* mendapatkan hasil yang sangat baik pada kasus lain yang membutuhkan pemrosesan gambar, yaitu sistem rekomendasi (Deldjoo dkk, 2016), diagnosa kanker prostat (Nagpal dkk, 2019), dan pemantauan kondisi lingkungan (Zhao dkk, 2018).

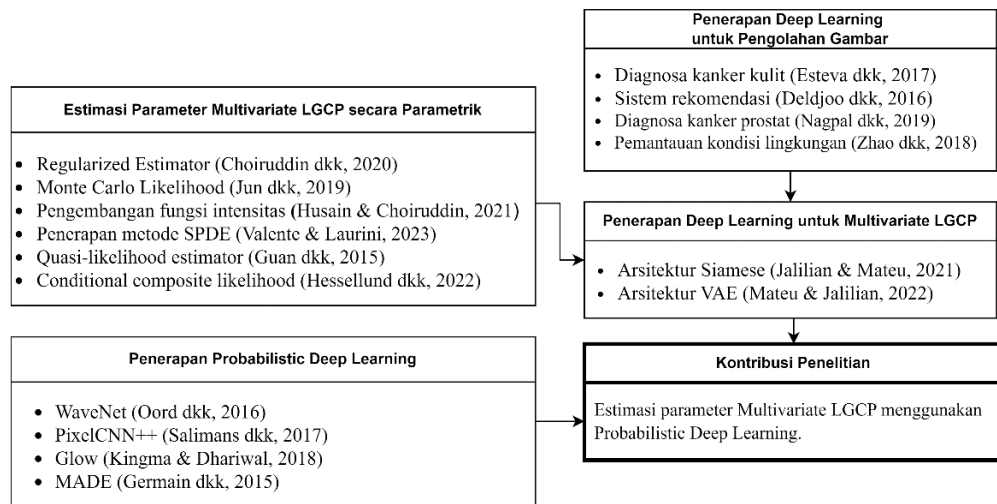
Point pattern pada ruang berdimensi dua dapat pula dilihat sebagai gambar dua dimensi yang mengandung titik-titik koordinat. Meskipun telah terbukti mendapatkan hasil yang sangat baik pada permasalahan klasifikasi gambar, hanya sedikit penelitian yang mencoba menerapkan *deep learning* untuk permasalahan *spatial point process*. Vihrs (2022) mencoba untuk melakukan estimasi parameter pada model *spatial point process* menggunakan *deep learning*. Pada penelitian tersebut, *point pattern* didapatkan dengan cara mensimulasikan model *spatial point*

process pada sekumpulan nilai parameter. *Point pattern* yang telah dibangkitkan tersebut bertindak sebagai variabel prediktor, sedangkan nilai parameter model bertindak sebagai variabel respon. Berdasarkan simulasi yang dilakukan, model *deep learning* berhasil melakukan estimasi *parameter* dengan baik. Namun, Vihrs (2022) hanya menerapkan *deep learning* sebagai *estimator* untuk kasus *univariate* dan tidak menggunakan variabel *covariate* sebagai prediktor (stasioner) sehingga model hanya menerima data input berupa *point pattern univariate*. Penelitian lainnya oleh Jalilian dan Mateu (2021) juga menerapkan *deep learning* (Arsitektur Siamese) untuk klasifikasi antar 130 spesies pohon. Selain itu, Mateu dan Jalilian (2022) menerapkan model *deep learning* yang tergolong sebagai arsitektur *variational autoencoder* (VAE) sebagai *estimator* parameter pada kasus *highly multivariate spatio-temporal point pattern*. Penelitian ini menyimpulkan bahwa *deep learning* mampu melakukan estimasi parameter dengan baik. Namun demikian, pendekatan Mateu dan Jalilian (2022) melibatkan penghitungan matriks yang sangat besar sehingga semakin besar jumlah variabel *multivariate* yang digunakan, maka komputasi yang dibutuhkan juga akan meningkat secara signifikan. Hal tersebut mengakibatkan waktu pelatihan dan prediksi menjadi sangat lama.

Penelitian ini terinspirasi oleh Mateu & Jalilian (2022) dalam menggunakan VAE sebagai *estimator* untuk model *spatio-temporal* LGCP. Namun demikian, penelitian ini merevisi VAE pada model yang dikembangkan oleh Mateu & Jalilian (2022) dengan mempertimbangkan konsep *probabilistic deep learning*. Berbeda dengan *deep learning* konvensional yang mengeluarkan sebuah nilai output sebagai estimasi titik, *probabilistic deep learning* mengeluarkan estimasi berupa suatu distribusi peluang tertentu dengan memanfaatkan *probabilistic layer* (Dürr dkk, 2020). Model *probabilistic deep learning* dapat digunakan untuk membangkitkan sampel data baru berdasarkan distribusi yang diestimasi, misalnya untuk membangkitkan gambar (Kingma & Welling, 2014). Beberapa penelitian lain telah menerapkan *probabilistic deep learning* pada kasus kompleks yang melibatkan estimasi suatu distribusi, yaitu WaveNet dan PixelCNN++ untuk melakukan estimasi distribusi pada model *autoregressive* (Oord dkk, 2016; Salimans dkk, 2017), model Glow untuk membangkitkan gambar wajah berdasarkan fitur

berdistribusi *multivariate* normal (Kingma & Dhariwal, 2018), dan model MADE untuk melakukan estimasi suatu distribusi berdasarkan sekumpulan data input (Germain dkk, 2015).

Pada penelitian ini, setiap parameter yang ingin diestimasi dimodelkan menggunakan *probabilistic layer* sehingga setiap parameter tersebut tetap mengikuti distribusi yang sesuai dengan asumsinya. Dengan demikian, model yang dikembangkan tidak perlu melakukan proses penghitungan matriks berukuran besar yang mengakibatkan model menjadi lebih efisien daripada pendekatan Mateu dan Jalilian (2022). Adapun peta penelitian ditunjukkan pada Gambar 1.1.



Gambar 1.1 Peta Penelitian

Model yang dikembangkan dievaluasi pada studi simulasi dan diterapkan pada dataset sekunder untuk studi terapan. Studi simulasi dilakukan dengan membangkitkan data *multivariate point pattern* berdasarkan nilai parameter pada model *multivariate* LGCP yang telah ditentukan sebelumnya. Data simulasi dibangkitkan tanpa terdapat unsur *temporal* karena keterbatasan dalam membangkitkan data *multivariate point pattern* yang mengandung unsur waktu dan untuk mengurangi beban komputasi. Sementara itu, *dataset* sekunder yang digunakan di dalam penelitian ini adalah *dataset* BCI. Selain terdiri dari ratusan tipe *multivariate*, *dataset* ini juga menyediakan 13 variabel *covariate* yang mencerminkan kondisi tanah pada area hutan seluas 50 hektar (Detail dataset dijelaskan pada Bab 3.3).

1.2 Perumusan Masalah

Perumusan masalah berdasarkan latar belakang penelitian adalah:

1. Bagaimana arsitektur *probabilistic deep learning* untuk estimasi parameter pada data *highly multivariate spatio-temporal point pattern*?
2. Bagaimana evaluasi model *probabilistic deep learning* untuk estimasi parameter *multivariate* LGCP pada studi simulasi?
3. Bagaimana penerapan model *probabilistic deep learning* untuk mengetahui pola persebaran banyak spesies pohon di Pulau Barro Colorado, Panama?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, tujuan yang ingin dicapai adalah:

1. Mengembangkan arsitektur *probabilistic deep learning* untuk estimasi parameter pada data *highly multivariate spatio-temporal point pattern*.
2. Mengevaluasi model *probabilistic deep learning* untuk estimasi parameter *multivariate* LGCP pada studi simulasi.
3. Menerapkan model *probabilistic deep learning* untuk mengetahui pola persebaran banyak spesies pohon di Pulau Barro Colorado, Panama.

1.4 Manfaat Penelitian

Dari segi akademis, penelitian ini bermanfaat sebagai referensi bahwa model *deep learning* dapat digunakan sebagai *estimator* parameter untuk data *highly multivariate spatio-temporal point pattern*, sehingga peneliti lain dapat mengembangkan dan menerapkan *deep learning* pada permasalahan kompleks lain di bidang statistika. Dari segi praktis, hasil estimasi parameter pada penelitian ini dapat bermanfaat sebagai pendukung pengambilan keputusan oleh pihak-pihak yang berkepentingan.

1.5 Batasan Penelitian

Batasan penelitian yang digunakan di dalam penelitian ini adalah:

1. Penelitian ini fokus untuk menganalisis 25 dan 100 spesies pohon yang diobservasi pada 4 dan 8 kali sensus di hutan Pulau Barro Colorado.
2. Studi simulasi menggunakan *multivariate* spasial tanpa unsur *temporal* karena:

- a) keterbatasan dalam membangkitkan *multivariate spatio-temporal point pattern*,
 - b) mengurangi beban komputasi.
3. Data simulasi yang dibangkitkan memiliki jumlah tipe *multivariate* sebanyak 10 tipe.
 4. Fungsi aktivasi yang digunakan adalah fungsi *sigmoid*, *softplus*, *tanh*, dan eksponensial.
 5. Pelatihan *deep learning* dilakukan sebanyak 50 kali perulangan (*epoch*).
 6. Jumlah *latent field* yang dipakai pada studi simulasi adalah 4.
 7. Jumlah *latent field* yang dipakai pada studi terapan adalah 5.
 8. Kebaikan model dan *loss function* yang dipakai adalah MSE dan RMSE.

(Halaman ini sengaja dikosongkan)

BAB 2

TINJAUAN PUSTAKA

2.1 Multivariate Spatio-Temporal Point Process

Spatio-temporal point process merupakan sebuah model statistika yang digunakan untuk menganalisis sebaran titik (*point*) pada suatu bidang spasial yang diobservasi interval waktu. *Spatio-temporal point process* \mathbf{X} bersifat acak yang memiliki *output* berupa *spatio-temporal point pattern* $\mathbf{x} = \{x_1, \dots, x_{N(\mathbf{X})}\}$, yaitu suatu bidang berdimensi d yang mengandung titik-titik koordinat sejumlah $N(\mathbf{X})$. Setiap titik tersebut merepresentasikan lokasi terjadinya suatu *event* yang terjadi pada suatu waktu, misalkan lokasi pohon pada sebuah hutan pada suatu tahun, lokasi pusat kebakaran pada suatu bulan, atau lokasi tindakan kriminal pada suatu tahun tertentu (Baddeley, 2007).

Dalam bidang statistika, jumlah titik di dalam suatu *point pattern* bersifat *finite*. Dengan kata lain, sebuah *finite spatio-temporal point process* \mathbf{X} memiliki 2 karakteristik, yaitu (1) setiap realisasi \mathbf{X} adalah sebuah *point pattern* yang mengandung jumlah titik yang *finite*, (2) pada setiap *observation window* W , jumlah titik yang ada di dalam W , atau $N(\mathbf{X} \cap W)$ adalah *random variable*.

Jika diketahui $N(\mathbf{X} \cap W) = N(W)$, dimana $W = S \times T$, $S \subseteq \mathbb{R}^2$, $T \subset \mathbb{N}$, maka *first order moment property* atau nilai ekspektasi dari jumlah titik yang diobservasi pada W adalah:

$$\mathbb{E}[N(W)] = \mu(W) = \sum_{t=1}^T \int_W \lambda(u) du dt \quad (2.1)$$

Dimana $\lambda(u)$ adalah fungsi intensitas yang diinterpretasikan sebagai peluang mendapatkan sebuah titik pada titik koordinat $u = (u_1, u_2)$, area du , dan dt (Møller & Waagepetersen, 2004). Pada teori statistika, hubungan antar variable diestimasi menggunakan fungsi *covariance*. Serupa dengan teori tersebut, hubungan antar 2 titik dapat dihitung menggunakan konsep *covariance* sebagai *second-order moment measure* $\mu^{(2)}$ yang didefinisikan sebagai:

$$\mu^{(2)}(A) = \mathbb{E} \sum_{t=1}^T \sum_{u,v \in X}^{\neq} \mathbb{I}[(u, v) \in A], \quad A \subseteq \mathbb{R}^2 \times \mathbb{R}^2 \times T \quad (2.2)$$

Dimana tanda \neq menunjukkan bahwa operator *sum* berjalan untuk semua pasangan titik yang berbeda u, v . Pada sebagian kasus, $\mu^{(2)}$ dapat pula didefinisikan berdasarkan nilai *second order product density* $\lambda^{(2)}$ yang telah diketahui sebelumnya.

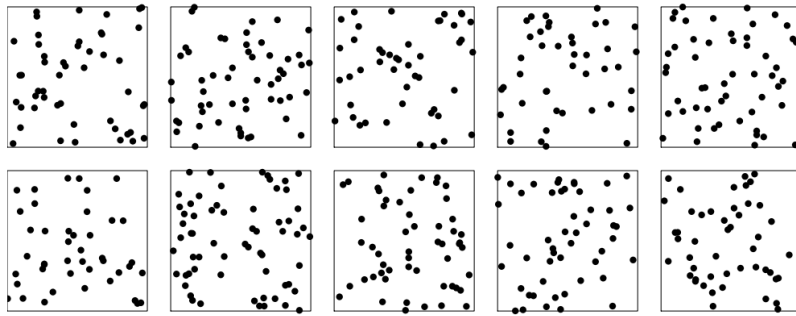
$$\mu^{(2)}(A) = \sum_{t=1}^T \int_A \mathbb{I}[(u, v) \in A] \lambda^{(2)}(u, v) du dv dt \quad (2.3)$$

dimana $\lambda^{(2)}(u, v)$ direpresentasikan sebagai peluang mendapatkan sebuah titik pada masing-masing region di lokasi u dan v serta waktu t .

Interaksi antara 2 titik dapat ditunjukkan dengan *pair correlation function* yang didefinisikan pada persamaan berikut.

$$o(u, v) = \frac{\lambda^{(2)}(u, v)}{\lambda(u)\lambda(v)} \quad (2.4)$$

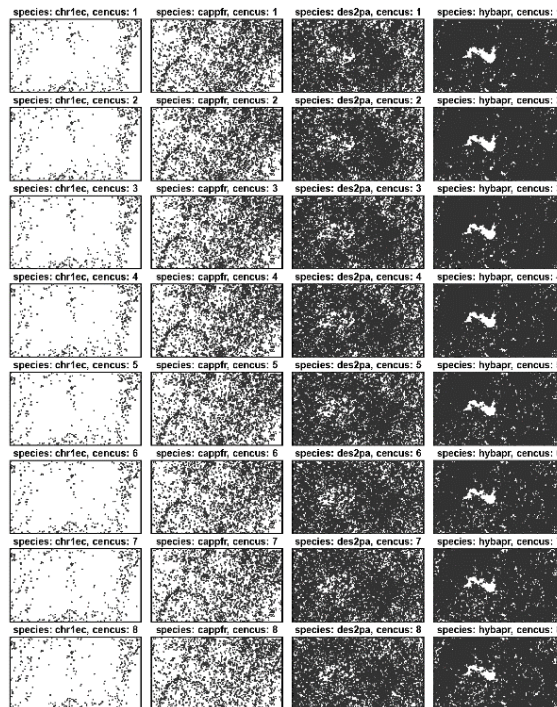
Ketika nilai $o(u, v) = 1$, maka kedua titik dikatakan independen atau stasioner. *Point process* dikatakan stasioner apabila (1) jumlah titik pada setiap daerah cenderung sama sehingga $\lambda(u) = \lambda$, (2) jumlah titik pada suatu daerah tidak mempengaruhi jumlah titik pada daerah lainnya. Jika sebuah eksperimen diulang-ulang pada kondisi serupa, $N(\mathbf{X} \cap W)$ tetap memiliki distribusi serupa walaupun realisasi dari *point process* stasioner yang dihasilkan akan berbeda. Pada kasus stasioner, $N(\mathbf{X} \cap W)$ berdistribusi Poisson dengan intensitas $\sum_{t=1}^{|T|} \int_W \lambda(u) du dt = \lambda|W|$ yang bersifat konstan pada seluruh daerah. Gambar 2.1 menunjukkan 10 realisasi *point process* stasioner pada nilai intensitas yang serupa.



Gambar 2.1 Contoh 10 Realisasi *Stationary Point Process* dengan Nilai Intensitas yang Serupa

Ketika nilai $o(u, v) > 1$, maka kedua titik saling mendekat (*attraction*), sedangkan apabila $o(u, v) < 1$, maka kedua titik saling menjauh (*repulsion*). Kedua peristiwa tersebut menunjukkan adanya ketergantungan antar titik yang dipengaruhi oleh variabel lainnya.

Pada kasus sebenarnya, seringkali dijumpai lebih dari satu *event* yang terjadi pada bidang berdimensi d yang sama dan interval waktu yang berbeda. Kejadian tersebut disebut sebagai *multivariate spatio-temporal point pattern* $\tilde{\mathbf{x}} = \{\mathbf{x}^{(1,1)}, \mathbf{x}^{(2,1)}, \dots, \mathbf{x}^{(s,t)}, \dots, \mathbf{x}^{(m,T)}\}$ dengan $m > 1$ dan $T > 1$ yang sangat umum ditemui pada bidang ekologi, epidemiologi, dan ilmu lingkungan karena melibatkan banyak variabel (Baddeley, 2015). Salah satu contoh *multivariate spatio-temporal point pattern* adalah sekumpulan *point pattern* yang mencerminkan lokasi setiap pohon pada setiap spesies dan waktu observasi di salah satu hutan. Pada kasus ini, setiap spesies pohon dan waktu observasi memiliki *point pattern* masing-masing pada area hutan yang sama. Gambar 2.2 menunjukkan contoh *multivariate spatio-temporal point pattern* yang merepresentasikan lokasi pohon pada setiap spesies dan waktu observasi.



Gambar 2.2 Contoh *Multivariate Spatio-Temporal Point Pattern* yang Merepresentasikan Lokasi Pohon pada 4 Spesies yang Diobservasi pada 8 Waktu di Suatu Hutan (Condit dkk, 2019)

Multivariate spatio-temporal point process dinotasikan sebagai $\check{X} = \{X^{(1,1)}, X^{(2,1)}, \dots, X^{(s,t)}, \dots, X^{(m,T)}\}$. Jika diketahui $N(X^{(s,t)} \cap W)$ adalah jumlah titik di dalam W pada $X^{(s,t)}$ dengan $s = 1, 2, \dots, m$ dan $t = 1, 2, \dots, T$, maka karakteristik *first order* dari $X^{(s,t)}$ pada kasus stasioner atau homogen adalah:

$$\mathbb{E}[N(X^{(s,t)} \cap W)] = \lambda^{(s,t)} |W| \quad (2.5)$$

dimana $\lambda^{(s,t)}$ adalah nilai intensitas yang menunjukkan rata-rata jumlah titik di suatu wilayah pada *point pattern* $x^{(s,t)}$. *Multivariate spatio-temporal point process* \check{X} dikatakan stasioner apabila setiap elemen $X^{(s,t)}$ juga bersifat stasioner. Nilai intensitas \check{X} pada kasus stasioner didefinisikan sebagai:

$$\check{\lambda} = \sum_{t=1}^T \sum_{s=1}^m \lambda^{(s,t)} \quad (2.6)$$

Jika nilai intensitas berubah-ubah pada setiap wilayah (inhomogen), maka karakteristik *first order* dari $X^{(s,t)}$ adalah:

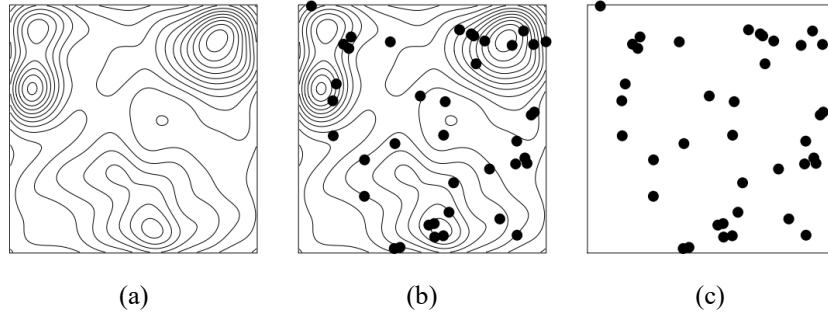
$$\mathbb{E}[N(X^{(s,t)} \cap W)] = \sum_{t=1}^T \int_W \lambda^{(s,t)}(u) du dt \quad (2.7)$$

Nilai intensitas \check{X} pada kasus inhomogen didefinisikan sebagai:

$$\check{\lambda}(u) = \sum_{t=1}^T \sum_{s=1}^m \lambda^{(s,t)}(u) \quad (2.8)$$

2.2 Multivariate Spatio-Temporal log-Gaussian Cox Process

Ketika melakukan analisis terhadap *point pattern* yang memiliki pola interaksi *clustered*, model *homogenous* Poisson Process tidak bisa digunakan karena model ini menggunakan asumsi *point pattern* harus bersifat independen atau stasioner. Oleh sebab itu, dikembangkan metode Cox process yang pada dasarnya merupakan Poisson process dengan *random intensity function* $\Lambda(u)$ (Cox, 1955). Cox process mengasumsikan bahwa pola interaksi *clustered* disebabkan oleh adanya pengaruh variable *covariate* yang teramati dan faktor eksternal yang tidak dapat dijelaskan (*unobservable external effect*) yang bersifat acak. Gambar 2.3 menunjukkan ilustrasi Cox process.



Gambar 2.3 Ilustrasi Cox process. (a) Realisasi *Random Intensity Function* $\Lambda(u)$. (b) Realisasi *Poisson point process* berdasarkan $\Lambda(u)$. (c) Realisasi titik berdasarkan $\Lambda(u)$.

(Baddeley dkk, 2015)

Menurut Møller & Waagepetersen (2007), Cox process dapat didefinisikan menggunakan persamaan:

$$\begin{aligned} \log \Lambda(u) &= \mathbf{G}(u) \\ \Lambda(u) &= \exp(\mathbf{G}(u)) \end{aligned} \quad (2.9)$$

dimana $\mathbf{G}(u)$ diasumsikan sebagai komponen acak yang berdistribusi normal (Gaussian) di setiap titik u (Baddeley, 2015). Pada pasangan titik u dan v , maka $(\mathbf{G}(u), \mathbf{G}(v))$ memiliki distribusi *bivariate* normal. Karakteristik tersebut berlaku untuk setiap kumpulan titik $u_1, \dots, u_{N(X)}$ sehingga $\mathbf{G}(u_1), \dots, \mathbf{G}(u_{N(X)})$ berdistribusi *multivariate* normal. Oleh sebab itu $\mathbf{G}(u)$ disebut sebagai *Gaussian random field*. Pada persamaan (2.9), $\mathbf{G}(u)$ memiliki dua komponen utama, yaitu komponen teramati yang diwakili oleh variabel *covariate* dan komponen yang tidak dapat dijelaskan yang bersifat acak sehingga dapat dinyatakan pula sebagai:

$$\begin{aligned} \Lambda(u) &= \exp\{\mathbf{G}(u)\} \\ &= \exp\{\mu(u) + \Psi(u)\} \\ &= \exp\{\beta_0 + \mathbf{z}(u)^T \boldsymbol{\beta} + \Psi(u)\} \end{aligned} \quad (2.10)$$

dimana $\mu(u)$ adalah fungsi deterministik *mean* dari $\mathbf{G}(u)$ yang mengandung tren spasial, $\mathbf{z}(u) = (z^{(1)}(u), z^{(2)}(u), \dots, z^{(p)}(u))^T$ adalah vektor yang berisi variabel *covariate* pada lokasi u , β_0 adalah nilai *intercept* regresi, $\boldsymbol{\beta} = (\beta^{(1)}, \beta^{(2)}, \dots, \beta^{(p)})^T$ adalah vektor yang berisi koefisien regresi untuk setiap variabel *covariate*, dan $\Psi(u)$ merupakan *Gaussian random field* yang bersifat stasioner dengan *mean* 0 dan *covariance*:

$$\text{Cov}(\Psi(u), \Psi(v)) = \sigma^2 \exp\left(-\frac{\|u - v\|}{\psi}\right) \quad (2.11)$$

dimana $\sigma^2 > 0$ adalah variansi dan $\psi > 0$ adalah parameter skala korelasi, $\|u - v\| = \sqrt{(u^{(1)} - v^{(1)})^2 + (u^{(2)} - v^{(2)})^2}$ merupakan jarak Euclidean antar dua titik pada \mathbb{R}^2 . Berdasarkan persamaan (2.10), maka dapat ditunjukkan bahwa \mathbf{X} merupakan log Gaussian Cox Process (LGCP).

LGCP dapat dikembangkan ke dalam bentuk *multivariate* $\tilde{\mathbf{X}} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(m)}\}$, untuk $m > 1$, dimana setiap komponen $\mathbf{X}^{(s)}$ untuk $s = 1, 2, \dots, m$ merupakan sebuah LGCP yang dipengaruhi oleh *intensity function* $\Lambda^{(s)}(u)$. Persamaan *multivariate* LGCP ditunjukkan pada persamaan:

$$\begin{aligned} \Lambda^{(s)}(u) &= \exp\{\mathbf{G}^{(s)}(u)\} \\ &= \exp\{\mu^{(s)}(u) + \mathbf{F}^{(s)}(u) + \mathbf{K}^{(s)}(u)\} \\ &= \exp\{\mathbf{z}(u)^T \boldsymbol{\beta}^{(s)} + \mathbf{F}^{(s)}(u) + \mathbf{K}^{(s)}(u)\} \end{aligned} \quad (2.12)$$

dimana $\mathbf{F}^{(s)}$ dan $\mathbf{K}^{(s)}$ merupakan Gaussian *random fields* dengan *mean* 0, dan $\mu^{(s)}(u)$ merupakan *mean* pada tiap jenis *multivariate*. Pada contoh kasus persebaran lokasi spesies pohon, $\mathbf{K}^{(s)}$ merepresentasikan *source of clustering* yang disebabkan oleh faktor internal masing-masing spesies dan diasumsikan bersifat independen antar spesies dengan variansi $(\sigma^{(s)})^2 > 0$. Proses dekomposisi dapat diterapkan terhadap variabel $\mathbf{K}^{(s)}$ sehingga:

$$\mathbf{K}^{(s)}(u) = \sigma^{(s)} \mathbf{V}^{(s)}(u) \quad (2.13)$$

Dengan demikian, fungsi *covariance* dari \mathbf{V} adalah:

$$\text{Cov}(\mathbf{V}^{(s)}(u), \mathbf{V}^{(s)}(v)) = \boldsymbol{\Sigma}_V = \exp\left(-\frac{\|u - v\|}{\psi^{(s)}}\right) \quad (2.14)$$

Sedangkan $\mathbf{F}^{(s)}$ merepresentasikan pengaruh yang disebabkan oleh pengaruh luar yang tidak dapat dijelaskan (*unobserved factor*) dan bisa saling dependen. Selain itu, $\mathbf{F}^{(s)}$ juga diasumsikan sebagai:

$$\mathbf{F}^{(s)}(u) = \sum_{l=1}^a \alpha_{sl} \mathbf{E}^{(l)}(u) \quad (2.15)$$

dimana $a > 1$, α merupakan matriks berukuran $m \times a$, dan $\mathbf{E}^{(l)}$, $l = 1, 2, \dots, a$ merupakan Gaussian *random field* dengan *mean* 0 dan bersifat independen (Choiruddin dkk, 2020). \mathbf{E} memiliki fungsi *covariance*:

$$\begin{aligned} \text{Cov}(\mathbf{E}^{(l)}(u), \mathbf{E}^{(l)}(v)) &= \Sigma_{\mathbf{E}} \\ &= \text{diag} \left\{ \exp\left(-\frac{\|u-v\|}{\phi^{(1)}}\right), \dots, \exp\left(-\frac{\|u-v\|}{\phi^{(a)}}\right) \right\} \end{aligned} \quad (2.16)$$

dimana $\text{diag} \left\{ \exp\left(-\frac{\|u-v\|}{\phi^{(1)}}\right), \dots, \exp\left(-\frac{\|u-v\|}{\phi^{(a)}}\right) \right\}$ menunjukkan matriks diagonal yang memiliki elemen diagonal $\exp\left(-\frac{\|u-v\|}{\phi^{(1)}}\right), \dots, \exp\left(-\frac{\|u-v\|}{\phi^{(a)}}\right)$. Sementara itu, $\phi^{(l)} > 0$ merupakan parameter skala korelasi. Dengan demikian, *multivariate LGCP* memiliki parameter $\theta = (\mu, \alpha, \sigma, \psi, \phi)^T$.

Data *multivariate point pattern* dapat memiliki unsur waktu. Salah satu contohnya adalah data lokasi banyak spesies pohon yang diobservasi pada beberapa waktu berbeda. Data tersebut disebut sebagai *multivariate spatio-temporal point pattern*. Menurut Mateu & Jalilian (2022), persamaan *multivariate spatio-temporal LGCP* dituliskan sebagai:

$$\Lambda^{(s,t)}(u) = \exp\{\mu^{(s,t)}(u) + \mathbf{F}^{(s,t)}(u) + \mathbf{K}^{(s,t)}(u)\} \quad (2.17)$$

dimana $t = 1, 2, \dots, T$ dengan $T > 1$ merupakan waktu terjadinya suatu kejadian. Dengan menggunakan dekomposisi untuk $\mathbf{K}^{(s,t)}$:

$$\mathbf{K}^{(s,t)}(u) = \sigma^{(s,t)} \mathbf{V}^{(s,t)}(u) \quad (2.18)$$

dan asumsi untuk $\mathbf{F}^{(s,t)}$:

$$\mathbf{F}^{(s,t)}(u) = \sum_{l=1}^a \alpha_{sl} \mathbf{E}^{(l,t)}(u) \quad (2.19)$$

maka fungsi *covariance* untuk \mathbf{E} dan \mathbf{V} pada kasus *multivariate spatio-temporal* adalah:

$$\begin{aligned}\text{Cov}\left(\mathbf{E}^{(l,t)}(u), \mathbf{E}^{(l,t')}(v)\right) &= \boldsymbol{\Sigma}_E \\ &= \exp\left\{\left(-\frac{\|u-v\|}{\phi^{(l)}}\right)^{2+\varphi^{(l)}}\right\} (\rho^{(l)})^{|t-t'|}\end{aligned}\quad (2.20)$$

$$\begin{aligned}\text{Cov}\left(\mathbf{V}^{(s,t)}(u), \mathbf{V}^{(s,t')}(v)\right) &= \boldsymbol{\Sigma}_V \\ &= \exp\left(-\frac{\|u-v\|}{\psi^{(s)}}\right) (\zeta^{(s)})^{|t-t'|}\end{aligned}\quad (2.21)$$

dimana $|t - t'|$ merupakan nilai absolut dari selisih antara dua waktu yang berbeda, $\varphi^{(l)} \in (0,2]$, $\rho^{(l)}$ dan $\zeta^{(s)}$ adalah parameter korelasi waktu dengan $\rho^{(l)} \in [-1,1]$ dan $\zeta^{(s)} \in [-1,1]$. Oleh sebab itu, *multivariate spatio-temporal LGCP* memiliki parameter $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\varphi}, \boldsymbol{\rho}, \boldsymbol{\zeta})^T$ dengan:

- a) $\boldsymbol{\mu}$ berukuran m .
- b) $\boldsymbol{\alpha}$ berukuran $a \times m$,
- c) $\boldsymbol{\sigma}$ berukuran m ,
- d) $\boldsymbol{\psi}$ berukuran m ,
- e) $\boldsymbol{\phi}$ berukuran a ,
- f) $\boldsymbol{\varphi}$ berukuran a ,
- g) $\boldsymbol{\rho}$ berukuran a ,
- h) $\boldsymbol{\zeta}$ berukuran m .

2.3 Interpretasi Model

Hasil estimasi parameter $\hat{\boldsymbol{\theta}}$ dapat diolah lebih lanjut untuk mengetahui informasi pola persebaran titik pada *multivariate spatio-temporal point pattern*.

2.3.1 Hierarchical Clustering

Clustering atau analisis *cluster* merupakan analisis untuk mengelompokkan data-data dengan karakteristik serupa ke dalam kelompok (*cluster*) yang sama (Lim dkk, 2021). Dengan demikian, masing-masing *cluster* memiliki sifat homogen karena terdiri dari data-data berkarakteristik serupa. Secara umum, metode *clustering* terbagi menjadi 2 bagian, yaitu (1) *hierarchical clustering*, dan (2) *non-hierarchical clustering*. Salah satu kelebihan metode *hierarchical clustering*

dibandingkan metode lainnya adalah metode ini tidak memerlukan penentuan jumlah *cluster* sebelum proses *clustering* dimulai. Hasil pengelompokkan divisualisasikan melalui *dendogram* yang menunjukkan hubungan hierarkis antar data dan dapat dipakai untuk menentukan jumlah *cluster* yang paling sesuai.

Salah satu metode *hierarchical clustering* yang paling populer adalah metode Ward (Ward, 1963). Metode tersebut tergolong sebagai *agglomerative clustering* yang bertujuan untuk meminimalkan *within-cluster sum of squares* pada setiap tahap pengelompokkan. Kelebihan utama metode Ward adalah kemampuannya untuk meminimalkan variansi *within-cluster* secara efektif, sehingga setiap *cluster* terdiri dari data-data yang memiliki tingkat kemiripan tinggi. Berikut merupakan tahap *clustering* menggunakan metode Ward:

1. Pembentukan *cluster* untuk setiap individu data.
2. Menghitung jarak antar *cluster*. Jarak antara 2 *cluster* diukur dengan melihat peningkatan *within-cluster sum of squares* atau WCSS ketika kedua *cluster* digabungkan.

$$WCSS(C) = \sum_{i' \in C} \|x^{(i')} - \bar{x}_C\|^2 \quad (2.22)$$

dimana \bar{x}_C adalah mean dari *cluster* C.

3. Menggabungkan 2 *cluster* yang memiliki peningkatan WCSS terkecil. Untuk setiap pasangan *cluster* C_1 dan C_2 , peningkatan WCSS untuk $C_{gab} = C_1 \cup C_2$ adalah:

$$\Delta WCSS = WCSS(C_{gab}) - (WCSS(C_1) + WCSS(C_2)) \quad (2.23)$$

Metode Ward memilih pasangan *cluster* yang memiliki $\Delta WCSS$ terkecil untuk dikelompokkan ke dalam satu *cluster*.

4. Ulangi langkah penggabungan hingga semua data tergabung dalam sebuah *cluster* besar.

Menurut Waagetersen dkk (2015), jarak antar spesies s dan s' dihitung berdasarkan jarak Euclid $\|\hat{\alpha}_s - \hat{\alpha}_{s'}\|$ dan dapat dipakai untuk mengelompokkan banyak spesies berdasarkan pola pengaruhnya terhadap *unobservable factor* $\mathbf{E}^{(l)}$. Pengelompokkan *hierarchical clustering* dihitung berdasarkan jarak Euclid

$\|\hat{\alpha}_s - \hat{\alpha}_{s'}\|$ menggunakan metode Ward yang dapat divisualisasikan ke dalam plot *dendogram*.

2.3.2 Korelasi Antar Spesies

Hubungan antar spesies juga dapat diketahui berdasarkan nilai korelasi antar spesies pada Gaussian *random field* $\mathbf{G}^{(s)}$ yang memuat kontribusi dari *unobservable factor* $\mathbf{F}^{(s)}$ dan faktor internal setiap spesies $\mathbf{K}^{(s)}$ (lihat persamaan 2.12) (Choiruddin dkk, 2020). Korelasi antar spesies tersebut dihitung menggunakan:

$$\text{Corr}\left(\mathbf{G}^{(s)}(u), \mathbf{G}^{(s')}(u)\right) = \text{Corr}\left(\mathbf{F}^{(s)}(u), \mathbf{F}^{(s')}(u)\right) \sqrt{\text{PV}^{(s)}(0)} \sqrt{\text{PV}^{(s')}(0)} \quad (2.24)$$

dimana $\text{Corr}\left(\mathbf{F}^{(s)}(u), \mathbf{F}^{(s')}(u)\right)$ merepresentasikan nilai korelasi antar spesies yang dipengaruhi oleh *unobservable factor*. $\text{Corr}\left(\mathbf{F}^{(s)}(u), \mathbf{F}^{(s')}(u)\right)$ dihitung berdasarkan:

$$\text{Corr}\left(\mathbf{F}^{(s)}(u), \mathbf{F}^{(s')}(u)\right) = \frac{\hat{\alpha}_s^T \hat{\alpha}_{s'}}{\sqrt{\|\hat{\alpha}_s\|^2 \times \|\hat{\alpha}_{s'}\|^2}} \quad (2.25)$$

sementara $\text{PV}^{(s)}(0)$ merupakan proporsi variansi untuk *lag* 0 pada spesies ke- s yang dihitung menggunakan persamaan:

$$\begin{aligned} \text{PV}^{(s)}(0) &= \frac{\text{Cov}\left(\mathbf{F}^{(s)}(u), \mathbf{F}^{(s)}(u)\right)}{\text{Cov}\left(\mathbf{G}^{(s)}(u), \mathbf{G}^{(s)}(u)\right)} \\ &= \frac{\hat{\alpha}_s^T \hat{\alpha}_s}{\hat{\alpha}_s^T \hat{\alpha}_s + (\hat{\sigma}^{(s)})^2} \end{aligned} \quad (2.26)$$

Estimasi korelasi dapat dikembangkan pada kasus *spatio-temporal*, sehingga:

$$\begin{aligned} \text{Corr}\left(\mathbf{G}^{(s,t)}(u), \mathbf{G}^{(s',t)}(u)\right) &= \text{Corr}\left(\mathbf{F}^{(s,t)}(u), \mathbf{F}^{(s',t)}(u)\right) \\ &\quad \times \sqrt{\text{PV}^{(s,t)}(0)} \sqrt{\text{PV}^{(s',t)}(0)} \end{aligned} \quad (2.27)$$

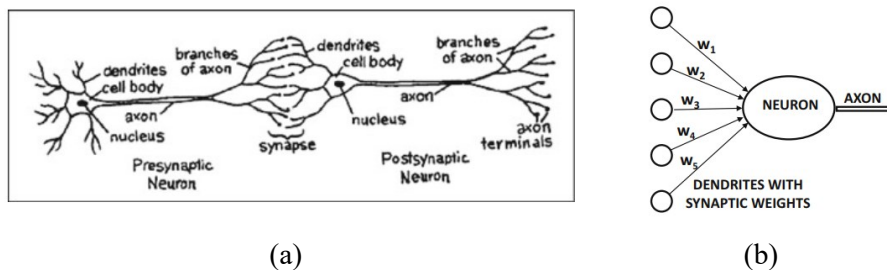
$$\text{PV}^{(s,t)}(0) = \frac{\text{Cov}\left(\mathbf{F}^{(s,t)}(u), \mathbf{F}^{(s,t)}(u)\right)}{\text{Cov}\left(\mathbf{G}^{(s,t)}(u), \mathbf{G}^{(s,t)}(u)\right)} \quad (2.28)$$

$$= \frac{\hat{\mathbf{a}}_s^T \hat{\mathbf{a}}_s}{\hat{\mathbf{a}}_s^T \hat{\mathbf{a}}_s + (\hat{\sigma}^{(s,t)})^2}$$

Nilai korelasi pada persamaan (2.24) dan persamaan (2.27) menghasilkan matriks korelasi yang dapat divisualisasikan menggunakan *heatmap plot*.

2.4 Neural Network

Artificial Neural Network (ANN) atau *Neural Network* (NN) merupakan sebuah metode yang terinspirasi dari mekanisme dan proses pembelajaran yang terjadi pada otak makhluk hidup (Aggarwal, 2018). Otak makhluk hidup terdiri dari banyak *neuron* yang dihubungkan satu sama lain oleh *axon* dan *dendrite*. Daerah penghubung yang terletak diantara *axon* dan *dendrite* disebut sebagai *synapses*. Setiap *neuron* selalu mendapatkan sinyal atau *stimuli* melalui *dendrite*. Sinyal tersebut berasal dari indera makhluk hidup yang berinteraksi dengan dunia luar secara terus menerus. Selanjutnya, *neuron* menjumlahkan sinyal-sinyal yang masuk. Apabila penjumlahan tersebut melebihi sebuah batas tertentu, maka *neuron* akan meneruskan sinyal tersebut ke *neuron* lainnya melalui *axon*. Sinyal yang didapatkan oleh *neuron* sangat mempengaruhi kekuatan hubungan *synaptic* antar *neuron*. Proses tersebut berulang kepada *neuron-neuron* lainnya sehingga terbentuk sebuah mekanisme pembelajaran yang dipengaruhi oleh input dari dunia luar. Gambar 2.4 (a) menunjukkan struktur *biological neuron* pada makhluk hidup, sedangkan Gambar 2.4 (b) menunjukkan struktur NN yang mensimulasikan mekanisme pembelajaran *biological neuron*.



Gambar 2. 4 Perbandingan Struktur antara (a) *Biological Neural Network* dan (b) *Artificial Neural Network* (Aggarwal, 2018)

Mekanisme pembelajaran tersebut kemudian diadaptasi ke dalam metode NN. NN memiliki unit komputasi yang disebut sebagai *neuron* yang saling

terhubung satu sama lain. *Neuron-neuron* yang saling berdekatan membentuk sebuah *layer*. Setiap *neuron* pada suatu *layer* memiliki hubungan koneksi pada setiap *neuron* pada *layer* berikutnya. Kekuatan hubungan antar *neuron* tersebut ditentukan berdasarkan nilai bobot.

Setiap *neuron* mendapatkan data input, kemudian data input tersebut dihitung bersama dengan nilai bobot untuk mendapatkan nilai *output*. Ketika nilai *output* melewati nilai ambang batas, maka *neuron* akan mengirimkan sinyal positif ke *neuron* lainnya seperti yang terjadi pada mekanisme pembelajaran *biological neuron*. Nilai ambang batas tersebut ditentukan oleh sebuah fungsi aktivasi pada setiap *neuron*. Mekanisme pembelajaran pada NN terjadi dengan mengubah nilai bobot selama proses pelatihan dilakukan (Singh dan Banerjee, 2019). Secara matematis, proses pengolahan data input menjadi *output* pada *single layer perceptron* ditunjukkan pada persamaan:

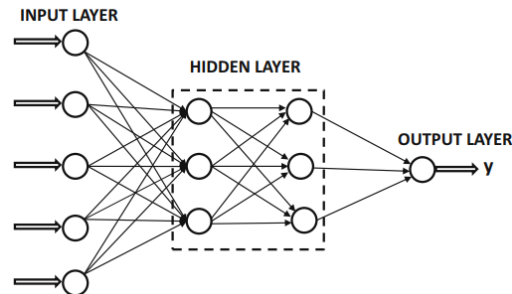
$$\hat{y} = f(b + \mathbf{w}^T \mathbf{X}) = f\left(b + \sum_{k'=1}^L w_{k'} X_{k'}\right) \quad (2.29)$$

dimana f merupakan sebuah fungsi aktivasi, b sebagai *intercept* atau bias, $\mathbf{w} = \{w_1, \dots, w_L\}$ merupakan bobot, \hat{y} merupakan nilai prediksi yang dikeluarkan oleh *neural network*, dan $\mathbf{X} = \{X_1, \dots, X_L\}$ merupakan data input.

Seperti yang ditunjukkan pada Gambar 2.5, sebuah *neural network* terdiri dari 3 jenis *layer* (Uzair dan Jamil, 2020), yaitu:

- a. *Input layer*, merupakan tempat masuknya informasi dari luar. *Layer* ini terdiri dari *neuron-neuron* yang bertugas menampung data sebagai *input*.
- b. *Output layer*, merepresentasikan nilai keluaran yang telah diproses oleh *neural network*.
- c. *Hidden layer*, merupakan *layer* penghubung antara *input layer* dan *output layer*. Jumlah *hidden layer* pada suatu *network* sangat bergantung terhadap kompleksitas permasalahan yang dihadapi. Jumlah *hidden layer* yang terlalu sedikit umumnya akan menyebabkan *underfitting*, yaitu ketidakmampuan *neural network* untuk melakukan prediksi pada data latih dan data validasi. Sementara itu, jumlah *hidden layer* yang terlalu besar akan mengakibatkan terjadinya *overfitting*, dimana *neural network* memiliki performa yang bagus

pada data latih namun mendapatkan hasil prediksi yang buruk pada data validasi.



Gambar 2. 5 Contoh *Neural Network* dengan *Hidden Layer* (Aggarwal, 2018)

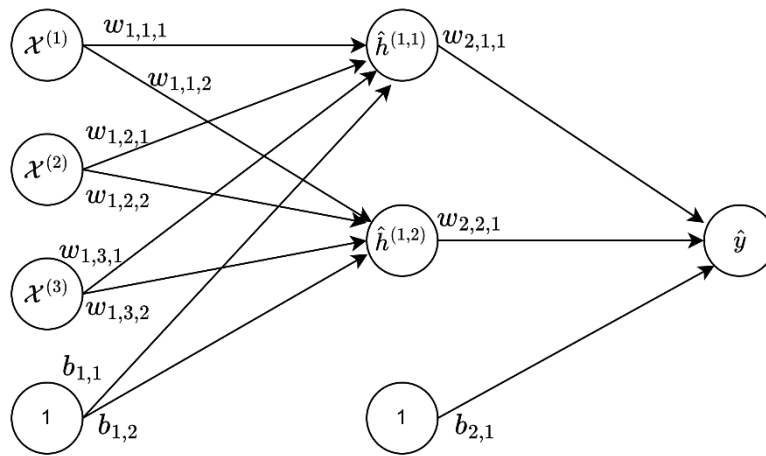
Bentuk *neural network* yang paling sederhana adalah *single layer perceptron*. Pada model ini, *neural network* hanya terdiri dari sebuah *input layer* dan sebuah *output node* (Singh dan Banerjee, 2019). *Single layer perceptron* hanya dapat dipakai untuk melakukan klasifikasi secara linear. Beberapa tahun kemudian, penelitian lain menyebutkan bahwa *neural network* dengan 2 atau lebih *hidden layer* memiliki performa yang lebih baik daripada *single layer perceptron* karena model ini dapat memodelkan data secara nonlinier. Model dengan lebih dari satu *hidden layer* ini disebut sebagai *multilayer perceptron* (Singh dan Banerjee, 2019).

Neural network termasuk ke dalam kelompok pembelajaran terbimbing karena metode ini menggunakan data latih yang sudah diberi label untuk melakukan proses pembelajaran. Ketika pembelajaran berlangsung, nilai *output* yang dihasilkan *neural network* dibandingkan dengan nilai target yang sudah diberi label sebelumnya sehingga didapatkan nilai galat. Apabila galat bernilai kecil, maka *neural network* memiliki performa yang baik.

2.4.1 RMSProp

Root Mean Square Propagation (RMSProp) merupakan salah satu algoritma optimisasi yang pertama kali diperkenalkan oleh Hinton dkk (2017). RMSProp merupakan pengembangan dari algoritma optimisasi Stochastic Gradient Descent (SGD) dan Adagrad yang membutuhkan penghitungan *gradient* untuk melakukan optimisasi. Pada *neural network*, *gradient* dihitung ketika proses pelatihan berlangsung menggunakan algoritma *backpropagation* (Goodfellow dkk, 2016). Nilai *gradient* tersebut dipakai oleh RMSProp untuk mengubah nilai bobot

sehingga nilai galat menjadi semakin kecil. Dengan demikian, sebelum melakukan *backpropagation*, nilai *output* harus dihitung terlebih dahulu melalui *neural network* (*forward propagation*). Proses *forward propagation* dan *backpropagation* dilakukan berulang kali sehingga ketika proses pembelajaran telah selesai, *neural network* memiliki nilai bobot paling optimal karena menghasilkan nilai galat yang sangat kecil. Contoh arsitektur *neural network* yang menerapkan algoritma pembelajaran *backpropagation* ditunjukkan pada Gambar 2.6.



Gambar 2.6 Contoh Arsitektur *Neural Network*

Pada Gambar 2.6, *neural network* terdiri dari sebuah *input layer*, *hidden layer*, dan *output layer*. *Input layer* memiliki 3 *neuron* atau input, yaitu $\mathcal{X}^{(1)}$, $\mathcal{X}^{(2)}$, dan $\mathcal{X}^{(3)}$ sehingga $\mathcal{L}_1 = 3$. *Hidden layer* terdiri dari 2 *neuron*, yaitu $\hat{h}^{(1,1)}$ dan $\hat{h}^{(1,2)}$, sehingga $\mathcal{L}_2 = 2$. Sedangkan *output layer* terdiri dari 1 *neuron* \hat{y} , sehingga $\mathcal{L}_3 = 1$. *Input layer* dan *hidden layer* dihubungkan oleh bobot $w_{1,i,j}$, $i = 1, 2, 3$ dan $j = 1, 2$. Sedangkan *hidden layer* dan *output layer* dihubungkan oleh bobot $w_{2,j,k}$, $k = 1$. Variabel $b_{1,j}$ merupakan *bias* pada *hidden layer*, dan variabel $b_{2,1}$ menunjukkan *bias* pada *output layer*.

Menurut Tian dkk (2023), algoritma pembelajaran RMSProp dilakukan dengan mengikuti langkah-langkah berikut:

1. Inisialisasi nilai bobot secara acak mengikuti distribusi normal

2. Ketika suatu kondisi berhenti, misalkan jumlah pengulangan *epoch*, tidak terpenuhi maka dilakukan langkah selanjutnya.

3. Fase *forward propagation*:

Umumnya *neural network* dilatih menggunakan ukuran *batch* $b' > 1$ agar pelatihannya lebih efisien (Goodfellow dkk, 2016), sehingga tahapan *forward propagation* adalah:

- a) Setiap *neuron* pada *input layer* $\mathcal{X}_{i'}^{(i)}$; $i = 1,2,3$; $i' = 1,2, \dots, b'$, menerima data input dan meneruskannya ke semua *neuron* pada *hidden layer*.
- b) Setiap *neuron* pada *hidden layer* $\hat{h}_{in_{i'}}^{(1,j)}$, $j = 1,2$ menjumlahkan sinyal-sinyal dari *input layer* dengan persamaan:

$$\hat{h}_{in_{i'}}^{(1,j)} = b_{1,j} + \sum_{i=1}^{\mathcal{L}_1} w_{1,i,j} \mathcal{X}_i^{(i)} \quad (2.30)$$

dimana $\mathcal{L}_1 = 3$ merupakan jumlah *neuron* pada *input layer*, \mathbf{b}_1 merupakan vektor bias, dan \mathbf{w}_1 merupakan matriks bobot. Selanjutnya, diterapkan fungsi aktivasi f_1 terhadap nilai $\hat{h}_{in_{i'}}^{(1,j)}$:

$$\hat{h}_{i'}^{(1,j)} = f_1 \left(\hat{h}_{in_{i'}}^{(1,j)} \right) = f_1 \left(b_{1,j} + \sum_{i=1}^{\mathcal{L}_1} w_{1,i,j} \mathcal{X}_i^{(i)} \right) \quad (2.31)$$

dimana f_1 merupakan fungsi aktivasi untuk menghitung sinyal *output*. Seluruh *output* dari *hidden layer* diteruskan ke *output layer*.

- c) Setiap *neuron* pada *output layer* $\hat{y}^{(k)}$, $k = 1$ menerima sinyal dari *hidden layer* dan dilakukan penghitungan menggunakan persamaan:

$$\hat{y}_{in_{i'}}^{(k)} = b_{2,k} + \sum_{j=1}^{\mathcal{L}_2} w_{2,j,k} \hat{h}_{i'}^{(1,j)} \quad (2.32)$$

dimana $\mathcal{L}_2 = 2$ merupakan jumlah *neuron* pada *layer* kedua, \mathbf{b}_2 merupakan vektor bias, dan \mathbf{w}_2 merupakan matriks bobot. Selanjutnya, diterapkan fungsi aktivasi f_2 terhadap nilai $\hat{y}_{in_{i'}}^{(k)}$.

$$\hat{y}_{i'}^{(k)} = f_2 \left(\hat{y}_{in_{i'}}^{(k)} \right) = f_2 \left(b_{2,k} + \sum_{j=1}^{\mathcal{L}_2} w_{2,j,k} \hat{h}_{i'}^{(1,j)} \right) \quad (2.33)$$

4. Penghitungan *loss*:

Pada contoh kasus ini, *neural network* mengeluarkan estimasi nilai kontinyu sehingga tergolong sebagai permasalahan regresi. Oleh sebab itu, fungsi *loss* yang dipakai adalah Mean Squared Error (MSE). Fungsi *loss* didefinisikan sebagai berikut:

$$\text{MSE} = L = \frac{1}{b' \times \mathcal{L}_3} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} \left(y_{i'}^{(k)} - \hat{y}_{i'}^{(k)} \right)^2 \quad (2.34)$$

dimana $\mathcal{L}_3 = 2$ merupakan jumlah *neuron* pada *output layer*.

5. Fase *backpropagation*:

Backpropagation dimulai dari *output layer* hingga *input layer* dengan menerapkan operasi *chain rule* untuk menghitung *gradient* fungsi *loss* terhadap setiap bobot *neural network*.

- a) Setiap *neuron* pada *output layer* $\hat{y}_{i'}^{(k)}$ memiliki nilai target $y_{i'}^{(k)}$ dan selanjutnya dihitung informasi *gradient* dengan persamaan:

$$\begin{aligned} \frac{\partial L}{\partial w_{2,j,k}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \frac{\partial L}{\partial \hat{y}_{i'}^{(k)}} \cdot \frac{\partial \hat{y}_{i'}^{(k)}}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot \frac{\partial \hat{y}_{in_{i'}}^{(k)}}{\partial w_{2,j,k}} \\ \frac{\partial L}{\partial w_{2,j,k}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \left(\hat{y}_{i'}^{(k)} - y_{i'}^{(k)} \right) \cdot \frac{\partial f_2 \left(\hat{y}_{in_{i'}}^{(k)} \right)}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot \hat{h}_{i'}^{(1,j)} \end{aligned} \quad (2.35)$$

Penghitungan serupa juga dilakukan untuk menghitung *gradient* pada bias.

$$\frac{\partial L}{\partial b_{2,k}} = \frac{1}{b'} \sum_{i'=1}^{b'} \left(\hat{y}_{i'}^{(k)} - y_{i'}^{(k)} \right) \cdot \frac{\partial f_2 \left(\hat{y}_{in_{i'}}^{(k)} \right)}{\partial \hat{y}_{in_{i'}}^{(k)}} \quad (2.36)$$

- b) Penghitungan *gradient* dilakukan terhadap seluruh bias dan bobot pada *hidden layer* dengan persamaan:

$$\begin{aligned}\frac{\partial L}{\partial w_{1,i,j}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} \frac{\partial L}{\partial \hat{y}_{i'}^{(k)}} \cdot \frac{\partial \hat{y}_{i'}^{(k)}}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot \frac{\partial \hat{y}_{in_{i'}}^{(k)}}{\partial \hat{h}_{i'}^{(1,j)}} \cdot \frac{\partial \hat{h}_{i'}^{(1,j)}}{\partial \hat{h}_{in_{i'}}^{(1,j)}} \cdot \frac{\partial \hat{h}_{in_{i'}}^{(1,j)}}{\partial w_{1,i,j}} \\ \frac{\partial L}{\partial w_{1,i,j}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} (\hat{y}_{i'}^{(k)} - y_{i'}^{(k)}) \cdot \frac{\partial f_2(\hat{y}_{in_{i'}}^{(k)})}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot w_{2,j,k} \cdot \frac{\partial f_1(\hat{h}_{in_{i'}}^{(1,j)})}{\partial \hat{h}_{in_{i'}}^{(1,j)}} \cdot x_{i'}^{(i)}\end{aligned}\quad (2.37)$$

Penghitungan serupa juga dilakukan untuk menghitung *gradient* pada *bias*.

$$\frac{\partial L}{\partial b_{1,j}} = \frac{1}{b'} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} (\hat{y}_{i'}^{(k)} - y_{i'}^{(k)}) \cdot \frac{\partial f_2(\hat{y}_{in_{i'}}^{(k)})}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot w_{2,j,k} \cdot \frac{\partial f_1(\hat{h}_{in_{i'}}^{(1,j)})}{\partial \hat{h}_{in_{i'}}^{(1,j)}} \quad (2.38)$$

6. Fase perubahan bobot:

- a) Dilakukan perubahan bobot pada setiap *neuron* di *output layer* $\hat{y}^{(k)}$.

$$v_{w_{2,j,k}} = \eta v_{w_{2,j,k}} + (1 - \eta) \left(\frac{\partial L}{\partial w_{2,j,k}} \right)^2 \quad (2.39)$$

$$w_{2,j,k} = w_{2,j,k} - \frac{\gamma}{\sqrt{v_{w_{2,j,k}} + \epsilon}} \frac{\partial L}{\partial w_{2,j,k}} \quad (2.40)$$

dimana γ merupakan *learning rate* yang telah ditentukan sebelumnya, η merupakan *decay rate* yang umumnya bernilai 0.9, dan ϵ merupakan suatu konstan bernilai sangat kecil untuk mencegah pembagian 0. Perubahan serupa juga dilakukan pada setiap *bias* di *output layer*.

$$v_{b_{2,k}} = \eta v_{b_{2,k}} + (1 - \eta) \left(\frac{\partial L}{\partial b_{2,k}} \right)^2 \quad (2.41)$$

$$b_{2,k} = b_{2,k} - \frac{\gamma}{\sqrt{v_{b_{2,k}} + \epsilon}} \frac{\partial L}{\partial b_{2,k}} \quad (2.42)$$

b) Dilakukan perubahan bobot pada setiap *neuron* di *hidden layer* $\hat{h}^{(1,j)}$.

$$v_{w_{1,i,j}} = \eta v_{w_{1,i,j}} + (1 - \eta) \left(\frac{\partial L}{\partial w_{1,i,j}} \right)^2 \quad (2.43)$$

$$w_{1,i,j} = w_{1,i,j} - \frac{\gamma}{\sqrt{v_{w_{1,i,j}} + \epsilon}} \frac{\partial L}{\partial w_{1,i,j}} \quad (2.44)$$

Perubahan serupa juga dilakukan pada setiap *bias* di *output layer*.

$$v_{b_{1,j}} = \eta v_{b_{1,j}} + (1 - \eta) \left(\frac{\partial L}{\partial b_{1,j}} \right)^2 \quad (2.45)$$

$$b_{1,j} = b_{1,j} - \frac{\gamma}{\sqrt{v_{b_{1,j}} + \epsilon}} \frac{\partial L}{\partial b_{1,j}} \quad (2.46)$$

7. Melakukan pengujian apakah suatu kondisi berhenti telah terpenuhi. Apabila belum tercapai, maka dilakukan pengulangan kembali dari langkah 2 sampai 5. Apabila telah tercapai, maka pelatihan dihentikan.

2.4.2 Fungsi Aktivasi

Selain nilai bobot pada *neuron*, fungsi aktivasi berperan penting dalam menentukan karakteristik *output* dari *neural network* (Lederer, 2021). Fungsi aktivasi memiliki banyak jenis dan umumnya bersifat nonlinier sehingga memungkinkan *neural network* untuk memodelkan data secara nonlinier. Pemilihan jenis fungsi aktivasi dilakukan sebelum pembelajaran dimulai dan satu *neural network* bisa memiliki banyak jenis fungsi aktivasi sekaligus. Selain itu, fungsi aktivasi beserta turunan pertamanya sangat mempengaruhi kompleksitas model *neural network* karena keduanya digunakan untuk penghitungan optimasi bobot pada *backpropagation*. Penjelasan terkait beberapa jenis fungsi aktivasi yang digunakan di dalam penelitian ini dibahas pada subbab-subbab berikut.

1) Fungsi Sigmoid

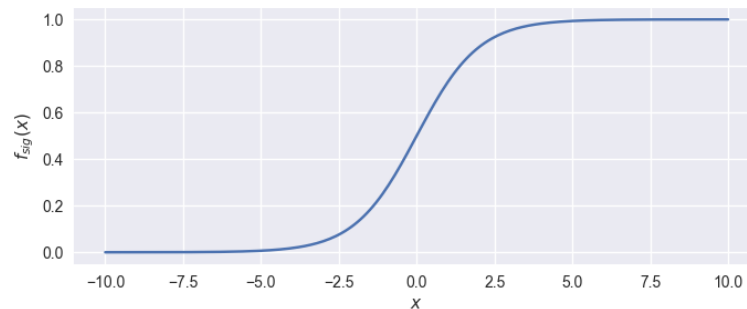
Fungsi *sigmoid* adalah sebuah fungsi *non-decreasing* dan memiliki kurva yang membentuk seperti huruf “S”. Fungsi *sigmoid* dapat dilihat pula sebagai

bentuk *smooth* dari *binary activation* yang digunakan pada *single layer perceptron*. Fungsi *sigmoid* dan turunan pertama didefinisikan sebagai:

$$f_{\text{sig}}(\mathcal{X}) = \frac{1}{1 + e^{-\mathcal{X}}}, \text{ dimana } f_{\text{sig}}(\mathcal{X}) \in (0,1) \quad (2.47)$$

$$\frac{\partial f_{\text{sig}}(\mathcal{X})}{\partial \mathcal{X}} = f_{\text{sig}}'(\mathcal{X}) = f_{\text{sig}}(\mathcal{X}) (1 - f_{\text{sig}}(\mathcal{X}))$$

dimana \mathcal{X} merupakan suatu nilai yang akan diterapkan fungsi aktivasi. Nilai *output* dari fungsi *sigmoid* ditunjukkan pada Gambar 2.7 berikut.



Gambar 2.7 Fungsi Aktivasi *Sigmoid*

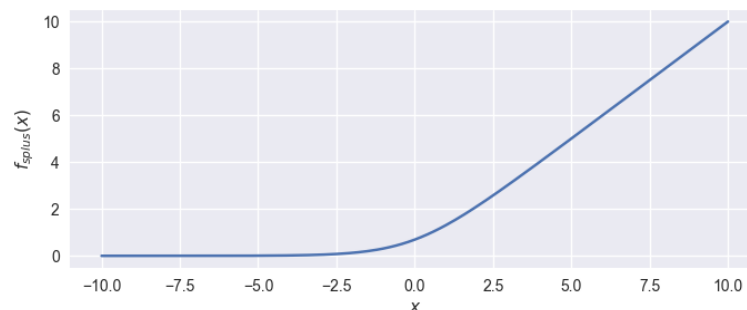
2) Fungsi Softplus

Fungsi *softplus* pertama kali diperkenalkan oleh Dugas dkk (2001). Fungsi *softplus* termasuk fungsi nonlinier yang didefinisikan sebagai:

$$f_{\text{splus}}(\mathcal{X}) = \log[1 + e^{\mathcal{X}}], \text{ dimana } f_{\text{splus}}(\mathcal{X}) \in (0, \infty) \quad (2.48)$$

$$\frac{\partial f_{\text{splus}}(\mathcal{X})}{\partial \mathcal{X}} = f_{\text{splus}}'(\mathcal{X}) = f_{\text{sig}}(\mathcal{X})$$

Nilai *output* dari fungsi *softplus* selalu bernilai positif. Grafik yang menunjukkan fungsi *softplus* ditunjukkan pada Gambar 2.8.



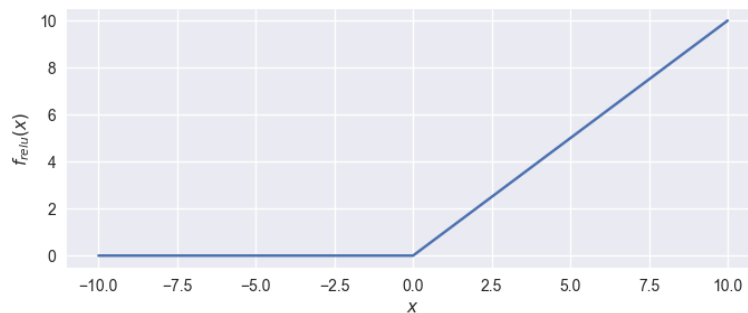
Gambar 2.8 Fungsi Aktivasi *Softplus*

3) Fungsi ReLU

Fungsi ReLU (Rectified Linear Unit) merupakan fungsi non-linear yang didefinisikan menggunakan persamaan:

$$f_{\text{relu}}(\mathcal{X}) = \max(0, \mathcal{X}), \text{ dimana } f_{\text{relu}}(\mathcal{X}) \in (0, \infty)$$
$$\frac{\partial f_{\text{relu}}(\mathcal{X})}{\partial \mathcal{X}} = f_{\text{relu}}'(\mathcal{X}) = \begin{cases} 1, & \text{jika } \mathcal{X} > 0 \\ 0, & \text{jika } \mathcal{X} \leq 0 \end{cases} \quad (2.49)$$

Fungsi ReLU selalu mengeluarkan nilai positif, sehingga fungsi aktivasi ini memiliki rentang *output* yang sama seperti fungsi aktivasi *softplus*. Walaupun demikian, fungsi ReLU lebih sering dipakai daripada *softplus* karena persamaan ReLU lebih sederhana sehingga *neural network* dengan fungsi aktivasi ReLU cenderung memiliki waktu pelatihan yang lebih cepat daripada *softplus*. Grafik yang menunjukkan fungsi ReLU ditunjukkan pada Gambar 2.9.



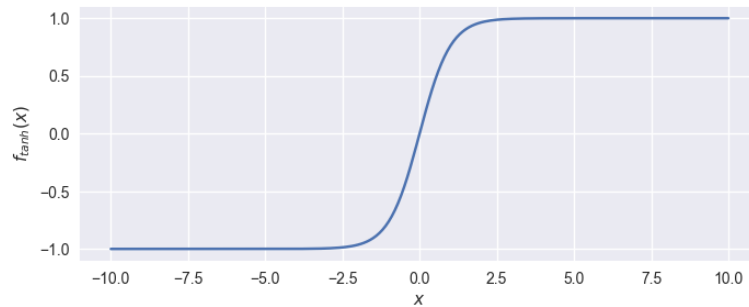
Gambar 2.9 Fungsi Aktivasi ReLU

4) Fungsi Tanh

Fungsi *hyperbolic tangent* atau fungsi *tanh* memiliki karakteristik yang serupa seperti fungsi *sigmoid*. Perbedaannya hanya terdapat pada domain yang digunakan. Persamaan fungsi *tanh* adalah:

$$f_{\text{tanh}}(\mathcal{X}) = \frac{e^{\mathcal{X}} - e^{-\mathcal{X}}}{e^{\mathcal{X}} + e^{-\mathcal{X}}}, \text{ dimana } f_{\text{tanh}}(\mathcal{X}) \in (-1, 1)$$
$$\frac{\partial f_{\text{tanh}}(\mathcal{X})}{\partial \mathcal{X}} = f_{\text{tanh}}'(\mathcal{X}) = 1 - (f_{\text{tanh}}(\mathcal{X}))^2 \quad (2.50)$$

Fungsi ini menghasilkan nilai yang berada pada rentang -1 dan 1 . Semakin kecil suatu nilai input, maka nilai yang dihasilkan akan mendekati nilai -1 , begitu pula sebaliknya. Gambar 2.10 menunjukkan penghitungan fungsi *tanh*.



Gambar 2.10 Fungsi Aktivasi *Tanh*

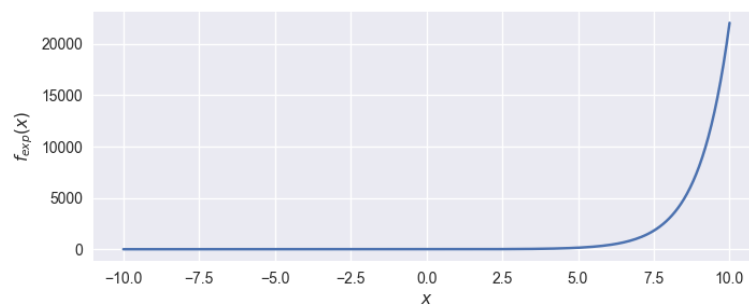
5) Fungsi Eksponensial

Sesuai dengan namanya, fungsi aktivasi eksponensial menggunakan fungsi eksponensial untuk memetakan nilai input. Fungsi ini termasuk ke dalam fungsi nonlinier dan memiliki domain yang serupa dengan fungsi *softplus*. Fungsi eksponensial didefinisikan sebagai:

$$f_{\text{exp}}(\mathcal{X}) = \exp(\mathcal{X}), \text{ dimana } f_{\text{exp}}(\mathcal{X}) \in (0, \infty)$$

$$\frac{\partial f_{\text{exp}}(\mathcal{X})}{\partial \mathcal{X}} = f_{\text{exp}}'(\mathcal{X}) = \exp(\mathcal{X}) \quad (2.51)$$

Grafik fungsi eksponensial ditunjukkan pada Gambar 2.11.

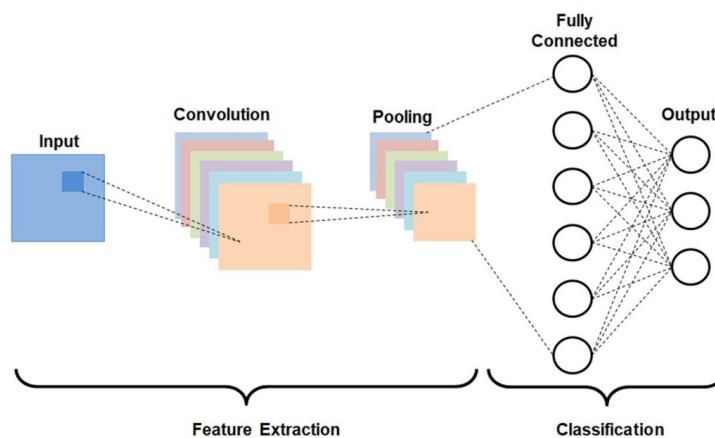


Gambar 2.11 Fungsi Eksponensial

2.5 Deep Learning

Deep learning merupakan salah satu bagian dari *machine learning* yang akhir-akhir ini menjadi metode yang paling banyak digunakan untuk menyelesaikan

permasalahan kompleks (Alzubaidi dkk, 2021). *Deep learning* merupakan neural network yang terdiri dari banyak *layer*. Pada umumnya, *neural network* konvensional hanya terdiri dari beberapa *hidden layer* beserta sebuah *input layer* dan *output layer*. Sedangkan *deep learning* dapat memiliki hingga ribuan *layer* sekaligus yang saling terhubung satu sama lain. Tidak jarang pula ditemui arsitektur *deep learning* yang memiliki beberapa *input layer* sekaligus. Dengan kata lain, *deep learning* memiliki arsitektur yang lebih kompleks dan lebih fleksibel daripada *neural network* konvensional. Berdasarkan karakteristik tersebut, *deep learning* sering pula disebut sebagai *deep neural network* atau *large neural network*. Gambar 2.12 menunjukkan contoh arsitektur *deep learning* menggunakan *convolution layer* untuk klasifikasi gambar.



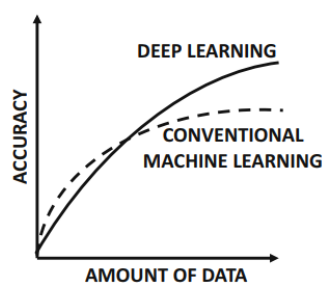
Gambar 2.12 Contoh Arsitektur *Deep Learning* (Tama dkk, 2023)

Salah satu kelebihan utama *deep learning* dibandingkan dengan metode *machine learning* yang lain adalah kemampuan *deep learning* untuk melakukan ekstraksi fitur secara otomatis tanpa campur tangan manusia, atau yang biasa disebut sebagai ekstraksi fitur secara *end-to-end*. Dengan demikian, peneliti tidak perlu melakukan pengolahan fitur terlebih dahulu sehingga meminimalkan terjadinya bias dan kesalahan manusia. Pada Gambar 2.12, ekstraksi fitur otomatis dilakukan oleh serangkaian *convolution layer* dan *pooling layer* yang disusun secara bertingkat. Setiap *layer* untuk ekstraksi fitur akan mencoba mengambil fitur yang merupakan representasi dari data, lalu menyalurkan hasil fitur tersebut ke *layer* selanjutnya. Oleh sebab itu, *layer* pada bagian awal arsitektur akan

mengambil fitur *low-level* sedangkan *layer* pada bagian akhir arsitektur akan mengambil fitur *high-level*. Kemudian, fitur yang telah diekstraksi disalurkan ke *fully connected layer* untuk dilakukan proses klasifikasi. Proses tersebut mirip dengan proses pembelajaran yang terjadi pada otak makhluk hidup.

Beberapa penelitian telah membuktikan bahwa *deep learning* mampu mendapatkan hasil yang sangat baik pada kasus-kasus yang kompleks seperti *text mining* (Amrit dkk, 2017), pendeteksian spam (Crawford dkk, 2015), dan klasifikasi gambar (Deldjoo dkk, 2016). Pada sebuah penelitian oleh Nagpal, dkk (2019), para ahli patologi diminta untuk menganalisis tingkat keparahan kanker prostat. Hasilnya, para ahli patologi mendapatkan tingkat akurasi sebesar 61%, sementara model *deep learning* yang dikembangkan berhasil mendapatkan akurasi sebesar 70%, sehingga penelitian ini menyimpulkan bahwa *deep learning* mampu untuk mengalahkan hasil analisis para ahli patologi.

Perkembangan pesat dan kepopuleran *deep learning* tidak terlepas dari tersedianya sumber daya komputasi dan banyaknya jumlah data yang tersedia pada akhir-akhir ini. Semakin banyak dan rumit *layer* yang digunakan, maka akan semakin banyak pula komputasi yang harus dilakukan untuk menarik pola dari sekumpulan data. Selain itu, *deep learning* juga membutuhkan data dengan jumlah yang besar untuk melakukan pelatihan.



Gambar 2.13 Perbandingan Performa *Deep Learning* dan Metode *Machine Learning* Lain Berdasarkan Jumlah Data (Aggarwal, 2018)

Gambar 2.13 menunjukkan perbandingan performa antara *deep learning* dengan metode *machine learning* lain berdasarkan jumlah data yang tersedia. Berdasarkan Gambar 2.13, terlihat bahwa performa *deep learning* akan semakin

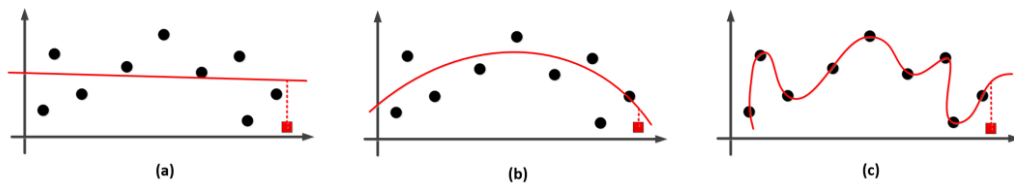
baik ketika menggunakan data yang berukuran semakin besar, berbeda dengan metode lainnya yang cenderung stagnan ketika telah mencapai jumlah data tertentu. Berdasarkan kelebihan tersebut, maka *deep learning* cocok untuk menganalisis dataset yang besar dan kompleks dengan memanfaatkan teknologi komputasi yang tersedia saat ini.

Penelitian ini menggunakan *package library* Keras yang berbasis pada bahasa pemrograman Python dan mampu berjalan di atas *library* Tensorflow, CNTK, dan Theano (Chollet, 2015). Keras mendukung banyak jenis *layer* yang umum digunakan di dalam model *deep learning*. Beberapa jenis *layer* yang digunakan pada penelitian ini adalah:

1. *Input layer*, bertujuan untuk menerima nilai masukan. Sebuah model *deep learning* dapat memiliki lebih dari satu *input layer*.
2. *Dense layer*, merupakan nama lain dari *hidden layer* pada terminologi *neural network*. Setiap *dense layer* memiliki sebuah fungsi aktivasi.
3. *Dropout layer*, merupakan *layer* yang menerapkan metode *dropout* untuk mencegah *overfitting*.
4. *Concatenate layer*, merupakan *layer* yang bertujuan untuk menggabungkan (*concatenate*) input dari beberapa *layer* yang berbeda menjadi sebuah *layer*.
5. *Custom layer*, merupakan *layer* yang proses penghitungannya didefinisikan secara manual oleh sebuah fungsi.
6. *Add layer*, merupakan *layer* yang bertujuan untuk melakukan operasi penjumlahan pada input data.
7. *Multiply layer*, merupakan *layer* yang bertujuan untuk melakukan operasi perkalian pada input data.

2.5.1 Dropout pada Deep Learning

Deep learning merupakan model kompleks yang terdiri dari banyak *layer* sehingga *deep learning* dapat memodelkan data secara *highly non-linear*. Karena kompleksitasnya tersebut, model *deep learning* seringkali mengalami *overfitting* apabila tidak dikembangkan dengan benar. Gambar 2.14 menunjukkan permasalahan yang sering muncul pada model *deep learning*.



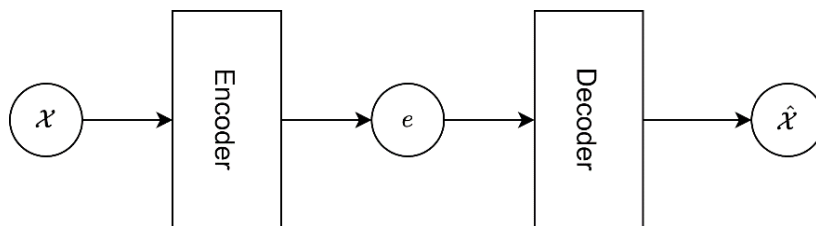
Gambar 2.14 Perbandingan Hasil Pemodelan antara Model *Underfit* (a), *Good Fit* (b), dan Model *Overfit* (c) (Ghojogh & Crowley, 2019)

Berdasarkan Gambar 2.14 (a), *underfitting* merupakan kebalikan dari *overfitting* yang terjadi ketika model gagal untuk mengikuti pola data latih maupun data validasi. *Overfitting* yang ditunjukkan pada gambar 2.14 (c) terjadi apabila model memiliki performa yang sangat baik pada data latih, namun mendapatkan performa yang buruk pada data validasi. *Overfitting* terjadi karena model mempelajari pula *random noise* dan fluktuasi pada data latih yang menyebabkan model tersebut gagal untuk melakukan prediksi pada data baru. Dengan kata lain, model tidak memiliki kemampuan generalisasi yang baik. Model yang baik adalah model yang mampu mengikuti pola data latih dengan baik serta memiliki kemampuan prediksi yang baik pula pada data baru seperti yang ditunjukkan pada Gambar 2.14 (b).

Salah satu cara untuk menanggulangi terjadinya *overfitting* adalah metode *dropout*. *Dropout* telah terbukti mampu untuk meningkatkan kemampuan generalisasi model *deep learning* (Srivastava dkk, 2014). Pada metode *dropout*, *neuron* akan dinonaktifkan sementara ketika proses pelatihan berlangsung. Penonaktifan *neuron* dilakukan secara acak dan masing-masing *neuron* memiliki peluang P yang sama untuk dinonaktifkan. Ketika proses pelatihan berlangsung, *neuron-neuron* yang telah dinonaktifkan tersebut tidak akan diikuti dalam penghitungan *backpropagation* dan *forward propagation*. Setelah dilakukan pelatihan model, maka akan didapatkan sekumpulan parameter bobot \mathbf{w} . Ketika model *deep learning* digunakan untuk melakukan prediksi, penonaktifan *neuron* secara acak tidak lagi dilakukan. Semua *neuron* akan dipakai seluruhnya. Namun, nilai bobot \mathbf{w} akan diberi pembobot P yang telah ditentukan di awal pelatihan, sehingga nilai bobot menjadi $P \times \mathbf{w}$.

2.5.2 Variational Autoencoder

Variational autoencoder (VAE) merupakan salah satu jenis arsitektur *deep learning* yang terdiri dari 2 bagian utama, yaitu *encoder* dan *decoder* (Kingma dan Welling, 2014). VAE merupakan pengembangan dari *autoencoder* (AE) konvensional. Pada AE, tujuan utama model *autoencoder* adalah melakukan rekonstruksi berdasarkan data input sehingga hasil rekonstruksi tersebut sangat mirip dengan data input aslinya menggunakan *mean squared error* (MSE) sebagai *loss function* (Zhang dkk, 2018). Sementara itu, selain mempelajari representasi data yang efisien seperti pada *autoencoder*, VAE juga bertujuan untuk mempelajari karakteristik distribusi dari data sehingga VAE dapat dipakai untuk membangkitkan data baru berdasarkan karakteristik distribusi tersebut. Secara matematis, VAE bertujuan untuk mengestimasi suatu *posterior distribution* $p_{post}(\mathbf{e}|\boldsymbol{\tau})$ berdasarkan data input menggunakan suatu distribusi perkiraan $q(\mathbf{e}|\mathcal{X}, \hat{\boldsymbol{\tau}})$, dimana parameter $\boldsymbol{\tau}$ merupakan parameter suatu distribusi yang dikeluarkan oleh *neural network*.



Gambar 2.15 Arsitektur *Variational Autoencoder*

Visualisasi arsitektur VAE ditunjukkan pada Gambar 2.15 yang terdiri dari *encoder* dan *decoder*. *Encoder* dan *decoder* terdiri dari arsitektur *neural network* yang bisa dilatih secara terpisah. *Encoder* bertujuan untuk mengubah input data \mathcal{X} ke dalam distribusi perkiraan $q(\mathbf{e}|\mathcal{X}, \hat{\boldsymbol{\tau}})$. Kemudian estimasi \mathbf{e} didapatkan dengan melakukan *sampling* terhadap $q(\mathbf{e}|\mathcal{X}, \hat{\boldsymbol{\tau}})$. Sementara itu, *decoder* bertujuan untuk merekonstruksi *latent space* \mathbf{e} kembali ke bentuk semula yang didefinisikan sebagai $\hat{\mathcal{X}}$.

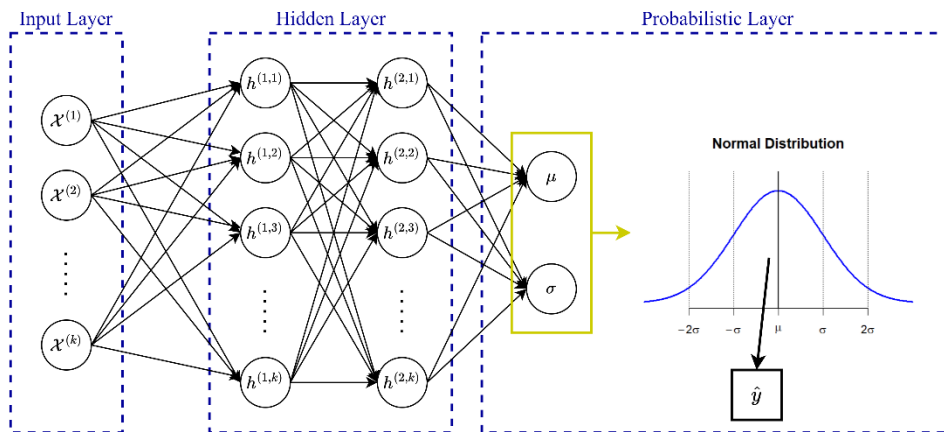
VAE sering dipakai untuk membangkitkan data baru dengan cara melakukan *sampling* pada $q(\mathbf{e}|\mathcal{X}, \hat{\boldsymbol{\tau}})$. Selain itu, VAE juga sering digunakan untuk pendeteksian data anomali dengan melatih VAE pada data kondisi normal. VAE akan mendapatkan hasil rekonstruksi yang buruk ketika diterapkan pada data yang

memiliki pola anomali. Hasil rekonstruksi yang buruk menandakan bahwa data tersebut berpotensi mengandung data anomali.

2.6 Probabilistic Deep Learning

Probabilistic deep learning merupakan model *deep learning* yang menggunakan *probabilistic layer* sehingga model ini dapat memperhitungkan adanya ketidakpastian (*uncertainty*) pada model (Chang, 2021). *Probabilistic deep learning* merupakan perpaduan antara *probabilistic model*, yaitu model yang menggunakan suatu distribusi peluang untuk mengukur ketidakpastian, dan *deep learning*. Berbeda dengan *deep learning* konvensional yang mengeluarkan sebuah nilai *output* sebagai estimasi titik, *probabilistic deep learning* mengeluarkan *output* berupa *probability density function* (pdf) dengan nilai parameter tertentu.

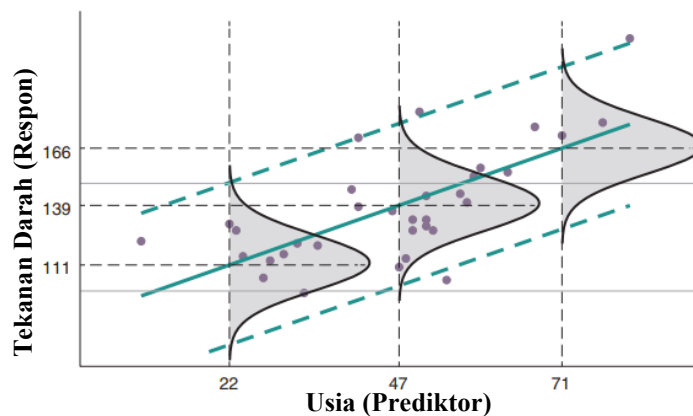
Pada pemodelan *probabilistic deep learning*, parameter yang diestimasi tergolong sebagai *random variable* yang mengikuti suatu distribusi peluang tertentu. Tujuan dari pemodelan *probabilistic deep learning* adalah memprediksi suatu *posterior distribution* $p_{post}(\mathbf{e}|\mathcal{X}, \boldsymbol{\tau})$ menggunakan suatu distribusi perkiraan $q(\mathbf{e}|\mathcal{X}, \hat{\boldsymbol{\tau}})$, dimana \mathbf{e} adalah *latent random vector*, \mathcal{X} merupakan data input, $\hat{\boldsymbol{\tau}}$ dan $\boldsymbol{\tau}$ adalah nilai parameter yang menentukan suatu distribusi peluang (Singh & Ogunfunmi, 2022). Optimasi dilakukan terhadap parameter $\hat{\boldsymbol{\tau}}$ sehingga nilai parameter tersebut mendekati nilai parameter sebenarnya $\boldsymbol{\tau}$. Apabila nilai kedua parameter tersebut mirip, maka distribusi $q(\mathbf{e}|\mathcal{X}, \hat{\boldsymbol{\tau}})$ berhasil memperkirakan $p_{post}(\mathbf{e}|\mathcal{X}, \boldsymbol{\tau})$ dengan baik. Konsep pelatihan seperti ini disebut sebagai *variational inference* (Blei dkk, 2017; Doersch, 2016). Pada *probabilistic deep learning*, nilai $\hat{\boldsymbol{\tau}}$ dioptimasi dan diprediksi menggunakan *probabilistic layer* sehingga *deep learning* memberikan estimasi bobot \mathbf{w} dan parameter distribusi $\hat{\boldsymbol{\tau}}$. *Probabilistic deep learning* memiliki *input layer* dan *hidden layer* sama seperti arsitektur *deep learning* pada umumnya. Perbedaannya, *probabilistic layer* yang merepresentasikan suatu distribusi tertentu diletakkan di bagian tengah atau akhir dari arsitektur tersebut. Gambar 2.16 menunjukkan contoh arsitektur *probabilistic deep learning* yang mengeluarkan *output* berupa distribusi normal dengan *mean* μ dan standar deviasi σ .



Gambar 2.16 Contoh Arsitektur *Probabilistic Deep Learning* (Chang, 2021)

Pada Gambar 2.16, $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(k)}$ merupakan data input, $h^{(1,k)}$ merupakan *neuron* ke-k pada *hidden layer* kesatu, dan $h^{(2,k)}$ merupakan *neuron* ke-k pada *hidden layer* kedua. *Probabilistic deep learning* mengeluarkan estimasi parameter $\hat{\tau} = (\mu, \sigma)^T$. Dengan demikian, *hidden layer* pada arsitektur tersebut dapat pula disebut sebagai *encoder network* seperti pada konsep arsitektur *autoencoder* yang direpresentasikan sebagai $q(\mathbf{e}|\mathcal{X}, \hat{\tau})$. Kemudian, nilai prediksi dari *probabilistic deep learning* \hat{y} didapatkan dengan melakukan *sampling* berdasarkan distribusi yang terbentuk. Nilai \hat{y} tersebut dapat bertindak sebagai nilai *output* akhir maupun dapat digunakan sebagai input kembali untuk *layer-layer* berikutnya. Oleh sebab itu, proses pengolahan data input \mathcal{X} hingga didapatkan \hat{y} adalah:

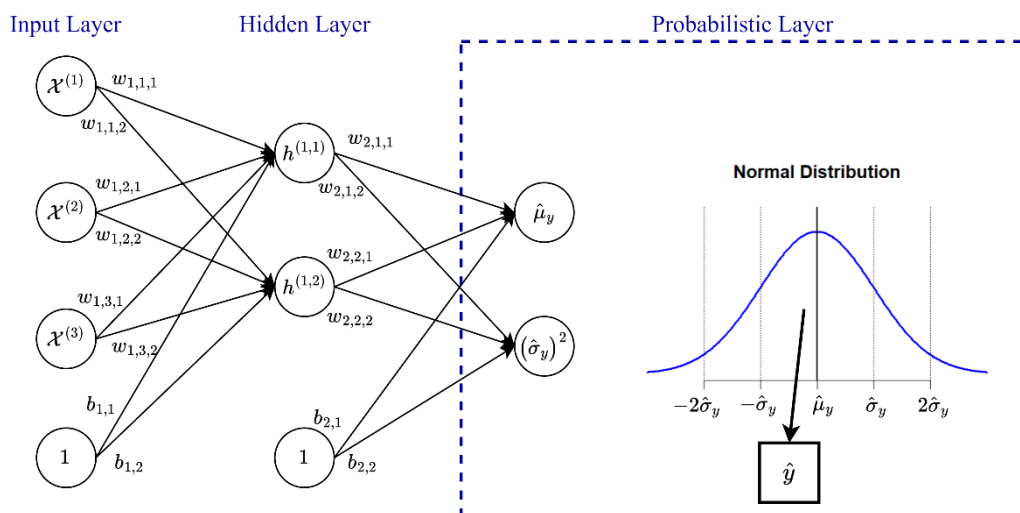
$$\begin{aligned} \hat{\tau} &= \text{EncoderNetwork}(\mathcal{X}|\mathbf{w}) \\ \hat{y} &\sim q(\hat{y}|\hat{\tau}) \end{aligned} \quad (2.52)$$



Gambar 2.17 Contoh Penerapan *Probabilistic Deep Learning* (Dürr dkk, 2020)

Gambar 2.17 menunjukkan plot hasil pemodelan *probabilistic deep learning* pada suatu contoh kasus. Pada Gambar 2.17, sumbu x merupakan data input (variabel prediktor) sedangkan sumbu y menunjukkan data hasil prediksi (variabel respon). Seperti yang ditunjukkan pada Gambar 2.17, model *probabilistic deep learning* mengeluarkan output berupa distribusi normal yang menunjukkan persebaran dari nilai prediksi. Hasil distribusi normal tersebut dapat digunakan untuk membentuk rentang kepercayaan dari nilai prediksi sehingga selain mendapatkan nilai estimasi titik, peneliti juga bisa mendapatkan estimasi rentang kepercayaannya.

Probabilistic deep learning memiliki tahapan-tahapan pembelajaran yang serupa dengan *deep learning* konvensional. Proses pelatihan dimulai dengan menghitung nilai output melalui tahapan *forward propagation*. Selanjutnya, nilai galat dihitung sebagai selisih antara nilai sebenarnya dan nilai prediksi. Algoritma *backpropagation* kemudian dimulai dari *output layer* hingga *input layer* sehingga didapatkan nilai *gradient* pada setiap *layer*. Nilai *gradient* tersebut dipakai oleh algoritma optimasi (misalnya: RMSProp, SGD, Adam) untuk melakukan perubahan bobot sehingga nilai galat menjadi semakin kecil. Proses ini dilakukan berulang-ulang dalam banyak iterasi hingga mencapai suatu kondisi tertentu. Tahapan pelatihan *probabilistic deep learning* dapat diterapkan terhadap *network* pada Gambar 2.18.



Gambar 2.18 Penerapan *Backpropagation* pada *Probabilistic Deep Learning*

Arsitektur *probabilistic deep learning* pada Gambar 2.18 memiliki struktur yang hampir sama dengan arsitektur pada Gambar 2.6. Model terdiri dari sebuah *input layer* dengan 3 *neuron*, sebuah *hidden layer* dengan 2 *neuron*, dan *output layer* dengan 2 *neuron* yang mencerminkan hasil estimasi parameter lokasi dan skala dari suatu distribusi normal. Menurut Dürr dkk (2020), tahapan-tahapan pembelajaran pada *probabilistic deep learning* adalah:

- 1) Inisialisasi nilai bobot secara acak,
- 2) Pengecekan kondisi berhenti. Apabila kondisi tidak terpenuhi maka dilakukan langkah berikutnya.
- 3) Tahapan *forward-propagation*:

Pada umumnya, pelatihan *neural network* menggunakan ukuran *batch* $b' > 1$ agar pelatihan model menjadi lebih efisien (Goodfellow dkk, 2016), sehingga tahapan *forward propagation* dengan menggunakan *batch* adalah:

- a) Setiap *neuron* pada *input layer* $\mathcal{X}_{i'}^{(i)}$; $i = 1, 2, 3$; $i' = 1, 2, \dots, b'$ menerima data input dan meneruskannya ke semua *neuron* pada *hidden layer*.
- b) Setiap *neuron* pada *hidden layer* $\hat{h}_{i'}^{(1,j)}$, $j = 1, 2$ menjumlahkan sinyal-sinyal dari *input layer* dan diterapkan fungsi aktivasi f_1 menggunakan persamaan:

$$\hat{h}_{in_{i'}}^{(1,j)} = b_{1,j} + \sum_{i=1}^{\mathcal{L}_1} w_{1,i,j} \mathcal{X}_i^{(i)} \quad (2.53)$$

$$\hat{h}_{i'}^{(1,j)} = f_1 \left(\hat{h}_{in_{i'}}^{(1,j)} \right) = f_1 \left(b_{1,j} + \sum_{i=1}^{\mathcal{L}_1} w_{1,i,j} \mathcal{X}_i^{(i)} \right) \quad (2.54)$$

dimana $\mathcal{L}_1 = 3$ merupakan jumlah *neuron* pada *input layer*, \mathbf{b}_1 merupakan vektor bias, dan \mathbf{w}_1 merupakan matriks bobot.

- c) Setiap *neuron* pada *output layer* menerima sinyal dari *hidden layer* dan dilakukan penghitungan menggunakan persamaan:

$$\hat{\mu}_{y_{in_{i'}}} = \hat{y}_{in_{i'}}^{(1)} = b_{2,1} + \sum_{j=1}^{\mathcal{L}_2} w_{2,j,1} \hat{h}_{i'}^{(1,j)} \quad (2.55)$$

$$(\hat{\sigma}_{y_{in}})_{i'}^2 = \hat{y}_{in_{i'}}^{(2)} = b_{2,2} + \sum_{j=1}^{\mathcal{L}_2} w_{2,j,2} \hat{h}_{i'}^{(1,j)}$$

dimana $\mathcal{L}_2 = 2$ merupakan jumlah *neuron* pada *layer* kedua, \mathbf{b}_2 merupakan vektor bias, dan \mathbf{w}_2 merupakan matriks bobot. Selanjutnya diterapkan fungsi aktivasi f_2 :

$$\begin{aligned} \hat{\mu}_{y_{i'}} &= \hat{y}_{i'}^{(1)} = f_2(\hat{\mu}_{y_{in_{i'}}}) = f_2\left(b_{2,1} + \sum_{j=1}^{\mathcal{L}_2} w_{2,j,1} \hat{h}_{i'}^{(1,j)}\right) \\ (\hat{\sigma}_y)_{i'}^2 &= \hat{y}_{i'}^{(2)} = f_2\left((\hat{\sigma}_{y_{in}})_{i'}^2\right) = f_2\left(b_{2,2} + \sum_{j=1}^{\mathcal{L}_2} w_{2,j,2} \hat{h}_{i'}^{(1,j)}\right) \end{aligned} \quad (2.56)$$

- d) Melakukan *sampling* untuk mendapatkan estimasi $\hat{y}_{i'}$ yang diasumsikan mengikuti distribusi Normal $(\hat{\mu}_{y_{i'}}, (\hat{\sigma}_y)_{i'}^2)$. Namun, proses *sampling* secara langsung $\hat{y}_{i'} \sim \text{Normal}(\hat{\mu}_{y_{i'}}, (\hat{\sigma}_y)_{i'}^2)$ tidak dapat dilakukan karena *neural network* menggunakan algoritma *backpropagation* yang membutuhkan operasi deterministik (non-stokastik). Oleh sebab itu, $\hat{y}_{i'}$ didapatkan dengan menggunakan *reparameterization trick* (Kingma dan Weling, 2014):

$$\hat{y}_{i'} = \hat{\mu}_{y_{i'}} + \hat{\sigma}_{y_{i'}} \odot \varepsilon_{i'} \quad (2.57)$$

Pada persamaan (2.57), $\varepsilon_{i'} \sim \text{Normal}(0,1)$ yang memperkenalkan elemen stokastik atau keacakan terhadap $\hat{y}_{i'}$, sedangkan \odot merupakan tanda perkalian elemen demi elemen (*element-wise multiplication*).

4. Penghitungan *loss function*:

Probabilistic deep learning dapat dikategorikan sebagai *autoencoder*. Dengan demikian, *loss* yang dipakai adalah MSE. Fungsi *loss* didefinisikan sebagai:

$$\text{MSE} = L = \frac{1}{b' \times \mathcal{L}_3} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} (y_{i'}^{(k)} - \hat{y}_{i'}^{(k)})^2 \quad (2.58)$$

dimana $\mathcal{L}_3 = 2$ merupakan jumlah neuron pada *layer* ketiga (*output layer*).

5. Tahapan *backpropagation*:

Tahapan *backpropagation* dimulai dari *output layer* hingga *input layer* dengan menerapkan operasi *chain rule* untuk menghitung *gradient* fungsi *loss* terhadap setiap bobot *neural network*.

a) *Gradient* bobot pada *output layer* adalah:

$$\begin{aligned} \frac{\partial L}{\partial w_{2,j,k}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \frac{\partial L}{\partial \hat{y}_{i'}^{(k)}} \cdot \frac{\partial \hat{y}_{i'}^{(k)}}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot \frac{\partial \hat{y}_{in_{i'}}^{(k)}}{\partial w_{2,j,k}} \\ \frac{\partial L}{\partial w_{2,j,k}} &= \frac{1}{b'} \sum_{i'=1}^{b'} (\hat{y}_{i'}^{(k)} - y_{i'}^{(k)}) \cdot \frac{\partial f_2(\hat{y}_{in_{i'}}^{(k)})}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot \hat{h}_{i'}^{(1,j)} \end{aligned} \quad (2.59)$$

Sementara itu, penghitungan *gradient* untuk bias pada *output layer* $b_{2,k}$ adalah:

$$\frac{\partial L}{\partial b_{2,k}} = \frac{1}{b'} \sum_{i'=1}^{b'} (\hat{y}_{i'}^{(k)} - y_{i'}^{(k)}) \cdot \frac{\partial f_2(\hat{y}_{in_{i'}}^{(k)})}{\partial \hat{y}_{in_{i'}}^{(k)}} \quad (2.60)$$

b) *Gradient* bobot pada *hidden layer* adalah:

$$\begin{aligned} \frac{\partial L}{\partial w_{1,i,j}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} \frac{\partial L}{\partial \hat{y}_{i'}^{(k)}} \cdot \frac{\partial \hat{y}_{i'}^{(k)}}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot \frac{\partial \hat{y}_{in_{i'}}^{(k)}}{\partial \hat{h}_{i'}^{(1,j)}} \cdot \frac{\partial \hat{h}_{i'}^{(1,j)}}{\partial \hat{h}_{in_{i'}}^{(1,j)}} \cdot \frac{\partial \hat{h}_{in_{i'}}^{(1,j)}}{\partial w_{1,i,j}} \\ \frac{\partial L}{\partial w_{1,i,j}} &= \frac{1}{b'} \sum_{i'=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} (\hat{y}_{i'}^{(k)} - y_{i'}^{(k)}) \cdot \frac{\partial f_2(\hat{y}_{in_{i'}}^{(k)})}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot w_{2,j,k} \cdot \frac{\partial f_1(\hat{h}_{in_{i'}}^{(1,j)})}{\partial \hat{h}_{in_{i'}}^{(1,j)}} \cdot x_{i'}^{(i)} \end{aligned} \quad (2.61)$$

Dengan demikian, *gradient* pada bias $b_{1,j}$ adalah:

$$\frac{\partial L}{\partial b_{1,j}} = \frac{1}{b'} \sum_{i=1}^{b'} \sum_{k=1}^{\mathcal{L}_3} (\hat{y}_{i'}^{(k)} - y_{i'}^{(k)}) \cdot \frac{\partial f_2(\hat{y}_{in_{i'}}^{(k)})}{\partial \hat{y}_{in_{i'}}^{(k)}} \cdot w_{2,j,k} \cdot \frac{\partial f_1(\hat{h}_{in_{i'}}^{(1,j)})}{\partial \hat{h}_{in_{i'}}^{(1,j)}} \quad (2.62)$$

6. Tahapan perubahan bobot dan bias menggunakan RMSProp (Hinton dkk, 2017).

a) Perubahan bobot dan bias di *output layer*.

$$\begin{aligned}
 v_{w_{2,j,k}} &= \eta v_{w_{2,j,k}} + (1 - \eta) \left(\frac{\partial L}{\partial w_{2,j,k}} \right)^2 \\
 w_{2,j,k} &= w_{2,j,k} - \frac{\gamma}{\sqrt{v_{w_{2,j,k}} + \epsilon}} \frac{\partial L}{\partial w_{2,j,k}}
 \end{aligned} \tag{2.63}$$

dimana γ merupakan *learning rate* yang telah ditentukan sebelumnya, η merupakan *decay rate* yang umumnya bernilai 0.9, ϵ merupakan suatu konstanta berukuran sangat kecil untuk mencegah pembagian 0, dan $v_{w_{2,j,k}}$ merupakan *moving average gradient* yang nilai awalnya umumnya adalah 0. Perubahan juga dilakukan terhadap bias $b_{2,k}$.

$$\begin{aligned}
 v_{b_{2,k}} &= \eta v_{b_{2,k}} + (1 - \eta) \left(\frac{\partial L}{\partial b_{2,k}} \right)^2 \\
 b_{2,k} &= b_{2,k} - \frac{\gamma}{\sqrt{v_{b_{2,k}} + \epsilon}} \frac{\partial L}{\partial b_{2,k}}
 \end{aligned} \tag{2.64}$$

b) Perubahan bobot dan bias di *hidden layer*.

$$\begin{aligned}
 v_{w_{1,i,j}} &= \eta v_{w_{1,i,j}} + (1 - \eta) \left(\frac{\partial L}{\partial w_{1,i,j}} \right)^2 \\
 w_{1,i,j} &= w_{1,i,j} - \frac{\gamma}{\sqrt{v_{w_{1,i,j}} + \epsilon}} \frac{\partial L}{\partial w_{1,i,j}}
 \end{aligned} \tag{2.65}$$

Perubahan juga dilakukan terhadap setiap bias $b_{1,j}$.

$$\begin{aligned}
 v_{b_{1,j}} &= \eta v_{b_{1,j}} + (1 - \eta) \left(\frac{\partial L}{\partial b_{1,j}} \right)^2 \\
 b_{1,j} &= b_{1,j} - \frac{\gamma}{\sqrt{v_{b_{1,j}} + \epsilon}} \frac{\partial L}{\partial b_{1,j}}
 \end{aligned} \tag{2.66}$$

7. Melakukan pengujian apakah suatu kondisi berhenti telah terpenuhi. Apabila belum tercapai maka dilakukan pengulangan kembali dari langkah 3 hingga 6. Apabila telah tercapai maka pelatihan berhenti.

Pada penelitian ini, *probabilistic layer* digunakan untuk melakukan estimasi parameter pada model *multivariate LGCP* yang memiliki asumsi mengikuti suatu distribusi tertentu. Salah satu contohnya adalah parameter $\mu(u)$ yang mengikuti asumsi berdistribusi Poisson dan parameter $\sigma(u)$ yang mengikuti asumsi berdistribusi Normal. Penelitian ini menggunakan *package* TensorFlow Probability (TFP) yang berjalan pada bahasa pemrograman Python untuk membangun model *probabilistic deep learning*. TFP merupakan ekstensi dari *package* TensorFlow yang bertujuan untuk memudahkan pembuatan *probabilistic modeling* dan *probabilistic machine learning*. TFP menyediakan implementasi *probabilistic layer* dengan berbagai macam jenis distribusi peluang, seperti Normal, Poisson, dan Binomial. *Package* tersebut juga mendukung distribusi campuran (*mixture distribution*) berdasarkan beberapa distribusi standar sehingga semakin memudahkan pengembang dalam mengolah berbagai jenis distribusi. TFP juga menyediakan kompatibilitas untuk *package* Keras sehingga dapat memudahkan peneliti untuk mengkombinasikan *deep learning* dan *probabilistic modelling*.

2.7 Mean Squared Error

Penelitian ini menggunakan Mean Squared Error (MSE) sebagai kriteria kebaikan model. Mean Squared Error (MSE) merupakan sebuah pengukuran kebaikan model untuk model yang mengeluarkan prediksi berupa data kontinyu (Hodson dkk, 2021). MSE merupakan rata-rata selisih kuadrat antara nilai prediksi dan nilai sebenarnya. Selisih tersebut disebut sebagai galat yang menunjukkan seberapa besar perbedaan hasil prediksi dengan data sebenarnya. Oleh sebab itu, nilai MSE yang kecil menunjukkan bahwa model dapat melakukan prediksi dengan baik. Namun, salah satu kekurangan MSE adalah metode ini cenderung memberikan nilai deviasi yang besar karena adanya operasi pengkuadratan. Pada penelitian ini, MSE dihitung dengan menggunakan persamaan:

$$\text{MSE} = \frac{1}{(d_1 \times d_2 \times m)} \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \sum_{s=1}^m (\Lambda_{ij}^{(s)} - \widehat{\Lambda}_{ij}^{(s)})^2 \quad (2.67)$$

dimana d_1 dan d_2 merupakan ukuran diskretisasi yang dibahas pada subbab 3.1.1, $\Lambda_{ij}^{(s)}$ adalah jumlah titik pada suatu *grid* B_{ij} dengan spesies s , sedangkan $\widehat{\Lambda}_{ij}^{(s)}$ adalah nilai estimasinya.

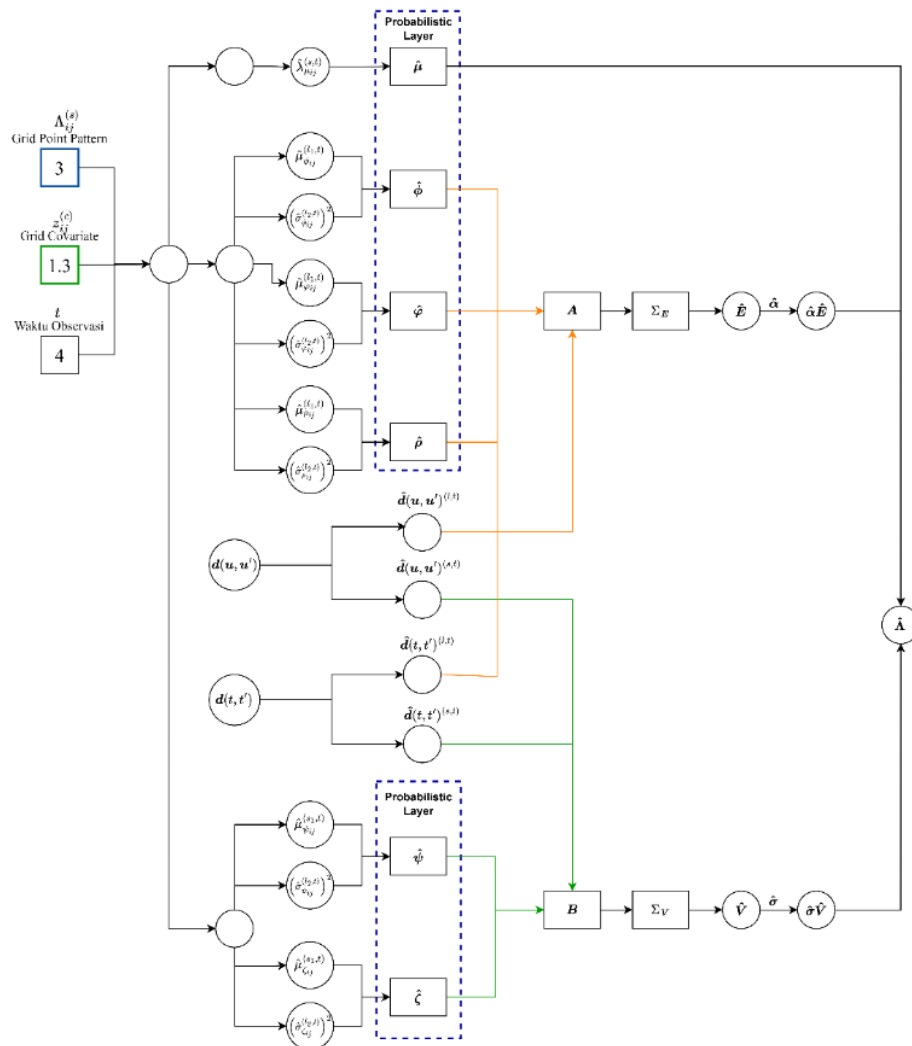
(Halaman ini sengaja dikosongkan)

BAB 3

METODOLOGI PENELITIAN

3.1 Desain Pengembangan Model

Subbab ini membahas tentang metodologi desain pengembangan model *probabilistic deep learning* untuk estimasi parameter *multivariate* LGCP dan *multivariate spatio-temporal* LGCP. Luaran yang dihasilkan adalah arsitektur model *probabilistic deep learning* beserta konfigurasi *hyperparameter*. Secara umum, arsitektur *probabilistic deep learning* untuk estimasi parameter *multivariate spatio-temporal* LGCP ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Ilustrasi *Probabilistic Deep Learning* untuk *Spatio-Temporal* LGCP

Model *probabilistic deep learning* pada Gambar 3.1 menerima *point pattern* dan variabel *covariate* yang telah didiskretisasi. *Point pattern* dan variabel *covariate* tergolong variabel kontinyu, sehingga diperlukan proses diskretisasi untuk mengubah kedua variabel tersebut menjadi diskrit (dibahas lebih dalam pada subbab 3.1.1). Selain itu, model *probabilistic deep learning* untuk *spatio-temporal LGCP* juga menerima jarak antar titik $\mathbf{d}(u, u')$ dan jarak antar waktu $\mathbf{d}(t, t')$ sebagai input. Selanjutnya, model memberikan estimasi $\hat{\boldsymbol{\theta}}$ pada *hidden layer*. Estimasi $\hat{\Lambda}$ didapatkan melalui rekonstruksi berdasarkan persamaan (3.8).

3.1.1 Diskretisasi

Diskretisasi merupakan langkah pertama setelah data *point pattern* didapatkan. Diskretisasi dilakukan dengan cara membagi *observation window* W menjadi sel-sel berukuran sangat kecil yang didefinisikan sebagai B_{ij} dengan $i = 1, 2, \dots, d_1$ dan $j = 1, 2, \dots, d_2$. Dengan demikian, jumlah *grid* untuk setiap *point pattern* adalah $d_1 \times d_2$. Selanjutnya, dihitung jumlah titik pada setiap *grid* B_{ij} yang dinotasikan sebagai matriks $\tilde{\mathbf{x}} = [x_{ij}]$, dimana:

$$x_{ij} = N(\mathbf{X} \cap B_{ij}) \quad (3.1)$$

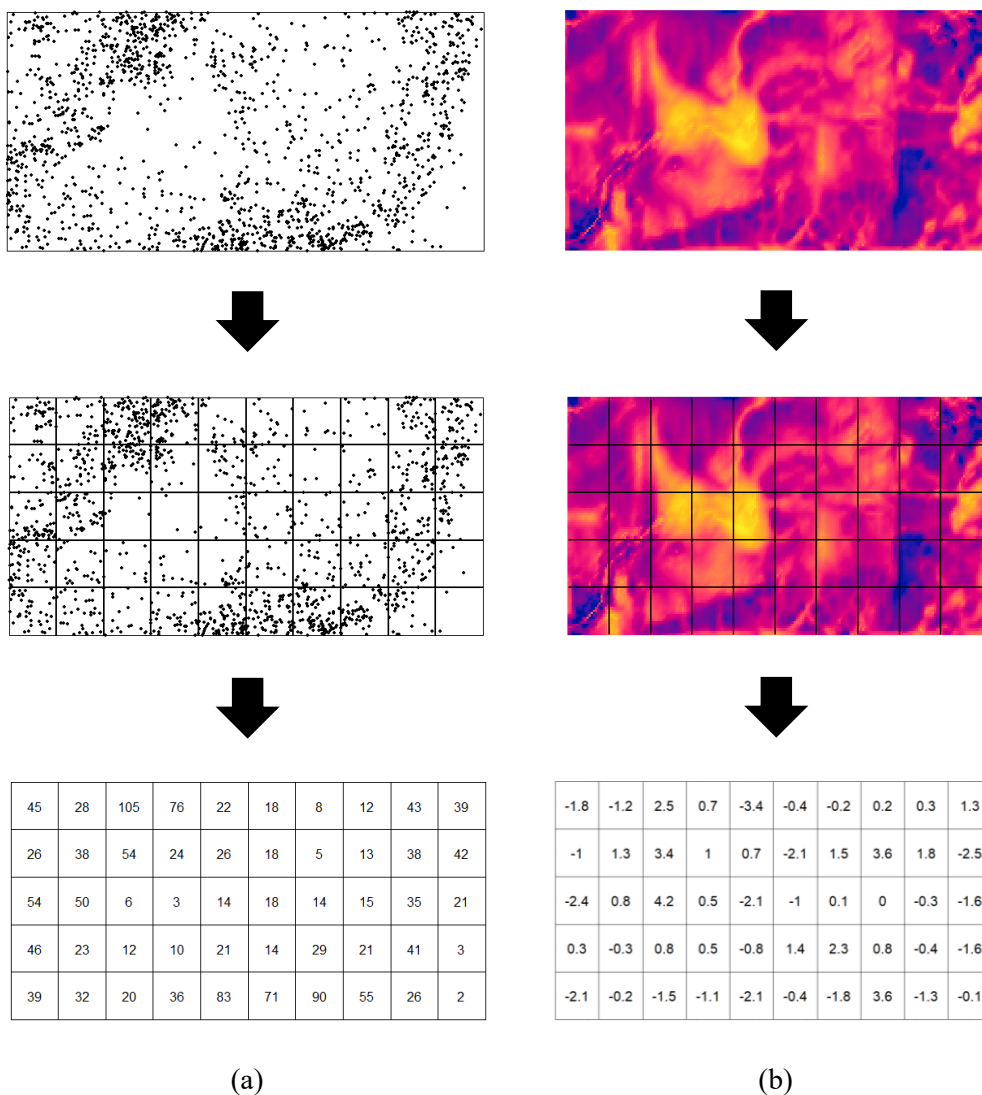
Pada Persamaan (3.1), $N(\mathbf{X} \cap B_{ij})$ merupakan jumlah titik di dalam daerah B_{ij} .

Proses diskretisasi juga dapat dilakukan terhadap variabel *covariate* sehingga didapatkan $\tilde{\mathbf{z}} = [z_{ij}^{(p)}]$ dimana:

$$z_{ij}^{(c)} = \int_{B_{ij}} z^{(c)}(u) du \approx z^{(c)}(u_{ij})|B_{ij}|, c = 1, 2, \dots, p \quad (3.2)$$

Pada Persamaan (3.2), $|B_{ij}|$ merupakan luas daerah B_{ij} dan u_{ij} adalah titik pusat *grid* B_{ij} yang mewakili keseluruhan daerah B_{ij} . Dengan demikian, setiap *grid covariate* diwakili oleh nilai *covariate* pada titik u_{ij} , yaitu titik pusat B_{ij} . Setiap hasil diskretisasi pada *grid* B_{ij} digunakan sebagai data input oleh model *probabilistic deep learning*.

Gambar 3.2 menunjukkan ilustrasi proses diskretisasi pada data (a) *point pattern* dan (b) *covariate*. Pada diskretisasi *point pattern*, *point pattern* dibagi menjadi *grid-grid* B_{ij} , kemudian dilakukan penghitungan jumlah titik di setiap *grid* tersebut. Sementara itu, diskretisasi *covariate* dilakukan dengan membagi *covariate* menjadi *grid-grid* berukuran kecil B_{ij} , kemudian didapatkan koordinat titik pusat pada setiap *grid*, yaitu u_{ij} . Setelah itu, dilakukan interpolasi berdasarkan nilai *covariate* pada titik pusat u_{ij} menggunakan persamaan (3.2) sehingga didapatkan nilai *covariate* pada setiap *grid* B_{ij} .



Gambar 3. 2 Ilustrasi Diskretisasi pada (a) *Point Pattern*, (b) *Variabel Covariate*. Hasil diskretisasi digunakan sebagai *Input* pada Model Deep Learning.

3.1.2 Multivariate LGCP menggunakan Neural Network

Proses diskretisasi pada pembahasan 3.1.1 dapat pula diterapkan ke dalam kasus *multivariate*, sehingga *multivariate point pattern* dengan jenis *multivariate* $s = 1, 2, \dots, m$ yang telah didiskretisasi dapat dituliskan sebagai $\tilde{\mathbf{x}}^{(s)} = \left[x_{ij}^{(s)} \right]$. Jumlah titik setiap *grid* B_{ij} , yaitu $\tilde{\mathbf{x}}^{(s)}$, dapat diasumsikan sebagai *random matrix* $\Lambda^{(s)} = \left[\Lambda_{ij}^{(s)} \right]$ yaitu nilai intensitas pada *grid* B_{ij} dimana $\Lambda_{ij}^{(s)} = N(\mathbf{X}^{(s)} \cap B_{ij})$. Pada kasus ini, $N(\mathbf{X}^{(s)} \cap B_{ij})$ adalah jumlah titik di dalam daerah B_{ij} untuk jenis ke- s . Dengan demikian, nilai intensitas $\Lambda^{(s)}$ dapat direpresentasikan sebagai $\tilde{\mathbf{x}}^{(s)}$, atau $x_{ij}^{(s)} = \Lambda_{ij}^{(s)}$. Dengan menggunakan diskretisasi yang serupa dengan sebelumnya, setiap *random field* pada persamaan *multivariate* LGCP dapat didiskretisasi menjadi:

$$\Lambda_{ij}^{(s)} = \int_{B_{ij}} \Lambda^{(s)}(u) du \approx \Lambda^{(s)}(u_{ij})|B_{ij}| \quad (3.3)$$

$$\mathbf{E}_{ij}^{(l)} = \int_{B_{ij}} \mathbf{E}^{(l)}(u) du \approx \mathbf{E}^{(l)}(u_{ij})|B_{ij}| \quad (3.4)$$

$$\mathbf{V}_{ij}^{(s)} = \int_{B_{ij}} \mathbf{V}^{(s)}(u) du \approx \mathbf{V}^{(s)}(u_{ij})|B_{ij}| \quad (3.5)$$

Pada persamaan (3.3), $\Lambda_{ij}^{(s)}$ menyatakan nilai intensitas yang direpresentasikan sebagai jumlah titik pada setiap *grid* B_{ij} . Berdasarkan diskretisasi pada persamaan (3.1) hingga persamaan (3.5), maka *multivariate* LGCP dapat didefinisikan sebagai:

$$\Lambda_{ij}^{(s)} = \exp \left(\beta_0^{(s)} + \sum_{c=1}^p \beta_c^{(s)} z_{ij}^{(c)} + \sum_{l=1}^a \alpha_{sl} \mathbf{E}_{ij}^{(l)} + \sigma^{(s)} \mathbf{V}_{ij}^{(s)} \right) \quad (3.6)$$

dimana $\mathbf{E}_{ij}^{(l)}$ dan $\mathbf{V}_{ij}^{(s)}$ merupakan Gaussian *random field* dengan *mean* $\mathbf{0}$ yang menangkap komponen acak pada setiap *grid* B_{ij} yang disebabkan oleh *unobserved factor* dan faktor internal setiap tipe *multivariate*. Sementara itu, $z_{ij}^{(c)}$ adalah variabel *covariate* yang dapat menjelaskan tren spasial atau pola persebaran titik pada setiap *grid* B_{ij} . Nilai tren spasial tersebut dinotasikan sebagai $\mu_{ij}^{(s)}$ yang

dipengaruhi oleh variabel *covariate* \mathbf{z} sehingga persamaan (3.6) dapat disederhanakan menjadi:

$$\Lambda_{ij}^{(s)} = \exp\left(\mu_{ij}^{(s)} + \sum_{l=1}^a \alpha_{sl} \mathbf{E}_{ij}^{(l)} + \sigma^{(s)} \mathbf{V}_{ij}^{(s)}\right) \quad (3.7)$$

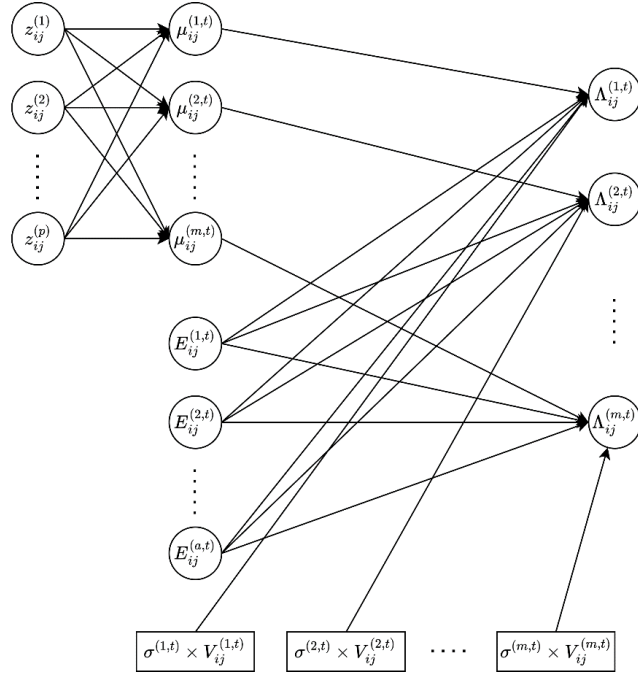
dimana Gaussian *random field* $\mathbf{E}^{(l)}$ dan $\mathbf{V}^{(s)}$ dibangkitkan berdasarkan fungsi *covariance* $\Sigma_{\mathbf{E}}$ dan $\Sigma_{\mathbf{V}}$ yang bergantung terhadap parameter $\phi^{(l)}$ dan $\psi^{(s)}$ sesuai dengan persamaan (2.14) dan persamaan (2.16). Sementara itu, *multivariate spatio-temporal* LGCP dapat didefinisikan sebagai:

$$\Lambda_{ij}^{(s,t)} = \exp\left(\mu_{ij}^{(s,t)} + \sum_{l=1}^a \alpha_{sl} \mathbf{E}_{ij}^{(l,t)} + \sigma^{(s,t)} \mathbf{V}_{ij}^{(s,t)}\right) \quad (3.8)$$

Gaussian *random field* $\mathbf{E}^{(l,t)}$ dan $\mathbf{V}^{(s,t)}$ dibangkitkan berdasarkan fungsi *covariance* $\Sigma_{\mathbf{E}}$ dan $\Sigma_{\mathbf{V}}$ yang bergantung terhadap parameter $\phi^{(l,t)}$, $\varphi^{(l,t)}$, $\rho^{(l,t)}$, $\psi^{(s,t)}$, dan $\zeta^{(s,t)}$ sesuai dengan persamaan (2.20) dan persamaan (2.21).

Persamaan (3.7) dan persamaan (3.8) menunjukkan bahwa *multivariate* LGCP memiliki parameter yang sangat banyak terutama ketika jumlah variabel *multivariate* (dan waktu) yang ingin dianalisis cukup besar, sehingga estimasi menggunakan metode parametrik sangat sulit dilakukan.

Menurut Waagepetersen dkk (2016), *multivariate* LGCP adalah model yang menghubungkan variabel *covariate* $\left(\mathbf{z}_{ij}^{(1)}(u), \mathbf{z}_{ij}^{(2)}(u), \dots, \mathbf{z}_{ij}^{(p)}(u)\right)^T$, Gaussian *random field* yang menangkap faktor yang tidak dapat dijelaskan (*unobservable factor*) $\left(\mathbf{E}_{ij}^{(1)}, \mathbf{E}_{ij}^{(2)}, \dots, \mathbf{E}_{ij}^{(a)}\right)^T$, dan Gaussian *random field* yang menangkap faktor internal pada masing-masing spesies $\sigma^{(s)} \mathbf{V}_{ij}^{(s)}$, dengan fungsi intensitas $\Lambda_{ij}^{(s)}$. Oleh sebab itu, parameter pada *multivariate* LGCP $\theta_{ij}^{(s)} = \left\{\mu_{ij}^{(s)}, \alpha_{sl}, \sigma^{(s)}, \psi^{(s)}, \phi^{(l)}\right\}$ dapat dimodelkan menggunakan *neural network* yang mengeluarkan estimasi $\hat{\theta}_{ij}^{(s)}$ pada *hidden layer*. Kemudian, estimasi $\hat{\Lambda}_{ij}^{(s)}$ didapatkan menggunakan persamaan (3.7). Diagram struktur *multivariate* LGCP ditunjukkan pada Gambar 3.3.



Gambar 3. 3 Model *Multivariate LGCP* menggunakan *Neural Network*

Konsep tersebut juga dapat diterapkan pada kasus *multivariate spatio-temporal LGCP* dengan parameter $\theta_{ij}^{(s,t)} = \{\mu_{ij}^{(s,t)}, \alpha_{sl}, \sigma^{(s,t)}, \psi^{(s,t)}, \phi^{(l)}, \varphi^{(l)}, \rho^{(l)}, \zeta^{(s)}\}$. Permasalahan *multivariate point pattern* dan *multivariate spatio-temporal point pattern* termasuk ke dalam permasalahan kompleks dan *highly non linear*, sehingga dibutuhkan model *deep learning* yang dapat memodelkan data secara *highly non linear* pula.

Parameter $\mu_{ij}^{(s)}$, $\mathbf{E}^{(l)}$, $\mathbf{V}^{(s)}$ pada persamaan (3.7) dan parameter skala korelasi $\phi_{ij}^{(l)}$ dan $\psi_{ij}^{(s)}$ diasumsikan mengikuti sebuah distribusi tertentu, yaitu:

$$\begin{aligned}
 \beta_0^{(s)} + \sum_{c=1}^p \beta_c^{(s)} z_{ij}^{(c)} &= \mu_{ij}^{(s)} \sim \text{Poisson}(\lambda_{ij}^{(s)}) \\
 \phi_{ij}^{(l)} &\sim \text{Normal}\left(\mu_{\phi_{ij}^{(l)}}, (\sigma_{\phi_{ij}^{(l)}})^2\right) \\
 \mathbf{E}^{(l)} &\sim \text{Multivariate Normal}(\mathbf{0}, \mathbf{\Sigma}_E) \\
 \psi_{ij}^{(s)} &\sim \text{Normal}\left(\mu_{\psi_{ij}^{(s)}}, (\sigma_{\psi_{ij}^{(s)}})^2\right) \\
 \mathbf{V}^{(s)} &\sim \text{Multivariate Normal}(\mathbf{0}, \mathbf{\Sigma}_V)
 \end{aligned} \tag{3.9}$$

Serupa dengan *multivariate* LGCP, Parameter $\mu_{ij}^{(s,t)}$, $\mathbf{E}^{(l,t)}$, $\mathbf{V}^{(s,t)}$, $\phi_{ij}^{(l,t)}$, $\varphi_{ij}^{(l,t)}$, $\rho_{ij}^{(l,t)}$, $\psi_{ij}^{(s,t)}$, dan $\zeta_{ij}^{(s,t)}$ pada persamaan (3.8) diasumsikan mengikuti distribusi tertentu, yaitu:

$$\begin{aligned}
\beta_0^{(s,t)} + \sum_{c=1}^p \beta_c^{(s,t)} z_{ij}^{(c)} &= \mu_{ij}^{(s,t)} \sim \text{Poisson}(\lambda_{ij}^{(s,t)}) \\
\phi_{ij}^{(l,t)} &\sim \text{Normal}\left(\mu_{\phi_{ij}}^{(l,t)}, (\sigma_{\phi_{ij}}^{(l,t)})^2\right) \\
\varphi_{ij}^{(l,t)} &\sim \text{Normal}\left(\mu_{\varphi_{ij}}^{(l,t)}, (\sigma_{\varphi_{ij}}^{(l,t)})^2\right) \\
\rho_{ij}^{(l,t)} &\sim \text{Normal}\left(\mu_{\rho_{ij}}^{(l,t)}, (\sigma_{\rho_{ij}}^{(l,t)})^2\right) \\
\mathbf{E}^{(l,t)} &\sim \text{Multivariate Normal}(\mathbf{0}, \boldsymbol{\Sigma}_E) \\
\psi_{ij}^{(s,t)} &\sim \text{Normal}\left(\mu_{\psi_{ij}}^{(s,t)}, (\sigma_{\psi_{ij}}^{(s,t)})^2\right) \\
\zeta_{ij}^{(l,t)} &\sim \text{Normal}\left(\mu_{\zeta_{ij}}^{(s,t)}, (\sigma_{\zeta_{ij}}^{(s,t)})^2\right) \\
\mathbf{V}^{(s,t)} &\sim \text{Multivariate Normal}(\mathbf{0}, \boldsymbol{\Sigma}_V)
\end{aligned} \tag{3.10}$$

Berdasarkan asumsi parameter pada persamaan (3.9) dan persamaan (3.10), maka model yang dikembangkan menggunakan konsep *probabilistic deep learning*, sehingga setiap parameter tetap mengikuti asumsi teoritisnya. Model mengeluarkan estimasi parameter $\hat{\boldsymbol{\theta}}_{ij}^{(s)}$ atau $\hat{\boldsymbol{\theta}}_{ij}^{(s,t)}$ pada *hidden layer* kemudian nilai $\hat{\boldsymbol{\Lambda}}_{ij}^{(s)}$ atau $\hat{\boldsymbol{\Lambda}}_{ij}^{(s,t)}$ direkonstruksi menggunakan persamaan (3.7) atau persamaan (3.8). Nilai $\hat{\boldsymbol{\Lambda}}_{ij}^{(s)}$ atau $\hat{\boldsymbol{\Lambda}}_{ij}^{(s,t)}$ merupakan nilai *output* akhir dari model dan diinterpretasikan sebagai ekspektasi nilai intensitas pada setiap sel B_{ij} .

Pada estimasi parameter *multivariate* LGCP, model *probabilistic neural network* yang dikembangkan mendapatkan data input $\mathcal{X} = \{\boldsymbol{\Lambda}^{(s)}\}$, sedangkan pada kasus *multivariate spatio-temporal* LGCP model *probabilistic neural network* mendapatkan data input $\mathcal{X} = \{\boldsymbol{\Lambda}^{(s)}, \mathbf{t}\}$. Model juga dapat menerima data variabel *covariate* $\tilde{\mathbf{z}}$ apabila tersedia. Secara umum, model terdiri dari 2 bagian, yaitu *encoder* yang memberikan estimasi $\hat{\boldsymbol{\theta}}_{ij}^{(s)}$ atau $\hat{\boldsymbol{\theta}}_{ij}^{(s,t)}$, dan *decoder function* berdasarkan persamaan (3.7) atau persamaan (3.8) yang memberikan estimasi $\hat{\boldsymbol{\Lambda}}_{ij}^{(s)}$

atau $\widehat{\Lambda}_{ij}^{(s,t)}$. Dengan demikian, secara matematis model yang dikembangkan untuk estimasi parameter pada model *multivariate* LGCP memiliki persamaan:

$$\begin{aligned}\widehat{\theta}_{ij}^{(s)} &= \text{EncoderNetwork}(\mathcal{X}|\mathbf{w}) \\ \widehat{\Lambda}_{ij}^{(s)} &= \text{DecoderFunction}(\widehat{\theta}_{ij}^{(s)})\end{aligned}\tag{3.11}$$

Model *probabilistic deep learning* secara umum memiliki persamaan sebagai berikut:

$$\begin{aligned}\widehat{h}_{ij}^{(1,k)} &= f_1 \left(b_{1,k} + \sum_{s=1}^m w_{1,s,k} \Lambda_{ij}^{(s)} \right); k = 1, 2, \dots, \mathcal{L}_2 \\ \widehat{h}_{ij}^{(2,k)} &= f_2 \left(b_{2,k} + \sum_{k'=1}^{\mathcal{L}_2} w_{2,k',k} \widehat{h}_{ij}^{(1,k')} \right); k = 1, 2, \dots, \mathcal{L}_3 \\ &\vdots \\ \widehat{\lambda}_{ij}^{(s)} &= f_{\text{exp}} \left(b_{n_\lambda, s} + \sum_{k'=1}^{\mathcal{L}_{(n_\lambda-1)}} w_{n_\lambda, k', s} \widehat{h}_{ij}^{((n_\lambda-1), k')} \right); s = 1, 2, \dots, m \\ \mathbb{E} \left(\widehat{\mu}_{ij}^{(s)} \right) &= \widehat{\lambda}_{ij}^{(s)} \\ &\vdots \\ \widehat{\mu}_{\phi_{ij}}^{(l_1)} &= f_{\text{sig}} \left(b_{n_\phi, l_1} + \sum_{k'=1}^{\mathcal{L}_{(n_\phi-1)}} w_{n_\phi, k', l_1} \widehat{h}_{ij}^{((n_\phi-1), k')} \right); l_1 = 1, 2, \dots, a \\ \left(\widehat{\sigma}_{\phi_{ij}}^{(l_2)} \right)^2 &= f_{\text{splus}} \left(b_{n_\phi, l_2} + \sum_{k'=1}^{\mathcal{L}_{(n_\phi-1)}} w_{n_\phi, k', l_2} \widehat{h}_{ij}^{((n_\phi-1), k')} \right); \\ l_2 &= (a + 1), \dots, (a \times 2) \\ \widehat{\phi}_{ij}^{(l)} &\sim \text{Normal} \left(\widehat{\mu}_{\phi_{ij}}^{(l_1)}, \left(\widehat{\sigma}_{\phi_{ij}}^{(l_2)} \right)^2 \right) \\ &\vdots\end{aligned}$$

$$\hat{\mu}_{\psi_{ij}}^{(s_1)} = f_{\text{sig}} \left(b_{n_\psi, s_1} + \sum_{k'=1}^{\mathcal{L}(n_\psi-1)} w_{n_\psi, k', s_1} \hat{h}_{ij}^{((n_\psi-1), k')} \right); s_1 = 1, 2, \dots, m$$

$$\left(\hat{\sigma}_{\psi_{ij}}^{(s_2)} \right)^2 = f_{\text{splus}} \left(b_{n_\psi, s_2} + \sum_{k'=1}^{\mathcal{L}(n_\psi-1)} w_{n_\psi, k', s_2} \hat{h}_{ij}^{((n_\psi-1), k')} \right);$$

$$s_2 = (m + 1), \dots, (m \times 2)$$

$$\hat{\psi}_{ij}^{(s)} \sim \text{Normal} \left(\hat{\mu}_{\psi_{ij}}^{(s_1)}, \left(\hat{\sigma}_{\psi_{ij}}^{(s_2)} \right)^2 \right)$$

⋮

$$\Sigma_E = \left(\mathbf{w}_{n_E} \times \hat{\mathbf{h}}^{(n_E-1)} \right)$$

$$\hat{\mathbf{E}}^{(l)} \sim \text{Multivariate Normal}(\mathbf{0}, \Sigma_E)$$

⋮

$$\Sigma_V = \left(\mathbf{w}_{n_V} \times \hat{\mathbf{h}}^{(n_V-1)} \right)$$

$$\hat{\mathbf{V}}^{(s)} \sim \text{Multivariate Normal}(\mathbf{0}, \Sigma_V)$$

⋮

$$\hat{\alpha} \hat{\mathbf{E}}_{ij}^{(s)} = \left(\sum_{l=1}^a w_{n_{\alpha E}, l, s} \hat{\mathbf{E}}_{ij}^{(l)} \right); l = 1, 2, \dots, a; w_{n_{\alpha E}, l, s} = \hat{\alpha}_{sl}$$

$$\hat{\sigma} \hat{\mathbf{V}}_{ij}^{(s)} = \left(w_{n_{\sigma V}, 1, s} \times \mathbf{V}_{ij}^{(s)} \right); s = 1, 2, \dots, m; w_{n_{\sigma V}, 1, s} = \hat{\sigma}^{(s)}$$

$$\hat{\Lambda}_{ij}^{(s)} = \hat{\mu}_{ij}^{(s)} \times f_{\text{exp}} \left(\hat{\alpha} \hat{\mathbf{E}}_{ij}^{(l)} + \hat{\sigma} \hat{\mathbf{V}}_{ij}^{(s)} \right)$$

Pada persamaan di atas, n_λ , n_ϕ , n_ψ , n_E , dan n_V adalah urutan *layer* untuk estimasi parameter $\boldsymbol{\mu}$, $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, \mathbf{E} , dan \mathbf{V} . Notasi \mathcal{L} menunjukkan jumlah *neuron* pada suatu *layer* sedangkan bobot $w_{n_{\alpha E}, l, s}$ dan $w_{n_{\sigma V}, 1, s}$ dapat direpresentasikan sebagai estimasi $\hat{\alpha}_{sl}$ dan $\hat{\sigma}^{(s)}$.

3.1.3 Pembagian Dataset

Sebelum pelatihan *deep learning* dimulai, data input dibagi menjadi data latih dan data validasi secara acak. Rasio pembagian yang digunakan adalah 70% data dipakai untuk data latih, dan 30% data dipakai untuk data validasi. Data latih dipakai untuk proses pelatihan model *deep learning* sebanyak 150 *epoch* pada studi simulasi dan 50 *epoch* pada studi terapan. Kebaikan model dan *loss function* pada pelatihan model dihitung menggunakan metode MSE. Kebaikan model untuk estimasi parameter *multivariate* LGCP dihitung dengan persamaan:

$$\text{MSE} = \frac{1}{(m \times d_1 \times d_2)} \sum_{s=1}^m \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \left(\Lambda_{ij}^{(s)} - \widehat{\Lambda}_{ij}^{(s)} \right)^2 \quad (3.12)$$

Sedangkan MSE pada model untuk estimasi parameter *multivariate spatio-temporal* LGCP adalah:

$$\text{MSE} = \frac{1}{(m \times T \times d_1 \times d_2)} \sum_{s=1}^m \sum_{t=1}^T \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \left(\Lambda_{ij}^{(s,t)} - \widehat{\Lambda}_{ij}^{(s,t)} \right)^2 \quad (3.13)$$

Proses pelatihan dan estimasi parameter diulang sebanyak jumlah n_{trial} , sehingga rata-rata keseluruhan nilai MSE dan rentang kepercayaannya adalah:

$$\text{Mean MSE} = \frac{1}{n_{\text{trial}}} \sum_{j=1}^{n_{\text{trial}}} \text{MSE}_j \quad (3.14)$$

$$\text{Mean MSE} \pm Z_{\alpha/2} \frac{s(\text{MSE})}{\sqrt{n_{\text{trial}}}} \quad (3.15)$$

dimana MSE_j merupakan nilai MSE pada iterasi ke-j, $s(\text{MSE})$ merupakan standar deviasi dari n_{trial} kali iterasi, dan $Z_{\alpha/2}$ merupakan titik kritis dari normal standar dengan $\alpha = 0.05$, yaitu sebesar 1.96.

3.2 Studi Simulasi

Studi simulasi digunakan untuk mengevaluasi model *probabilistic deep learning*. *Probabilistic deep learning* memberikan estimasi parameter $\widehat{\theta}_{ij}^{(s)}$. Hasil estimasi tersebut digunakan untuk merekonstruksi nilai *output* akhir $\widehat{\Lambda}_{ij}^{(s)}$

berdasarkan persamaan (3.7). Hasil estimasi parameter $\widehat{\theta}_{ij}^{(s)}$ dibandingkan dengan nilai parameter sebenarnya $\theta_{ij}^{(s)}$ untuk mengetahui kebaikan estimasi parameter. Subbab 3.2.1 hingga 3.2.4 membahas tentang rancangan desain simulasi pada penelitian ini.

3.2.1 Sumber Data dan Variabel Penelitian

Data simulasi didapatkan dengan membangkitkan data *multivariate point pattern* berdasarkan nilai parameter *multivariate* LGCP yang telah ditentukan sebelumnya. Proses pembangkitan data simulasi dan nilai parameternya mengikuti langkah-langkah studi simulasi pada Choiruddin dkk (2020) dengan jumlah tipe *multivariate* yang dibangkitkan adalah $m = 10$ dan jumlah *latent field* yang digunakan adalah $a = 4$. *Point pattern* bangkitan pada studi simulasi dapat dituliskan sebagai:

$$\mathcal{D}_{sim} = \{\mathbf{x}^{(s)}; s = 1, 2, \dots, m\} \quad (3.16)$$

Proses pembangkitan data simulasi adalah sebagai berikut:

- a) Menentukan nilai parameter sebenarnya $\theta_{sim} = \{\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{\phi}, \boldsymbol{\psi}\}$ dan ukuran *window* W .
- b) Membangkitkan Gaussian *random field* \mathbf{E} berdasarkan $\boldsymbol{\phi}$ dan \mathbf{V} berdasarkan $\boldsymbol{\psi}$. \mathbf{E} dan \mathbf{V} berdistribusi multivariate normal dengan *mean* $\mathbf{0}$ dan *covariance matrix* $\boldsymbol{\Sigma}_{\mathbf{E}}$ (Persamaan 2.14) dan $\boldsymbol{\Sigma}_{\mathbf{V}}$ (Persamaan 2.16) pada setiap titik koordinat u .
- c) Melakukan operasi $\sum_{l=1}^a \alpha_{sl} \mathbf{E}^{(l)}(u)$ dan $\sigma^{(s)} \mathbf{V}^{(s)}(u)$ sesuai dengan persamaan (3.7) untuk mendapatkan komponen interaksi antar tipe dan dalam tipe *multivariate*. Komponen deterministik $\mu^{(s)}(u)$ didapatkan dengan menggunakan persamaan $\mu^{(s)}(u) = \log(\mu^{(s)}) - \frac{\boldsymbol{\alpha}_s \boldsymbol{\alpha}_s^T}{2} + \frac{(\sigma^{(s)})^2}{2}$.
- d) Melakukan operasi $\boldsymbol{\Lambda}^{(s)}(u) = \exp(\mu^{(s)}(u) + \sum_{l=1}^a \alpha_{sl} \mathbf{E}^{(l)}(u) + \sigma^{(s)} \mathbf{V}^{(s)}(u))$ sesuai dengan persamaan (3.7) sehingga didapatkan $\boldsymbol{\Lambda}(u)$.
- e) Mensimulasikan Poisson process dengan intensitas $\boldsymbol{\Lambda}^{(s)}(u)$ sehingga didapatkan *multivariate point pattern* $\mathbf{x}^{(s)}$. *Window* W untuk masing-masing spesies dipecah menjadi *grid-grid* berukuran sangat kecil sehingga setiap

koordinat u dapat diwakili oleh sebuah *grid* tersebut. Kemudian, dibangkitkan sejumlah titik pada setiap *grid* yang ekspektasi jumlah titiknya mengikuti distribusi Poisson dengan *rate* $\Lambda^{(s)}(u)$. Setiap pembangkitan titik pada suatu *grid* dilakukan secara independen. Semua *grid* digabungkan untuk membentuk *point pattern* $\mathbf{x}^{(s)}$.

Data simulasi menggunakan ukuran *window* W sebesar 1×1 . Nilai parameter $\theta_{sim} = \{\boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{\phi}, \boldsymbol{\psi}\}$ untuk membangkitkan data *multivariate point pattern* adalah:

$$\boldsymbol{\alpha} = \begin{bmatrix} \sqrt{0.5} & 0.10 & -1 & 0 \\ 0 & 0 & -0.70 & 1 \\ 0 & -0.15 & \sqrt{0.5} & 0.10 \\ -1 & 0 & 0 & 0 \\ -0.70 & 1 & 0 & -0.15 \\ \sqrt{0.5} & 0.10 & -1 & 0 \\ 0 & 0 & -0.70 & 1 \\ 0 & -0.15 & \sqrt{0.5} & 0.10 \\ -1 & 0 & 0 & 0 \\ -0.70 & 1 & 0 & -0.15 \end{bmatrix} \quad (3.17)$$

$$\boldsymbol{\mu} = (5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000)^T \quad (3.18)$$

$$\boldsymbol{\sigma}^2 = (1, 1, 1.5, 1, 0.2, 0.2, 1, 1.5, 1.5, 1.5)^T \quad (3.19)$$

$$\boldsymbol{\phi} = (0.2, 0.3, 0.3, 0.5)^T \quad (3.20)$$

$$\boldsymbol{\psi} = (0.1, 0.2, 0.2, 0.3, 0.4, 0.4, 0.5, 0.6, 0.6, 0.7)^T \quad (3.21)$$

Jumlah titik pada setiap jenis *multivariate* adalah 5000 sehingga parameter $\mu^{(s)}$ bernilai 5000. Variabel $\boldsymbol{\phi}$ dan $\boldsymbol{\psi}$ merupakan parameter skala korelasi untuk Gaussian *random field* \mathbf{E} dan \mathbf{V} . Model *probabilistic deep learning* mengeluarkan nilai estimasi parameter $\hat{\theta}_{sim} = \{\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}}\}$ dengan $\hat{\boldsymbol{\mu}} = \left\{ \sum_{i=1}^{100} \sum_{j=1}^{200} \mu_{ij}^{(1)}, \dots, \sum_{i=1}^{100} \sum_{j=1}^{200} \mu_{ij}^{(m)} \right\}$, sehingga kebaikan model dalam memprediksi setiap parameter tersebut dapat diukur menggunakan Root Mean Squared Error (RMSE). Jika nilai parameter sebenarnya untuk $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\sigma}}$, dan $\hat{\boldsymbol{\psi}}$

dinotasikan sebagai ω dan nilai estimasi parameternya dinotasikan sebagai $\hat{\omega}$, maka RMSE dapat dihitung menggunakan persamaan:

$$\text{RMSE}(\hat{\omega}^{(s)}) = \sqrt{\sum_{k=1}^{n_{\text{trial}}} \frac{(\hat{\omega}_k^{(s)} - \omega^{(s)})^2}{n_{\text{trial}}}}; s = 1, 2, \dots, m \quad (3.22)$$

Proses estimasi parameter dilakukan berkali-kali sebanyak jumlah n_{trial} , sehingga rata-rata RMSE pada seluruh tipe *multivariate* beserta rentang kepercayaannya dapat dihitung menggunakan persamaan:

$$\text{Mean RMSE}(\hat{\omega}) = \frac{1}{m} \sum_{s=1}^m \text{RMSE}(\hat{\omega}^{(s)}) \quad (3.23)$$

$$\text{Mean RMSE}(\hat{\omega}) \pm Z_{\alpha/2} \frac{s(\hat{\omega})}{\sqrt{m}} \quad (3.24)$$

dimana $Z_{\alpha/2}$ merupakan titik kritis dari normal standar dengan $\alpha = 0.05$, yaitu sebesar 1.96, dan $s(\hat{\omega})$ merupakan standar deviasi dari seluruh tipe *multivariate*. Proses serupa juga diterapkan terhadap $\hat{\alpha}$.

$$\text{RMSE}(\hat{\alpha}_{sl}) = \sqrt{\sum_{k=1}^{n_{\text{trial}}} \frac{(\hat{\alpha}_{slk} - \alpha_{sl})^2}{n_{\text{trial}}}} \quad (3.25)$$

$$\text{Mean RMSE}(\hat{\alpha}) = \frac{1}{m \times a} \sum_{s=1}^m \sum_{l=1}^a \text{RMSE}(\hat{\alpha}_{sl}) \quad (3.26)$$

$$\text{Mean RMSE}(\hat{\alpha}) \pm Z_{\alpha/2} \frac{s(\hat{\alpha})}{\sqrt{m \times a}} \quad (3.27)$$

Sementara itu, RMSE untuk parameter $\hat{\phi}$ adalah:

$$\text{RMSE}(\hat{\phi}^{(l)}) = \sqrt{\sum_{k=1}^{n_{\text{trial}}} \frac{(\hat{\phi}_k^{(l)} - \phi^{(l)})^2}{n_{\text{trial}}}}; l = 1, 2, \dots, a \quad (3.28)$$

$$\text{Mean RMSE}(\hat{\phi}) = \frac{1}{a} \sum_{s=1}^m \text{RMSE}(\hat{\phi}^{(s)}) \quad (3.29)$$

$$\text{Mean RMSE}(\hat{\phi}) \pm Z_{\alpha/2} \frac{s(\hat{\phi})}{\sqrt{a}} \quad (3.30)$$

3.2.2 Struktur Data

Data simulasi merupakan data bangkitan yang memiliki struktur data seperti ditunjukkan pada Tabel 3.1.

Tabel 3. 1 Struktur Data Bangkitan pada Studi Simulasi

Jenis	$\mathbf{u}^{(1)}$	$\mathbf{u}^{(2)}$
1	$u_{1,1}^{(1)}$	$u_{1,1}^{(2)}$
1	$u_{1,2}^{(1)}$	$u_{1,2}^{(2)}$
...
1	$u_{1,N(\mathbf{X}^{(s)})}^{(1)}$	$u_{1,N(\mathbf{X}^{(s)})}^{(2)}$
...
s	$u_{s,1}^{(1)}$	$u_{s,1}^{(2)}$
...
m	$u_{m,N(\mathbf{X}^{(s)})}^{(1)}$	$u_{m,N(\mathbf{X}^{(s)})}^{(2)}$

Pada Tabel 3.1, variabel $s = 1, 2, \dots, m$ menunjukkan jenis variabel *multivariate*, sedangkan variabel $N(\mathbf{X}^{(s)})$ adalah jumlah titik pada suatu *point pattern* dengan tipe ke- s . Sementara itu, variabel $u^{(1)}$ dan $u^{(2)}$ merupakan variabel yang menunjukkan lokasi koordinat pada sumbu x dan sumbu y di bidang dua dimensi. *Library* “spatstat” dipakai untuk mengubah data tabular tersebut menjadi onjek *point pattern*.

Proses diskretisasi dilakukan terhadap data bangkitan \mathcal{D}_{sim} sesuai yang dijelaskan pada subbab 3.1.1 sehingga didapatkan data *point pattern* yang telah didiskretisasi $\tilde{\mathcal{D}}_{sim}$. Tabel 3.2 menunjukkan struktur data *point pattern* setelah

dilakukan diskretisasi. Pada Tabel 3.2, $\Lambda_{ij}^{(s)}$ merupakan jumlah titik di dalam wilayah *grid* B_{ij} untuk jenis $s = 1, 2, \dots, m$ dan diasumsikan sebagai $N(\mathbf{X}^{(s)} \cap B_{ij})$.

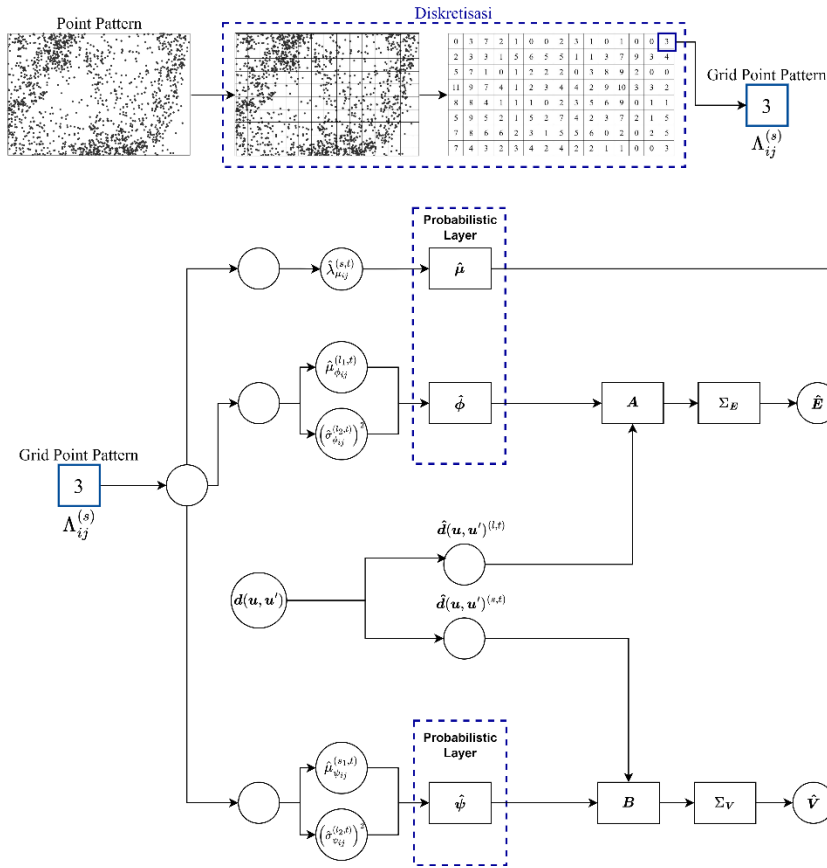
Tabel 3. 2 Struktur Data *Point Pattern* Setelah Proses Diskretisasi pada Data Simulasi

Grid i	Grid j	$\Lambda_{ij}^{(1)}$...	$\Lambda_{ij}^{(m)}$
1	1	$\Lambda_{1,1}^{(1)}$...	$\Lambda_{1,1}^{(m)}$
1	2	$\Lambda_{1,2}^{(1)}$...	$\Lambda_{1,2}^{(m)}$
...
1	d_2	$\Lambda_{1,d_2}^{(1)}$...	$\Lambda_{1,d_2}^{(m)}$
...
49	1	$\Lambda_{49,1}^{(1)}$...	$\Lambda_{49,1}^{(m)}$
49	2	$\Lambda_{49,2}^{(1)}$...	$\Lambda_{49,2}^{(m)}$
...
d_1	d_2	$\Lambda_{d_1,d_2}^{(1)}$...	$\Lambda_{d_1,d_2}^{(m)}$

3.2.3 Proses Pelatihan

Langkah pertama adalah membangkitkan data *multivariate point pattern* berdasarkan nilai parameter pada Persamaan (3.17) hingga persamaan (3.21). Model menerima data input berupa hasil diskretisasi *point pattern* $\mathcal{X} = \{\tilde{\mathcal{D}}_{sim}\}$ dan memberikan estimasi parameter $\hat{\theta}_{sim}$ pada *hidden layer*. Selanjutnya, dilakukan rekonstruksi menggunakan persamaan (3.7) untuk mendapatkan nilai *output* $\hat{\Lambda}_{ij}^{(s)}$.

Dataset $\tilde{\mathcal{D}}_{sim}$ dibagi secara acak menjadi data latih $\tilde{\mathcal{D}}_{sim_{latih}}$ dan data validasi $\tilde{\mathcal{D}}_{sim_{val}}$. Model *deep learning* dilatih menggunakan data $\tilde{\mathcal{D}}_{sim_{latih}}$ sebanyak 150 *epoch* dan dihitung nilai *loss function* dan kebaikan model menggunakan MSE berdasarkan $\tilde{\mathcal{D}}_{sim_{latih}}$, $\tilde{\mathcal{D}}_{sim_{val}}$, dan $\tilde{\mathcal{D}}_{sim}$. Selain itu, dihitung pula kebaikan hasil estimasi parameter $\hat{\theta}_{sim}$ menggunakan RMSE. Ilustrasi model *probabilistic deep learning* pada studi simulasi digambarkan pada Gambar 3.4.



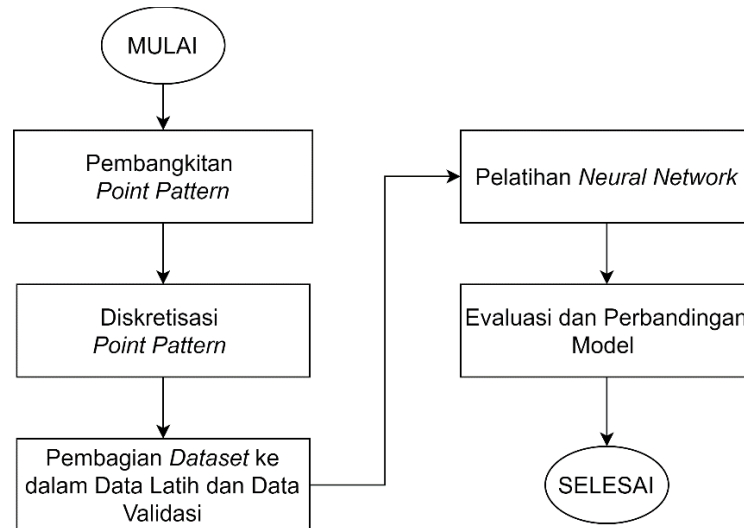
Gambar 3. 4 Ilustrasi Probabilistic Deep Learning pada Studi Simulasi

3.2.4 Tahapan Analisis pada Studi Simulasi

Diagram alir tahapan analisis pada studi simulasi ditunjukkan oleh Gambar 3.5. Langkah-langkah analisis yang dilakukan pada studi simulasi adalah:

1. Pembangkitan data *multivariate point pattern* berdasarkan nilai parameter θ_{sim} yang telah ditentukan sebelumnya (Persamaan (3.17) hingga persamaan (3.21)).
2. Diskretisasi data *point pattern* sesuai pembahasan pada 3.1.1.
3. Pembagian *dataset* ke dalam data latih dan data validasi secara acak seperti pembahasan pada 3.1.3.
4. Pelatihan model *probabilistic deep learning* seperti yang dibahas pada 3.2.3.
5. Evaluasi model berdasarkan *point pattern* yang telah dibangkitkan. Model menerima data input $\mathcal{X} = \{\tilde{\mathcal{D}}_{sim}\}$ dan memberikan estimasi parameter $\hat{\theta}_{sim}^{\square}$. Hasil estimasi tersebut digunakan untuk merekonstruksi nilai *output* akhir $\hat{\Lambda}_{ij}^{(s)}$ berdasarkan persamaan (3.7). Nilai yang dievaluasi adalah:

- a. $\Lambda_{ij}^{(s)}$ dan $\widehat{\Lambda}_{ij}^{(s)}$ menggunakan MSE pada persamaan (3.12) hingga persamaan (3.15),
- b. parameter θ_{sim}^{\square} dan $\widehat{\theta}_{sim}^{\square}$ menggunakan RMSE pada persamaan (3.22) hingga persamaan (3.30). Nilai MSE dan RMSE yang kecil menandakan bahwa model memberikan estimasi yang baik.



Gambar 3. 5 Diagram Alir Tahapan Analisis pada Studi Simulasi

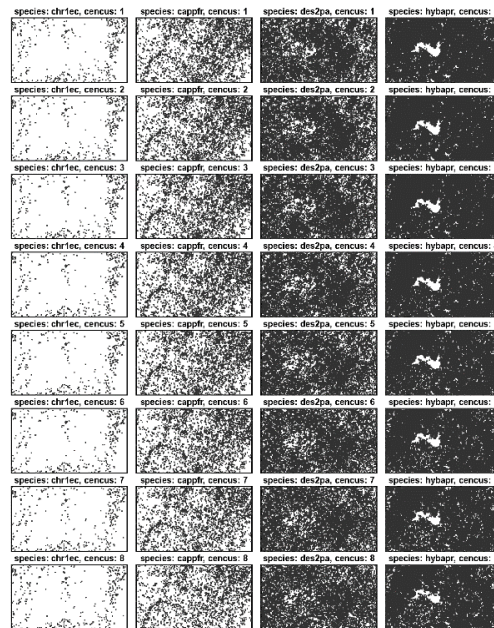
3.3 Studi Terapan

Model *probabilistic deep learning* digunakan untuk melakukan estimasi parameter dan analisis pada data sekunder. *Probabilistic deep learning* memberikan estimasi parameter $\widehat{\theta}_{ij}^{(s,t)}$. Hasil estimasi tersebut digunakan untuk merekonstruksi nilai *output* akhir $\widehat{\Lambda}_{ij}^{(s,t)}$ berdasarkan persamaan (3.8). Penjelasan terkait studi terapan dituliskan pada subbab 3.3.1 hingga 3.3.4.

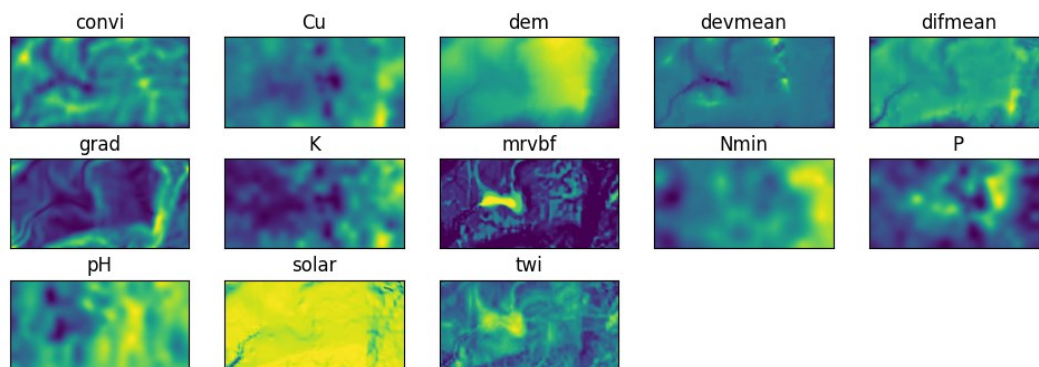
3.3.1 Sumber Data dan Variabel Penelitian

Data sekunder yang digunakan pada penelitian ini adalah Barro Colorado Island (BCI) *dataset* yang berisi lokasi koordinat setiap pohon pada hutan hujan tropis di Pulau Barro Colorado, Panama (Condit dkk, 2019). *Dataset* ini menyediakan lokasi setiap pohon dari ratusan spesies yang disensus sebanyak 8 kali dari tahun 1983 hingga 2015. Lokasi pohon direpresentasikan sebagai titik pada *point pattern* yang memiliki *observation window* W berbentuk persegi panjang dengan ukuran $500 \times 1000 \text{ m}^2$. *Dataset* ini memiliki banyak spesies pohon dan

setiap spesies tersebut memiliki $T = 8$ replikasi yang mencerminkan waktu pelaksanaan sensus. Oleh sebab itu, analisis pada *dataset* ini tergolong sebagai permasalahan *multivariate spatio-temporal point pattern*. Gambar 3.6 menunjukkan contoh *point pattern* dari keempat spesies pohon yang disensus sebanyak 8 kali.



Gambar 3. 6 Contoh *Point Pattern* dari 4 Spesies Pohon yang Disensus Sebanyak 8 Kali (Condit dkk, 2019)



Gambar 3. 7 Variabel *Covariate* yang Tersedia pada BCI Dataset

Selain lokasi pohon berupa *point pattern*, BCI *dataset* juga menyediakan $p = 13$ variabel *covariate* yang mencerminkan kondisi tanah pada *window W*.

Variabel-variabel *covariate* tersebut ditunjukkan pada Gambar 3.7 sedangkan penjelasannya dirangkum di dalam Tabel 3.3.

Tabel 3. 3 Penjelasan tentang Variabel *Covariate*

No.	Nama	Penjelasan
1	convi	Indeks konvergensi menuju <i>cell</i> pusat
2	Cu	Kandungan tembaga dalam tanah (mg/kg tanah)
3	dem	Ketinggian tanah
4	devmean	Simpangan dari <i>mean</i> pada <i>search radius</i> 15 piksel
5	difmean	Selisih dari <i>mean</i> pada <i>search radius</i> 15 piksel
6	grad	Kemiringan tanah
7	K	Kandungan Potassium dalam tanah (mg/kg tanah)
8	mrvbf	<i>Multi-resolution index</i> pada dataran dasar lembah
9	Nmin	Kebutuhan mineralisasi Nitrogen setelah 30 hari inkubasi (mg/kg tanah)
10	P	Kandungan fosfor dalam tanah (mg/kg tanah)
11	pH	Kandungan pH dalam tanah (mg/kg tanah)
12	solar	Rata-rata radiasi matahari tahunan yang masuk
13	twi	<i>Topographic wetness index</i>

Secara matematis, *spatial point pattern* dan variabel *covariate* pada BCI dataset dapat dituliskan sebagai:

$$\mathcal{D}_{bci} = \{\mathbf{x}^{(s,t)}; s = 1,2, \dots, m; t = 1,2, \dots, T\} \quad (3.31)$$

$$\mathcal{Z}_{bci} = \{(z^{(1)}(u), z^{(2)}(u), \dots, z^{(p)}(u)); p = 1,2, \dots, 13\} \quad (3.32)$$

Selain itu, digunakan pula variabel \mathcal{T} yang menunjukkan waktu observasi. Variabel tersebut didefinisikan sebagai:

$$\mathcal{T}_{bci} = \{1,2, \dots, T\} \quad (3.33)$$

Analisis pada *dataset* BCI tergolong sebagai *multivariate spatio-temporal* LGCP sehingga parameter yang harus diestimasi adalah $\theta_{bci} = \{\mu, \alpha, \sigma, \phi, \psi, \varphi, \rho, \zeta\}$. Studi terapan pada penelitian ini dibagi ke dalam dua jenis analisis, yaitu:

1. Analisis pada $m = 25$ spesies pohon yang berjumlah paling banyak dan diobservasi pada $T = 4$ waktu sensus. Pada analisis ini, *probabilistic deep learning* dan model Mateu & Jalilian (2022) dilatih sebanyak $n_{trial} = 100$ kali.

2. Analisis pada $m = 100$ spesies pohon yang berjumlah paling banyak dan diobservasi pada $T = 8$ waktu sensus. Pada analisis ini, *probabilistic deep learning* dan model Mateu & Jalilian (2022) dilatih sebanyak $n_{trial} = 5$ kali.

Setiap kali pelatihan selesai, kedua model tersebut dipakai untuk melakukan prediksi dan dievaluasi menggunakan MSE sehingga didapatkan n_{trial} nilai MSE pada setiap model. Perbandingan performa kedua model dilakukan dengan membandingkan rata-rata dari keseluruhan MSE tersebut.

3.3.2 Struktur Data

Dataset BCI menyediakan data *point pattern* untuk setiap spesies dan waktu observasi yang memiliki struktur data pada Tabel 3.4. Pada Tabel 3.4, variabel $s = 1, 2, \dots, m$ menunjukkan jenis spesies yang dianalisis, variabel $t = 1, 2, \dots, T$ merepresentasikan waktu observasi, sedangkan variabel $N(\mathbf{X}^{(s,t)})$ adalah jumlah titik pada *point pattern* milik spesies s dan waktu observasi t . Sementara itu, variabel $u^{(1)}$ dan $u^{(2)}$ merupakan variabel yang menunjukkan lokasi koordinat pada sumbu x dan sumbu y pada bidang dua dimensi. Sementara itu, variabel *covariate* direpresentasikan sebagai gambar berukuran 500×1000 piksel sehingga nilai pada setiap piksel mencerminkan nilai dari *covariate* pada titik piksel tersebut.

Tabel 3. 4 Struktur Data *Point Pattern* pada BCI Dataset

Waktu	Spesies	$u^{(1)}$	$u^{(2)}$
1	1	$u_{1,1,1}^{(1)}$	$u_{1,1,1}^{(2)}$
1	1	$u_{1,1,2}^{(1)}$	$u_{1,1,2}^{(2)}$
...
1	1	$u_{1,1,N(\mathbf{X}^{(s,t)})}^{(1)}$	$u_{1,1,N(\mathbf{X}^{(s,t)})}^{(2)}$
...
1	2	$u_{2,1,1}^{(1)}$	$u_{2,1,1}^{(2)}$
...
1	2	$u_{2,1,N(\mathbf{X}^{(s,t)})}^{(1)}$	$u_{2,1,N(\mathbf{X}^{(s,t)})}^{(2)}$
...
3	14	$u_{14,3,1}^{(1)}$	$u_{14,3,1}^{(2)}$

Tabel 3. 4 Lanjutan Struktur Data *Point Pattern* pada BCI Dataset

Waktu	Spesies	$\mathbf{u}^{(1)}$	$\mathbf{u}^{(2)}$
...
3	14	$u_{14,3,N(X^{(s,t)})}^{(1)}$	$u_{14,3,N(X^{(s,t)})}^{(2)}$
...
t	s	$u_{s,t,1}^{(1)}$	$u_{s,t,1}^{(2)}$
...
T	m	$u_{m,T,N(X^{(s,t)})}^{(1)}$	$u_{m,T,N(X^{(s,t)})}^{(2)}$

Proses diskretisasi diterapkan terhadap data *point pattern* \mathcal{D}_{bci} dan variabel *covariate* \mathcal{Z}_{bci} seperti yang dijelaskan pada subbab 3.1.1. Hasil diskretisasi *point pattern* dan *covariate* dinotasikan sebagai $\tilde{\mathcal{D}}_{bci}$ dan $\tilde{\mathcal{Z}}_{bci}$. Tabel 3.5 menunjukkan struktur data *point pattern* setelah dilakukan diskretisasi. Pada Tabel 3.5, $\Lambda_{ij}^{(s,t)}$ diasumsikan sebagai $N(X^{(s,t)} \cap B_{ij})$ yang merupakan jumlah titik di dalam wilayah *grid* B_{ij} untuk spesies $s = 1, 2, \dots, m$ yang diobservasi pada $t = 1, 2, \dots, T$. Struktur data variabel *covariate* setelah proses diskretisasi ditunjukkan pada Tabel 3.6. Pada Tabel 3.6, $z_{ij}^{(c)}, c = 1, 2, \dots, p$ menunjukkan nilai *covariate* pada setiap variabel di daerah *grid* B_{ij} .

Tabel 3. 5 Struktur Data *Point Pattern* Setelah Proses Diskretisasi pada *Dataset* BCI

Grid i	Grid j	Waktu t	$\Lambda_{ij}^{(1,t)}$...	$\Lambda_{ij}^{(m,t)}$
1	1	1	$\Lambda_{1,1}^{(1,1)}$...	$\Lambda_{1,1}^{(m,1)}$
1	2	1	$\Lambda_{1,2}^{(1,1)}$...	$\Lambda_{1,2}^{(m,1)}$
...
1	d_2	1	$\Lambda_{1,d_2}^{(1,1)}$...	$\Lambda_{1,d_2}^{(m,1)}$
...
29	1	t	$\Lambda_{29,1}^{(1,t)}$...	$\Lambda_{29,1}^{(m,t)}$
29	2	t	$\Lambda_{29,2}^{(1,t)}$...	$\Lambda_{29,2}^{(m,t)}$
...
d_1	d_2	T	$\Lambda_{d_1,d_2}^{(1,T)}$...	$\Lambda_{d_1,d_2}^{(m,T)}$

Tabel 3. 6 Struktur Data Variabel *Covariate* Setelah Proses Diskretisasi pada *Dataset* BCI

Grid i	Grid j	$\mathbf{z}_{ij}^{(1)}$...	$\mathbf{z}_{ij}^{(p)}$
1	1	$\mathbf{z}_{1,1}^{(1)}$...	$\mathbf{z}_{1,1}^{(p)}$
1	2	$\mathbf{z}_{1,2}^{(1)}$...	$\mathbf{z}_{1,2}^{(p)}$
...
1	d_2	$\mathbf{z}_{1,d_2}^{(1)}$...	$\mathbf{z}_{1,d_2}^{(p)}$
...
49	1	$\mathbf{z}_{49,1}^{(1)}$...	$\mathbf{z}_{49,1}^{(p)}$
49	2	$\mathbf{z}_{49,2}^{(1)}$...	$\mathbf{z}_{49,2}^{(p)}$
...
d_1	d_2	$\mathbf{z}_{d_1,d_2}^{(1)}$...	$\mathbf{z}_{d_1,d_2}^{(p)}$

3.3.3 Proses Pelatihan

Langkah pertama setelah didapatkan *point pattern* \mathcal{D}_{bci} , variabel *covariate* \mathcal{Z}_{bci} , dan variabel waktu \mathcal{T}_{bci} adalah diskretisasi. Dengan demikian, model menerima data input $\mathcal{X} = \{\tilde{\mathcal{D}}_{bci}, \tilde{\mathcal{Z}}_{bci}, \mathcal{T}_{bci}\}$ dan mengeluarkan estimasi parameter $\hat{\theta}_{bci}$ pada *hidden layer*. Selanjutnya, dilakukan rekonstruksi menggunakan persamaan (3.8) untuk mendapatkan nilai *output* akhir $\hat{\Lambda}_{ij}^{(s,t)}$.

Sebelum pelatihan *neural network* dimulai, ketiga data input $\tilde{\mathcal{D}}_{bci}$, $\tilde{\mathcal{Z}}_{bci}$, dan \mathcal{T}_{bci} dibagi menjadi data latih dan data validasi sehingga didapatkan $\tilde{\mathcal{D}}_{bci,latih}$, $\tilde{\mathcal{D}}_{bci,val}$, $\tilde{\mathcal{Z}}_{bci,latih}$, $\tilde{\mathcal{Z}}_{bci,val}$, $\mathcal{T}_{bci,latih}$, dan $\mathcal{T}_{bci,val}$. Selanjutnya, $\tilde{\mathcal{D}}_{bci,latih}$, $\tilde{\mathcal{Z}}_{bci,latih}$, dan $\mathcal{T}_{bci,latih}$ dipakai untuk proses pelatihan *neural network* sebanyak 50 kali perulangan atau *epoch*. Kebaikan model *deep learning* dihitung menggunakan metode MSE berdasarkan $\tilde{\mathcal{D}}_{bci,latih}$, $\tilde{\mathcal{D}}_{bci,val}$, dan $\tilde{\mathcal{D}}_{bci}$. Ilustrasi model *probabilistic deep learning* pada studi terapan digambarkan pada Gambar 3.1.

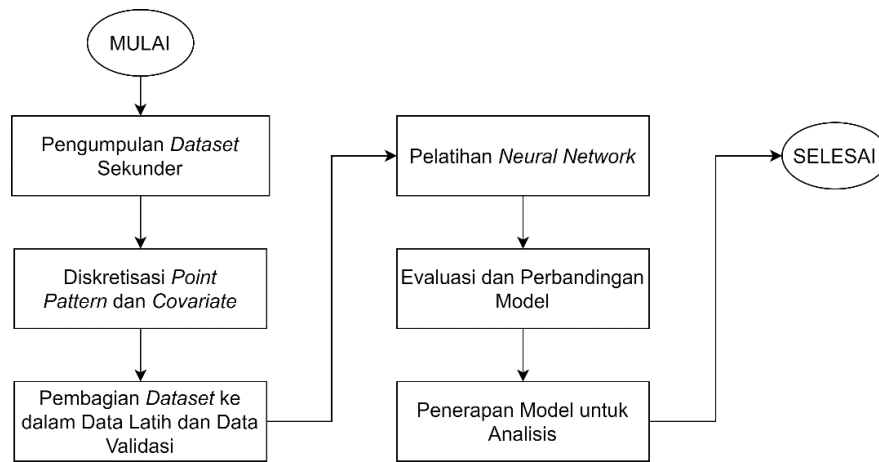
3.3.4 Interpretasi Model

Model *probabilistic deep learning* yang telah dilatih kemudian digunakan untuk estimasi parameter $\hat{\theta}_{bci}$ pada keseluruhan dataset \mathcal{X} pada kedua jenis

analisis, yaitu (1) analisis terhadap 25 spesies dengan 4 waktu observasi, dan (2) analisis terhadap 100 spesies dengan 8 waktu observasi. Hasil estimasi $\hat{\theta}_{bci}$ kemudian diolah lebih lanjut untuk mengetahui pola persebaran pohon pada BCI dataset, yaitu:

- 1) Interpretasi parameter $\hat{\Lambda}$, $\hat{\mu}$, $\hat{\alpha}$, $\hat{\sigma}$, dan *proportion of variance* (PV).
- 2) Korelasi Antar Spesies, dibahas lebih lanjut pada subbab 2.3.2.

3.3.5 Tahapan Analisis pada Studi Terapan



Gambar 3. 8 Diagram Alir Tahapan Analisis pada Studi Terapan

Diagram alir tahapan analisis pada studi simulasi ditunjukkan oleh Gambar 3.8. Langkah-langkah analisis yang dilakukan pada studi terapan adalah:

1. Pengumpulan Data. *Dataset* BCI didapatkan secara bebas dan *open source* (Condit dkk, 2019).
2. Diskretisasi data *point pattern* dan *covariate* sesuai pembahasan pada 3.1.1.
3. Pembagian *dataset* ke dalam data latih dan data validasi secara acak seperti pembahasan pada 3.1.3.
4. Pelatihan *neural network* seperti yang dibahas pada 3.3.3.
5. Evaluasi kebaikan model menggunakan MSE pada persamaan (3.13) hingga persamaan (3.15). Nilai MSE untuk $\hat{\Lambda}_{ij}^{(s,t)}$ yang kecil menandakan bahwa model memberikan estimasi yang baik.
6. Penerapan model *probabilistic deep learning* untuk menganalisis persebaran pohon pada kasus: (1) 25 spesies dengan 4 waktu observasi, dan (2) 100 spesies dengan 8 waktu observasi.

(Halaman ini sengaja dikosongkan)

BAB 4

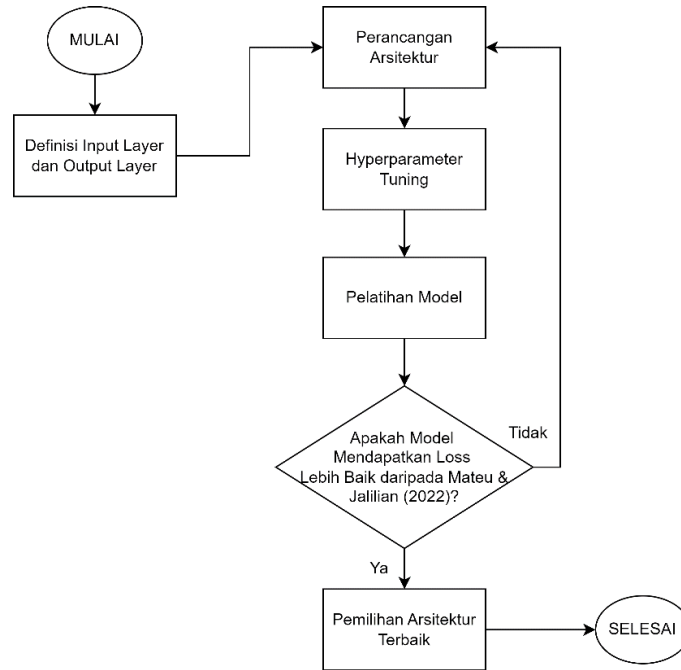
HASIL DAN PEMBAHASAN

4.1 Arsitektur Model

Dataset untuk studi terapan memiliki unsur temporal dan variabel *covariate* sedangkan *dataset* untuk studi simulasi tidak memiliki unsur tersebut. Studi terapan juga memiliki jumlah parameter lebih banyak daripada studi simulasi karena memuat unsur *temporal*. Oleh sebab itu, terdapat dua model *probabilistic deep learning* yang dipakai untuk masing-masing studi. Kedua model tersebut memiliki arsitektur yang mirip namun memiliki tambahan *input layer* dan *hidden layer* untuk mengakomodasi input dan parameter yang lebih banyak pada studi terapan. Kedua model juga memiliki jumlah *neuron* yang berbeda karena *dataset* pada studi simulasi hanya terdiri dari 10 spesies tanpa unsur temporal. Meskipun demikian, kedua model tetap menggunakan konfigurasi pelatihan yang serupa. Jumlah *epoch* adalah 150 untuk studi simulasi dan 50 untuk studi terapan. Fungsi optimasi yang digunakan adalah RMSProp (Hinton, 2012) dengan *learning rate* 0.0015. Kedua model *probabilistic deep learning* dilatih menggunakan MSE sebagai fungsi *loss*.

Langkah pemilihan arsitektur model ditunjukkan pada Gambar 4.1. Pemilihan arsitektur model dimulai dengan pendefinisian *input layer* untuk menerima data latih \mathcal{X} dan *output layer* yang mengeluarkan estimasi $\hat{\Lambda}_{ij}^{(s)}$ atau $\hat{\Lambda}_{ij}^{(s,t)}$. Arsitektur model diinisialisasi secara sederhana terlebih dahulu dengan beberapa *hidden layer* dan jumlah *neuron* yang sedikit. *Hyperparameter tuning* dilakukan sebelum pelatihan model. *Hyperparameter tuning* adalah pemilihan nilai *hyperparameter* model, seperti jumlah *epoch*, fungsi aktivasi, dan *learning rate* yang menghasilkan model dengan *loss* terkecil. Kemudian, model dilatih dan dicatat nilai *loss*-nya. Apabila nilai *loss* model lebih buruk daripada model Mateu & Jalilian (2022), maka dilakukan penambahan *hidden layer* dan *neuron* untuk menambah kompleksitas model. Proses ini diulang berkali-kali hingga didapatkan model paling optimal dengan nilai *loss* yang lebih kecil dari model Mateu & Jalilian (2022). Langkah pemilihan arsitektur seperti ini menghasilkan model *deep learning*

dengan kompleksitas sesederhana mungkin namun tetap mendapatkan performa yang baik, atau disebut juga sebagai model parsimoni (Goodfellow dkk, 2016).



Gambar 4. 1 Langkah Pemilihan Arsitektur Model

4.1.1 Model untuk Studi Simulasi

Model *probabilistic deep learning* untuk studi simulasi menerima data input berupa point pattern yang telah didiskretisasi $\tilde{\mathcal{D}}_{sim}$, jarak Euclid antar pasangan titik yang dinotasikan sebagai $\mathbf{d}(u, u')$, dan vektor kolom dengan semua elemen bernilai 1. Model memberikan estimasi $\hat{\theta}_{sim} = \{\hat{\mu}, \hat{\alpha}, \hat{\sigma}, \hat{\phi}, \hat{\psi}\}$ pada *hidden layer* dan dilakukan rekonstruksi $\hat{\Lambda}_{ij}^{(s)}$ berdasarkan persamaan (3.7).

Jumlah tipe *multivariate* dan *latent field* pada studi simulasi secara berturut-turut adalah $m = 10$ dan $a = 4$ (Choiruddin dkk, 2020). Ukuran *batch* pada model untuk studi simulasi adalah $b' = 50$. Lampiran 1 menunjukkan arsitektur model pada studi simulasi berdasarkan desain arsitektur pada Gambar 3.4. Secara umum, arsitektur model untuk studi simulasi merupakan versi yang lebih sederhana daripada model untuk studi terapan karena studi simulasi tidak mengikutsertakan unsur *temporal* dan variabel *covariate* (bersifat stasioner).

Model untuk studi simulasi dilatih sebanyak 150 perulangan *epoch* dan menggunakan konfigurasi yang serupa seperti model untuk studi terapan. Selain itu, model untuk studi simulasi juga memiliki jumlah *neuron* yang lebih sedikit karena jumlah tipe *multivariate* dan *latent field* yang lebih sedikit daripada studi terapan. Penjelasan terkait struktur dan persamaan *hidden layer* pada model untuk studi simulasi dijelaskan lebih lanjut pada Lampiran 1.

4.1.2 Model untuk Studi Terapan

Studi terapan dibagi menjadi 2 bagian, yaitu penerapan model *probabilistic deep learning* pada (1) 25 spesies dengan 4 repetisi waktu, dan (2) 100 spesies dengan 8 repetisi waktu. Jumlah *latent field* yang dipakai adalah $a = 5$ (Mateu & Jalilian, 2022). Model menerima data input berupa *point pattern* yang telah didiskretisasi $\tilde{\mathcal{D}}_{bci}$, *covariate* yang telah didiskretisasi $\tilde{\mathcal{Z}}_{bci}$, variabel \mathcal{J}_{bci} yang memuat informasi waktu terjadinya sensus, jarak Euclid antar pasangan titik yang dinotasikan sebagai $\mathbf{d}(u, u')$, jarak antar waktu sensus $\mathbf{d}(t, t')$, dan matriks \mathbf{D} yang menyimpan variabel *dummy* untuk \mathcal{J}_{bci} . Model untuk studi terapan memberikan estimasi $\hat{\boldsymbol{\theta}}_{bci} = \{\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\rho}}, \hat{\boldsymbol{\zeta}}\}$ melalui *hidden layer* dan merekonstruksi $\hat{\Lambda}_{ij}^{(s,t)}$ menggunakan persamaan (3.8). Model pada studi terapan menggunakan ukuran *batch* $b' = 200$. Tabel 4.1 menunjukkan arsitektur model pada studi terapan berdasarkan desain arsitektur pada Gambar 3.1.

Tabel 4. 1 Arsitektur Model pada Studi Terapan

No.	Jenis Layer	Nama Layer	Dimensi Output	Fungsi Aktivasi	Layer Sebelumnya
1	Input	xi	m	-	-
2	Input	zi	$p = 13$	-	-
3	Input	ui	2	-	-
4	Input	ti	1	-	-
5	Input	d _{ti}	T	-	-
6	Custom	udist	200 × 200	-	ui
7	Custom	tdist	200 × 200	-	ti
8	Dense	h1	n_{node}	tanh	xi

Tabel 4. 1 Lanjutan Arsitektur Model pada Studi Terapan

No.	Jenis Layer	Nama Layer	Dimensi Output	Fungsi Aktivasi	Layer Sebelumnya
9	Dense	h2	128	tanh	zi
10	Dense	h3	n_{node}	tanh	udist
11	Dense	h_udist	m	tanh	h3
12	Dense	h_udist2	$a = 5$	tanh	h3
13	Dense	ht	128	tanh	tdist
14	Dense	h_tdist	m	tanh	ht
15	Dense	h_tdist2	a	tanh	ht
16	Concatenate	h_concat_1	n_{node} + 128	-	h1, h2
17	Dense	h4	n_{node} × 2	relu	h_concat_1
18	Dropout	dr1	n_{node} × 2	-	h4
19	Dense	h5	n_{node} × 2	relu	h_concat_1
20	Dropout	dr2	n_{node} × 2	-	h5
21	Dense	mu_branch	n_{node}	tanh	dr1
22	Dense	par_branch	n_{node}	tanh	dr2
23	Dropout	dr_par_branch	n_{node}	-	par_branch
24	Dense	par_branch_2	n_{node}	tanh	dr2
25	Dropout	dr_par_branch_2	n_{node}	-	par_branch_2
26	Dense	mu_params	m	exp	mu_branch
27	Probabilistic Layer	mu_dist	m	-	mu_params
28	Dense	sphi_params	$a \times 2$	sigmoid, softplus	dr_par_branch
29	Probabilistic Layer	sphi_dist	a	-	sphi_params
30	Dense	salf_params	$a \times 2$	sigmoid, softplus	dr_par_branch
31	Probabilistic Layer	salf_dist	a	-	salf_params
32	Dense	tphi_params	$a \times 2$	tanh, softplus	dr_par_branch

Tabel 4. 1 Lanjutan Arsitektur Model pada Studi Terapan

No.	Jenis Layer	Nama Layer	Dimensi Output	Fungsi Aktivasi	Layer Sebelumnya
33	Probabilistic Layer	tphi_dist	a	-	tphi_params
34	Dense	spsi_params	$m \times 2$	sigmoid, softplus	dr_par_branch_2
35	Probabilistic Layer	spsi_dist	m	-	spsi_params
36	Dense	tpsi_params	$m \times 2$	tanh, softplus	dr_par_branch_2
37	Probabilistic Layer	tpsi_dist	m	-	tpsi_params
38	Multiply	pre_Emat	a	-	h_udist2, h_tdist2, sphi_dist, salf_dist, tphi_dist
39	Dense	Emat	$a \times 200 \times 200$	-	pre_Emat
40	Multiply	pre_Vmat	m	-	h_udist, h_tdist, spsi_dist, tpsi_dist
41	Dense	Vmat	$m \times 200 \times 200$	-	pre_Vmat
42	Probabilistic Layer	e_dist	a	-	Emat
43	Dense	Elin	m	-	e_dist
44	Probabilistic Layer	v_dist	m	-	Vmat
45	Dense	sigma_h	m	-	dti
46	Multiply	sigma_v_hat	m	-	sigma_h, v_dist
47	Add	lin	m	-	Elin, sigma_v_hat
48	Custom	exp_lin	m	exp	lin
49	Multiply	xihat	m	-	mu_dist, exp_lin

Secara umum, arsitektur model *probabilistic deep learning* pada studi terapan memiliki struktur yang serupa dengan arsitektur pada studi simulasi.

Perbedaannya, selain menerima input *point pattern* yang telah didiskretisasi, model pada studi terapan ini juga menerima input tambahan berupa variabel waktu \mathcal{J}_{bci} dan variabel *covariate* $\tilde{\mathcal{Z}}_{bci}$. Selain itu, penambahan unsur *temporal* mengakibatkan bertambahnya jumlah parameter yang harus diestimasi, yaitu $\theta_{bci} = (\mu, \alpha, \sigma, \psi, \phi, \varphi, \rho, \zeta)^T$. Perbedaan lainnya terletak pada jumlah *neuron* di beberapa *hidden layer* yang dinotasikan sebagai n_{node} dengan persamaan:

$$n_{node} = \min((m \times 5), 384) \quad (4.1)$$

Pada persamaan (4.1), n_{node} melambangkan jumlah *neuron* yang dipengaruhi oleh jumlah *species* m yang dianalisis. Semakin besar jumlah spesies, maka nilai n_{node} akan semakin besar, namun nilai n_{node} tidak lebih dari 384 untuk mengurangi resiko *overfitting* akibat model yang terlalu kompleks.

Sesuai dengan Tabel 4.1, model *deep learning* memberikan *output* berupa $\hat{\Lambda}_{ij}^{(s,t)}$ yang dikeluarkan oleh *layer* “xihat” (baris 49 pada Tabel 4.1). Variabel $\hat{\Lambda}_{ij}^{(s,t)}$ merupakan estimasi jumlah titik pada *grid* B_{ij} dengan tipe *multivariate* ke- s dan waktu ke- t . Penghitungan jarak Euclid antar titik $\mathbf{d}(u, u')$ diimplementasi pada *layer* “udist” (baris 6 pada Tabel 4.1). Jarak antar titik $\mathbf{d}(u, u')$ direpresentasikan dalam sebuah *hidden layer* dengan persamaan:

$$\begin{aligned} \hat{\mathbf{d}}(u, u')^{(l,t)} &= f_{\tanh}(\mathbf{b}_{11} + \mathbf{w}_{11}^T \times \mathbf{d}(u, u')) \\ \hat{\mathbf{d}}(u, u')^{(s,t)} &= f_{\tanh}(\mathbf{b}_{12} + \mathbf{w}_{12}^T \times \mathbf{d}(u, u')) \end{aligned} \quad (4.2)$$

dimana \mathbf{w}_{11} dan \mathbf{w}_{12} merupakan matriks bobot berukuran $200 \times a$ dan $200 \times m$, \mathbf{b}_{11} dan \mathbf{b}_{12} merupakan vektor bias berukuran a dan m , sedangkan $\mathbf{d}(u, u')$ adalah jarak Euclid antar titik berukuran 200×200 . Sementara itu, penghitungan jarak antar waktu $\mathbf{d}(t, t')$ dilakukan pada *layer* “tdist” (baris 7 pada Tabel 4.1) menggunakan persamaan:

$$\mathbf{d}(t, t') = |t - t'| \quad (4.3)$$

Jarak antar waktu juga direpresentasikan dalam sebuah *hidden layer* dengan persamaan:

$$\begin{aligned}\widehat{\mathbf{d}}(t, t')^{(l,t)} &= f_{\tanh}(\mathbf{b}_{14} + \mathbf{w}_{14}^T \times \mathbf{d}(t, t')) \\ \widehat{\mathbf{d}}(t, t')^{(s,t)} &= f_{\tanh}(\mathbf{b}_{15} + \mathbf{w}_{15}^T \times \mathbf{d}(t, t'))\end{aligned}\quad (4.4)$$

dimana \mathbf{w}_{14} dan \mathbf{w}_{15} merupakan matriks bobot berukuran $200 \times a$ dan $200 \times m$, \mathbf{b}_{14} dan \mathbf{b}_{15} merupakan vektor bias berukuran a dan m , sedangkan $\mathbf{d}(t, t')$ adalah jarak antar waktu berukuran 200×200 . Parameter $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\psi}$, dan $\boldsymbol{\phi}$ diimplementasi dengan metode yang serupa dengan studi simulasi namun dengan penambahan unsur *temporal*. *Layer* “mu_params” (baris 26 pada Tabel 4.1) memberikan estimasi $\hat{\lambda}_{ij}^{(s,t)}$ yang memiliki fungsi aktivasi eksponensial sehingga nilai estimasi parameter tersebut selalu positif. Nilai estimasi tersebut dipakai sebagai parameter untuk mendapatkan $\hat{\boldsymbol{\mu}} = [\hat{\mu}_{ij}^{(s,t)}]$ yang diasumsikan berdistribusi Poisson dengan ekspektasi $\hat{\boldsymbol{\lambda}} = [\hat{\lambda}_{ij}^{(s,t)}]$ (lihat Persamaan (3.10)). Proses mendapatkan estimasi $\hat{\boldsymbol{\lambda}}$ ditunjukkan pada persamaan:

$$\hat{\boldsymbol{\lambda}} = f_{\exp}(\mathbf{b}_{26} + \mathbf{w}_{26}^T \times \hat{\mathbf{h}}^{(21)}) \quad (4.5)$$

dimana $\mathcal{L}_{21} = n_{node}$ merupakan jumlah *neuron* pada *layer* ke-21 (lihat baris 21 kolom 4 pada Tabel 4.1), \mathbf{b}_{26} adalah vektor bias berukuran m , $\hat{\mathbf{h}}^{(21)}$ adalah matriks keluaran dari *layer* ke-21 berukuran $\mathcal{L}_{21} \times 200$, dan \mathbf{w}_{26} adalah matriks bobot berukuran $\mathcal{L}_{21} \times m$. Nilai $\hat{\mu}_{ij}^{(s)}$ diasumsikan merupakan ekspektasi dari distribusi Poisson, sehingga:

$$\mathbb{E}(\hat{\mu}_{ij}^{(s,t)}) = \hat{\lambda}_{ij}^{(s,t)} \quad (4.6)$$

Parameter $\boldsymbol{\phi} = [\phi_{ij}^{(l,t)}]$, $\boldsymbol{\varphi} = [\varphi_{ij}^{(l,t)}]$ dan $\boldsymbol{\rho} = [\rho_{ij}^{(l,t)}]$ dimodelkan pada *layer* “sphi_dist”, “salf_dist” dan “tphi_dist” (baris 29, 31, dan 33 pada Tabel 4.1) yang diasumsikan mengikuti distribusi Normal dengan parameter:

$$\boldsymbol{\mu}_{\phi} = [\mu_{\phi_{ij}}^{(l,t)}], (\boldsymbol{\sigma}_{\phi})^2 = [(\sigma_{\phi_{ij}}^{(l,t)})^2]$$

$$\boldsymbol{\mu}_{\varphi} = [\mu_{\varphi_{ij}}^{(l,t)}], (\boldsymbol{\sigma}_{\varphi})^2 = [(\sigma_{\varphi_{ij}}^{(l,t)})^2]$$

$$\boldsymbol{\mu}_\rho = \left[\mu_{\rho_{ij}}^{(l,t)} \right], (\boldsymbol{\sigma}_\rho)^2 = \left[\left(\sigma_{\rho_{ij}}^{(l,t)} \right)^2 \right]$$

Layer “sphi_params”, “salf_params” dan “tphi_params” (baris 28, 30, dan 32 pada Tabel 4.1) memiliki dimensi keluaran $a \times 2$ yang mencerminkan nilai estimasi parameter untuk setiap distribusi Normal. Nilai estimasi $\hat{\boldsymbol{\mu}}_\phi$ dan $(\hat{\boldsymbol{\sigma}}_\phi)^2$ didapatkan melalui persamaan:

$$\hat{\boldsymbol{\mu}}_\phi = f_{\text{sig}} \left(\mathbf{b}_{28}^{(1)} + \left(\mathbf{w}_{28}^{(1)} \right)^T \times \hat{\mathbf{h}}^{(23)} \right) \quad (4.7)$$

$$(\hat{\boldsymbol{\sigma}}_\phi)^2 = f_{\text{splus}} \left(\mathbf{b}_{28}^{(2)} + \left(\mathbf{w}_{28}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(23)} \right) \quad (4.8)$$

dimana $\mathcal{L}_{23} = n_{\text{node}}$ merupakan jumlah *neuron* pada *layer* ke-23 (lihat baris 23 kolom 4 pada Tabel 4.1), $\mathbf{b}_{28}^{(1)}$ dan $\mathbf{b}_{28}^{(2)}$ adalah vektor bias berukuran a , $\hat{\mathbf{h}}^{(23)}$ adalah matriks keluaran dari *layer* ke-23 berukuran $\mathcal{L}_{23} \times 200$, sedangkan $\mathbf{w}_{28}^{(1)}$ dan $\mathbf{w}_{28}^{(2)}$ adalah matriks bobot berukuran $\mathcal{L}_{23} \times a$. Dengan demikian, $\hat{\boldsymbol{\mu}}_\phi$ dan $(\hat{\boldsymbol{\sigma}}_\phi)^2$ adalah matriks berukuran $a \times 200$. Sementara itu, nilai estimasi $\hat{\boldsymbol{\mu}}_\phi$ dan $(\hat{\boldsymbol{\sigma}}_\phi)^2$ didapatkan melalui persamaan:

$$\hat{\boldsymbol{\mu}}_\phi = f_{\text{sig}} \left(\mathbf{b}_{30}^{(1)} + \left(\mathbf{w}_{30}^{(1)} \right)^T \times \hat{\mathbf{h}}^{(23)} \right) \quad (4.9)$$

$$(\hat{\boldsymbol{\sigma}}_\phi)^2 = f_{\text{splus}} \left(\mathbf{b}_{30}^{(2)} + \left(\mathbf{w}_{30}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(23)} \right) \quad (4.10)$$

Serupa dengan parameter sebelumnya, nilai estimasi $\hat{\boldsymbol{\mu}}_\rho$ dan $(\hat{\boldsymbol{\sigma}}_\rho)^2$ didapatkan melalui persamaan:

$$\hat{\boldsymbol{\mu}}_\rho = f_{\text{tanh}} \left(\mathbf{b}_{32}^{(1)} + \left(\mathbf{w}_{32}^{(1)} \right)^T \times \hat{\mathbf{h}}^{(23)} \right) \quad (4.11)$$

$$(\hat{\boldsymbol{\sigma}}_\rho)^2 = f_{\text{splus}} \left(\mathbf{b}_{32}^{(2)} + \left(\mathbf{w}_{32}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(23)} \right) \quad (4.12)$$

dimana $\mathcal{L}_{23} = n_{\text{node}}$ merupakan jumlah *neuron* pada *layer* ke-23 (lihat baris 23 kolom 4 pada Tabel 4.1), $\mathbf{b}_{32}^{(1)}$ dan $\mathbf{b}_{32}^{(2)}$ adalah vektor bias berukuran a , $\hat{\mathbf{h}}^{(23)}$ adalah matriks keluaran dari *layer* ke-23 berukuran $\mathcal{L}_{23} \times 200$, dan $\mathbf{w}_{32}^{(1)}$ dan $\mathbf{w}_{32}^{(2)}$ adalah

matriks bobot berukuran $\mathcal{L}_{23} \times a$. Dengan demikian, $\hat{\boldsymbol{\mu}}_\rho$ dan $(\hat{\boldsymbol{\sigma}}_\rho)^2$ adalah matriks berukuran $a \times 200$.

Pada *probabilistic deep learning*, proses *sampling* untuk mendapatkan estimasi parameter $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\varphi}}$, dan $\hat{\boldsymbol{\rho}}$ dilakukan melalui *reparameterization trick* (Kingma & Welling, 2014):

$$\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\mu}}_\phi + \hat{\boldsymbol{\sigma}}_\phi \odot \varepsilon_\phi \quad (4.13)$$

$$\hat{\boldsymbol{\varphi}} = \hat{\boldsymbol{\mu}}_\varphi + \hat{\boldsymbol{\sigma}}_\varphi \odot \varepsilon_\varphi \quad (4.14)$$

$$\hat{\boldsymbol{\rho}} = \hat{\boldsymbol{\mu}}_\rho + \hat{\boldsymbol{\sigma}}_\rho \odot \varepsilon_\rho \quad (4.15)$$

dimana \odot merupakan operasi perkalian elemen demi elemen (*element-wise multiplication*), sedangkan $\varepsilon_\phi \sim \text{Normal}(0,1)$, $\varepsilon_\varphi \sim \text{Normal}(0,1)$, dan $\varepsilon_\rho \sim \text{Normal}(0,1)$.

Proses serupa juga diterapkan terhadap parameter $\boldsymbol{\psi} = [\psi_{ij}^{(s,t)}]$ dan $\boldsymbol{\zeta} = [\zeta_{ij}^{(s,t)}]$ yang diestimasi pada *layer* “spsi_dist” dan “tpsi_dist” (baris 35 dan 37 pada Tabel 4.1). Kedua parameter tersebut diasumsikan mengikuti distribusi Normal dengan parameter:

$$\boldsymbol{\mu}_\psi = [\mu_{\psi_{ij}}^{(s,t)}], (\boldsymbol{\sigma}_\psi)^2 = [(\sigma_{\psi_{ij}}^{(s,t)})^2]$$

$$\boldsymbol{\mu}_\zeta = [\mu_{\zeta_{ij}}^{(s,t)}], (\boldsymbol{\sigma}_\zeta)^2 = [(\sigma_{\zeta_{ij}}^{(s,t)})^2]$$

Layer “spsi_params” dan “tpsi_params” (baris 34 dan 36 pada Tabel 4.1) memiliki dimensi keluaran $m \times 2$ untuk membentuk m distribusi normal. Proses estimasi $\hat{\boldsymbol{\mu}}_\psi$ dan $(\hat{\boldsymbol{\sigma}}_\psi)^2$ ditunjukkan pada persamaan:

$$\hat{\boldsymbol{\mu}}_\psi = f_{\text{sig}} \left(\mathbf{b}_{34}^{(1)} + \left(\mathbf{w}_{34}^{(1)} \right)^T \times \hat{\mathbf{h}}^{(25)} \right) \quad (4.16)$$

$$(\hat{\boldsymbol{\sigma}}_\psi)^2 = f_{\text{splus}} \left(\mathbf{b}_{34}^{(2)} + \left(\mathbf{w}_{34}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(25)} \right) \quad (4.17)$$

dimana $\mathcal{L}_{25} = n_{node}$ merupakan jumlah *neuron* pada *layer* ke-25 (lihat baris 25 kolom 4 pada Tabel 4.1), $\mathbf{b}_{34}^{(1)}$ dan $\mathbf{b}_{34}^{(2)}$ merupakan vektor bias berukuran m , $\hat{\mathbf{h}}^{(25)}$ adalah matriks keluaran dari *layer* ke-25 berukuran $\mathcal{L}_{25} \times 200$, sedangkan $\mathbf{w}_{34}^{(1)}$ dan $\mathbf{w}_{34}^{(2)}$ adalah matriks bobot berukuran $\mathcal{L}_{25} \times m$. Dengan demikian, $\hat{\boldsymbol{\mu}}_{\psi}$ dan $(\hat{\boldsymbol{\sigma}}_{\psi})^2$ adalah matriks berukuran $m \times 200$. Serupa dengan sebelumnya, nilai estimasi $\hat{\boldsymbol{\mu}}_{\zeta}$ dan $(\hat{\boldsymbol{\sigma}}_{\zeta})^2$ didapatkan melalui persamaan:

$$\hat{\boldsymbol{\mu}}_{\zeta} = f_{\tanh} \left(\mathbf{b}_{36}^{(1)} + \left(\mathbf{w}_{36}^{(1)} \right)^T \times \hat{\mathbf{h}}^{(25)} \right) \quad (4.18)$$

$$(\hat{\boldsymbol{\sigma}}_{\zeta})^2 = f_{\text{splus}} \left(\mathbf{b}_{36}^{(2)} + \left(\mathbf{w}_{36}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(25)} \right) \quad (4.19)$$

dimana $\mathcal{L}_{25} = n_{node}$ merupakan jumlah *neuron* pada *layer* ke-25 (lihat baris 25 kolom 4 pada Tabel 4.1), $\mathbf{b}_{36}^{(1)}$ dan $\mathbf{b}_{36}^{(2)}$ merupakan vektor bias berukuran m , $\hat{\mathbf{h}}^{(25)}$ adalah matriks keluaran dari *layer* ke-25 berukuran $\mathcal{L}_{25} \times 200$, sedangkan $\mathbf{w}_{36}^{(1)}$ dan $\mathbf{w}_{36}^{(2)}$ adalah matriks bobot berukuran $\mathcal{L}_{25} \times m$. Dengan demikian, $\hat{\boldsymbol{\mu}}_{\zeta}$ dan $(\hat{\boldsymbol{\sigma}}_{\zeta})^2$ adalah matriks berukuran $m \times 200$.

Setelah itu, dilakukan proses *sampling* menggunakan *reparameterization trick* untuk mendapatkan hasil estimasi $\hat{\boldsymbol{\psi}}$ dan $\hat{\boldsymbol{\zeta}}$ seperti yang ditunjukkan pada persamaan:

$$\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{\mu}}_{\psi} + \hat{\boldsymbol{\sigma}}_{\psi} \odot \boldsymbol{\varepsilon}_{\psi} \quad (4.20)$$

$$\hat{\boldsymbol{\zeta}} = \hat{\boldsymbol{\mu}}_{\zeta} + \hat{\boldsymbol{\sigma}}_{\zeta} \odot \boldsymbol{\varepsilon}_{\zeta} \quad (4.21)$$

dimana $\boldsymbol{\varepsilon}_{\psi} \sim \text{Normal}(0,1)$ dan $\boldsymbol{\varepsilon}_{\zeta} \sim \text{Normal}(0,1)$.

Pada penelitian Mateu & Jalilian (2022), Gaussian *random field* \mathbf{E} dan \mathbf{V} dibangkitkan berdasarkan distribusi *multivariate normal* berdimensi ukuran *batch* b' dengan *mean* $\mathbf{0}$ dan hasil Cholesky *decomposition* terhadap matriks *covariance* $\boldsymbol{\Sigma}_{\mathbf{E}}$ dan $\boldsymbol{\Sigma}_{\mathbf{V}}$, sehingga didapatkan matriks *lower triangular* \mathbf{A} dan \mathbf{B} sesuai dengan persamaan:

$$\Sigma_E = \mathbf{A}\mathbf{A}^T \quad (4.22)$$

$$\Sigma_V = \mathbf{B}\mathbf{B}^T \quad (4.23)$$

Estimasi matriks *lower triangular* \mathbf{A} dan \mathbf{B} diberikan oleh model *probabilistic deep learning* melalui *layer* “Emat” dan “Vmat” (baris 39 dan 41 pada Tabel 4.1). *Layer* “Emat” mengeluarkan estimasi matriks \mathbf{A} berdasarkan input dari *layer* “pre_Emat” (baris 38 pada Tabel 4.1) yang berisi informasi kalkulasi jarak Euclid antar titik $\widehat{\mathbf{d}}(u, u')^{(l,t)}$, kalkulasi jarak antar waktu $\widehat{\mathbf{d}}(t, t')^{(l,t)}$, estimasi $\widehat{\boldsymbol{\phi}}$, estimasi $\widehat{\boldsymbol{\psi}}$, dan estimasi $\widehat{\boldsymbol{\zeta}}$:

$$\begin{aligned} \widehat{\mathbf{h}}^{(39)} &= \left(\mathbf{w}_{39} (\widehat{\boldsymbol{\phi}} \times \widehat{\boldsymbol{\psi}} \times \widehat{\boldsymbol{\zeta}} \times \widehat{\mathbf{d}}(u, u')^{(l,t)} \times \widehat{\mathbf{d}}(t, t')^{(l,t)}) \right) \\ \widehat{\mathbf{h}}^{(39)} &= [\widehat{h}_{i'j'}^{(39)}] = \begin{cases} |\widehat{h}_{i'j'}^{(39)}| & \text{jika } i' = j' \\ \widehat{h}_{i'j'}^{(39)} & \text{jika } i' \neq j' \end{cases} \\ \mathbf{A} &= \begin{cases} \widehat{h}_{i'j'}^{(39)} & \text{jika } i' \geq j' \\ 0 & \text{jika } i' < j' \end{cases} \end{aligned} \quad (4.24)$$

dimana $i' = 1, 2, \dots, 200$, $j' = 1, 2, \dots, 200$, $\widehat{\mathbf{h}}^{(39)}$ merupakan matriks output *hidden layer* berukuran $a \times 200 \times 200$, \mathbf{w}_{39} merupakan matriks bobot berukuran $a \times 200 \times a$ pada *layer* ke-39, sedangkan $\widehat{\mathbf{d}}(u, u')^{(l,t)}$ dan $\widehat{\mathbf{d}}(t, t')^{(l,t)}$ adalah representasi jarak antar titik dan waktu (Persamaan (4.2) dan (4.4)). Semua elemen diagonal pada matriks $\widehat{\mathbf{h}}^{(39)}$ diubah ke bentuk positif dan semua elemen *upper diagonal* diubah ke 0 untuk mendapatkan estimasi *lower triangular* matriks \mathbf{A} . Proses estimasi matriks *lower triangular* \mathbf{B} dilakukan pada *layer* “Vmat” yang menerima input berupa informasi kalkulasi jarak Euclid antar titik $\mathbf{d}(u, u')$, kalkulasi jarak antar waktu $\mathbf{d}(t, t')$, estimasi $\widehat{\boldsymbol{\psi}}$, dan estimasi $\widehat{\boldsymbol{\zeta}}$:

$$\begin{aligned} \widehat{\mathbf{h}}^{(41)} &= \left(\mathbf{w}_{41} (\widehat{\boldsymbol{\psi}} \times \widehat{\boldsymbol{\zeta}} \times \widehat{\mathbf{d}}(u, u')^{(s,t)} \times \widehat{\mathbf{d}}(t, t')^{(s,t)}) \right) \\ \widehat{\mathbf{h}}^{(41)} &= [\widehat{h}_{i'j'}^{(41)}] = \begin{cases} |\widehat{h}_{i'j'}^{(41)}| & \text{jika } i' = j' \\ \widehat{h}_{i'j'}^{(41)} & \text{jika } i' \neq j' \end{cases} \\ \mathbf{B} &= \begin{cases} \widehat{h}_{i'j'}^{(41)} & \text{jika } i' \geq j' \\ 0 & \text{jika } i' < j' \end{cases} \end{aligned} \quad (4.25)$$

dimana $\hat{\mathbf{h}}^{(41)}$ merupakan matriks output *hidden layer* berukuran $m \times 200 \times 200$, \mathbf{w}_{41} merupakan matriks bobot berukuran $m \times 200 \times m$ pada layer ke-41, sedangkan $\hat{\mathbf{d}}(u, u')^{(s,t)}$ dan $\hat{\mathbf{d}}(t, t')^{(s,t)}$ adalah representasi dari jarak antar titik dan waktu (Persamaan (4.2) dan (4.4)). Semua elemen diagonal pada matriks $\hat{\mathbf{h}}^{(41)}$ diubah ke bentuk positif dan semua elemen *upper diagonal* diubah ke 0 untuk mendapatkan *lower triangular* matriks \mathbf{B} .

Seluruh variabel input pada persamaan (4.24) dan persamaan (4.25) merupakan variabel-variabel penyusun matriks *covariance* Σ_E dan Σ_V pada persamaan (2.20) dan persamaan (2.21). Nilai estimasi $\hat{\mathbf{E}}_{ij}^{(l,t)}$ dan $\hat{\mathbf{V}}_{ij}^{(s,t)}$ didapatkan melalui proses *sampling* menggunakan *reparameterization trick* pada kedua Gaussian *random field* \mathbf{E} dan \mathbf{V} seperti yang ditunjukkan pada persamaan:

$$\hat{\mathbf{E}}_{ij}^{(l,t)} = \mathbf{A}\boldsymbol{\varepsilon}_E \quad (4.26)$$

$$\hat{\mathbf{V}}_{ij}^{(s,t)} = \mathbf{B}\boldsymbol{\varepsilon}_V \quad (4.27)$$

dimana $\boldsymbol{\varepsilon}_E \sim \text{Multivariate Normal}(\mathbf{0}, \mathbf{I}_{200 \times 1})$ sedangkan $\boldsymbol{\varepsilon}_V \sim \text{Multivariate Normal}(\mathbf{0}, \mathbf{I}_{200 \times 1})$.

Persamaan *multivariate spatio-temporal* LGCP pada persamaan (3.8) mengandung komponen:

$$\sum_{l=1}^a \alpha_{sl} \mathbf{E}_{ij}^{(l,t)}$$

yang dapat direpresentasikan sebagai proses kalkulasi pada sebuah *hidden layer* tanpa fungsi aktivasi. Oleh sebab itu, nilai estimasi $\hat{\boldsymbol{\alpha}}$ dapat dimodelkan sebagai bobot pada *layer* “Elin” (baris 43 pada Tabel 4.1) yang menerima input $\hat{\mathbf{E}}_{ij}^{(l,t)}$ dari *layer* sebelumnya. Proses estimasi $\hat{\boldsymbol{\alpha}}$ ditunjukkan pada persamaan:

$$\hat{\boldsymbol{\alpha}} \hat{\mathbf{E}}_{ij}^{(s,t)} = \sum_{l=1}^a w_{43,l,s} \hat{\mathbf{E}}_{ij}^{(l,t)} \quad (4.28)$$

dimana \mathbf{w}_{43}^T merupakan matriks bobot berukuran $m \times a$ yang direpresentasikan sebagai hasil estimasi $\hat{\mathbf{a}}$.

Parameter $\hat{\sigma}^{(s,t)}$ tidak bergantung kepada *grid* B_{ij} . Oleh sebab itu, parameter tersebut dimodelkan sebagai bobot pada *layer* “sigma_h” yang tidak memiliki fungsi aktivasi. *Layer* “sigma_h” (baris 45 pada Tabel 4.1) menerima input data matriks \mathbf{D} , yaitu variabel *dummy* berdasarkan \mathcal{J}_{bci} , sehingga setiap spesies s memiliki nilai estimasi pada setiap waktu t . Tabel 4.2 menunjukkan contoh struktur data matriks \mathbf{D} pada *layer* “dti” apabila $T = 4$.

Tabel 4. 2 Contoh Struktur Data Variabel *Dummy*

t	Variabel <i>Dummy</i>			
1	1	0	0	0
...
3	0	0	1	0
...
4	0	0	0	1

Estimasi pada *layer* “sigma_h” ditunjukkan pada persamaan:

$$\hat{\sigma}^{(s,t)} = \sum_{t=1}^T w_{45,t,s} D^{(t)} \quad (4.29)$$

dimana $D^{(t)}$ merupakan nilai variabel *dummy* ke- t pada *layer* “dti”, dan \mathbf{w}_{45}^T merupakan matriks bobot berukuran $m \times T$. Sesuai dengan persamaan (3.8), hasil estimasi $\hat{\sigma}^{(s,t)}$ dikalikan dengan $\hat{\mathbf{V}}_{ij}^{(s,t)}$ seperti yang ditunjukkan pada persamaan:

$$\hat{\sigma} \hat{\mathbf{V}}_{ij}^{(s,t)} = \hat{\sigma}^{(s,t)} \times \hat{\mathbf{V}}_{ij}^{(s,t)} \quad (4.30)$$

Proses perkalian pada persamaan (4.30) terjadi pada *layer* “sigma_v_hat” (baris 46 pada Tabel 4.1). Selanjutnya, proses rekonstruksi sesuai dengan persamaan (3.8) dilakukan untuk mendapatkan hasil estimasi $\hat{\Lambda}_{ij}^{(s,t)}$:

$$\widehat{\Lambda}_{ij}^{(s,t)} = \widehat{\lambda}_{ij}^{(s,t)} f_{\text{exp}}\left(\widehat{\alpha}\widehat{E}_{ij}^{(s,t)} + \widehat{\sigma}\widehat{V}_{ij}^{(s,t)}\right) \quad (4.31)$$

Fungsi eksponensial telah diterapkan kepada $\widehat{\lambda}_{ij}^{(s,t)}$ pada persamaan (4.5), sehingga fungsi eksponensial tidak diterapkan kembali pada persamaan (4.31).

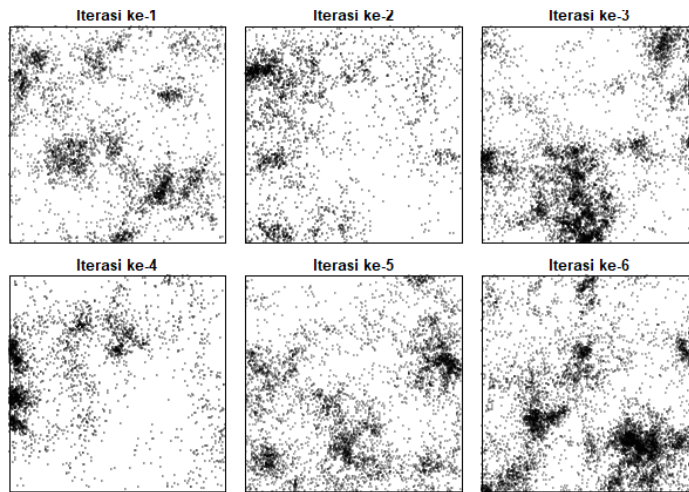
4.2 Studi Simulasi

Subbab 4.2.1 hingga 4.2.4 menjelaskan tentang studi simulasi untuk membandingkan performa model Choiruddin dkk (2020), Mateu & Jalilian (2022), dan model *probabilistic deep learning*.

4.2.1 Pembangkitan *Point Pattern*

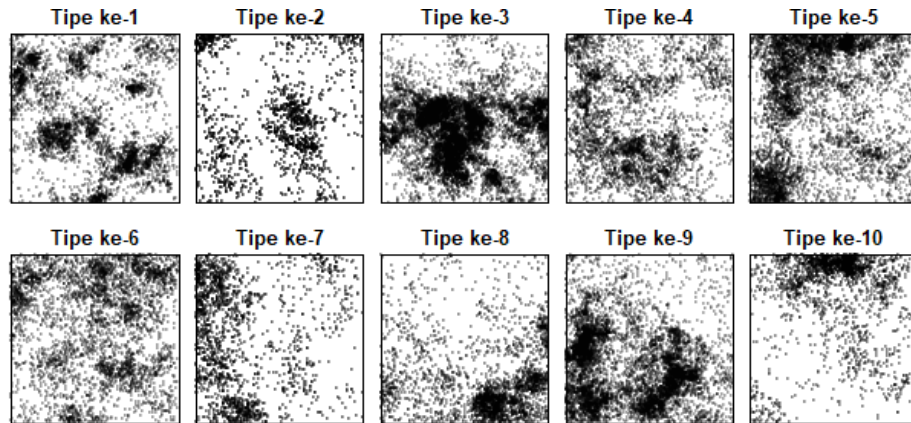
Point pattern dibangkitkan berdasarkan nilai parameter sebenarnya $\theta_{sim} = \{\mu, \alpha, \sigma, \phi, \psi\}$ pada persamaan (3.17) hingga persamaan (3.21). *Point pattern* bangkitan memiliki ukuran *window* W berukuran 1×1 , jumlah tipe *multivariate* $m = 10$, dan jumlah *latent field* $a = 4$. *Point pattern* bangkitan tidak memiliki unsur *temporal* dan variabel *covariate*.

Pembangkitan *point pattern* dilakukan sebanyak $n_{trial} = 200$ kali dengan *random seed* yang berbeda pada setiap proses pembangkitan. Nilai *random seed* berbeda menghasilkan realisasi *point pattern* yang selalu berbeda pula meskipun parameter θ_{sim} pada seluruh iterasi bernilai sama. Gambar 4.2 menunjukkan contoh 6 realisasi *point pattern* pada tipe *multivariate* pertama $s = 1$.



Gambar 4.2 Realisasi 6 *Point Pattern* Bangkitan pada Model *Multivariate* LGCP untuk Tipe Pertama

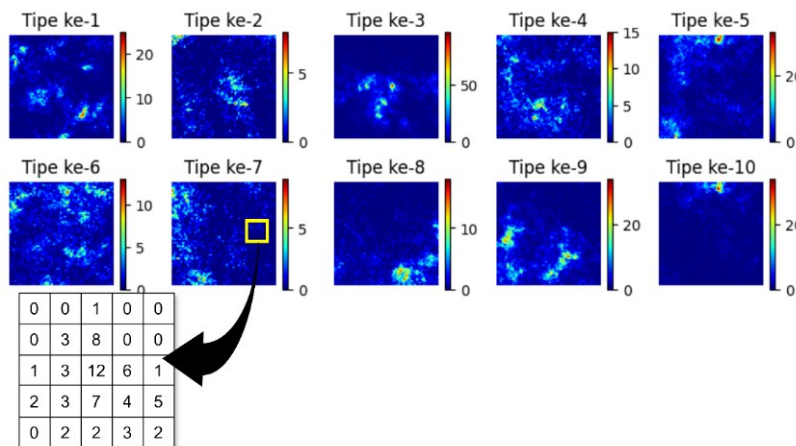
Seperti yang diperlihatkan pada Gambar 4.2, hasil realisasi *point pattern* selalu menunjukkan pola persebaran yang saling berbeda karena penggunaan nilai *random seed* berbeda pada setiap iterasi pembangkitan. Sementara itu, Gambar 4.3 menunjukkan realisasi *point pattern* pada tipe *multivariate* $s = 1$ hingga $s = 10$.



Gambar 4.3 Realisasi *Point Pattern* Bangkitan pada Model *Multivariate* LGCP untuk Semua Tipe

4.2.2 Diskretisasi *Point Pattern*

Diskretisasi merupakan proses pembagian *window* W menjadi *grid-grid* berukuran kecil yang dinotasikan sebagai B_{ij} , dimana $i = 1, 2, \dots, 60$ dan $j = 1, 2, \dots, 60$. Dengan demikian, sebuah *point pattern* dibagi menjadi 3600 *grid*. Selanjutnya, dilakukan penghitungan jumlah titik pada setiap *grid* B_{ij} yang dinotasikan sebagai $N(X^{(s)} \cap B_{ij})$ dan direpresentasikan sebagai $\Lambda_{ij}^{(s)}$. Proses diskretisasi dibahas secara mendalam pada subbab 3.1.1.



Gambar 4.4 Plot Hasil Diskretisasi pada Data Bangkitan

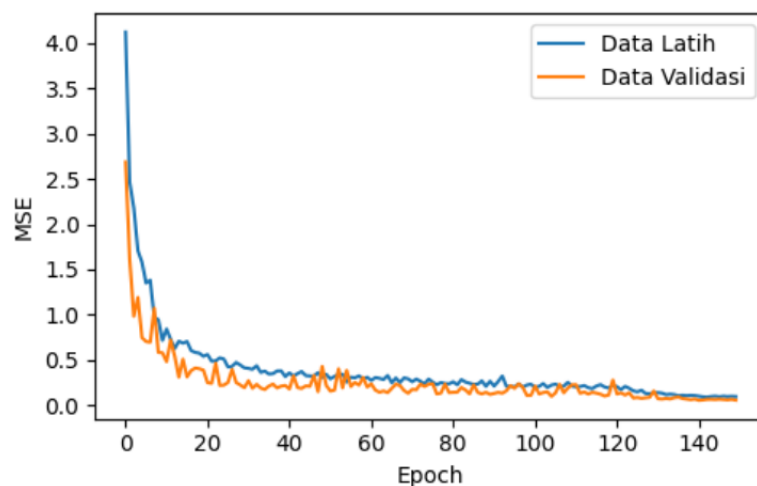
Hasil diskretisasi *point pattern* dinotasikan sebagai $\tilde{\mathcal{D}}_{sim}$ dan ditunjukkan pada Gambar 4.4. Setiap plot pada Gambar 4.4 dilengkapi dengan *heatmap* yang menunjukkan besaran jumlah titik pada setiap *grid* B_{ij} . Daerah berwarna terang menunjukkan jumlah titik yang banyak pada daerah tersebut

4.2.3 Pembagian Dataset dan Pelatihan Model

Pembagian *dataset* $\tilde{\mathcal{D}}_{sim}$ ke dalam data latih dan data validasi dilakukan sebelum proses pelatihan model *deep learning*. Pembagian *dataset* tersebut dilakukan sesuai dengan penjelasan pada subbab 3.1.3 sehingga didapatkan $\tilde{\mathcal{D}}_{sim_{latih}}$ dan $\tilde{\mathcal{D}}_{sim_{valid}}$.

Model *deep learning* dilatih menggunakan data $\tilde{\mathcal{D}}_{sim_{latih}}$ sebanyak 150 *epoch* sedangkan arsitektur modelnya ditunjukkan pada Lampiran 1. *Loss function* yang dipakai pada proses pelatihan adalah MSE. Komputer untuk pelatihan model *deep learning* dan estimasi Choiruddin dkk (2020) pada studi simulasi menggunakan *processor* Intel Core i7 @ 2.80GHz dengan *core* berjumlah 8, RAM berukuran 16GB, dan tidak ada dukungan GPU *accelerator*.

Ketika pelatihan berlangsung, model melakukan perubahan bobot berdasarkan data latih $\tilde{\mathcal{D}}_{sim_{latih}}$ dan perubahan tersebut dievaluasi menggunakan data validasi $\tilde{\mathcal{D}}_{sim_{valid}}$. Proses tersebut berlangsung berulang kali pada setiap *epoch* sehingga setiap *epoch* memiliki nilai MSE untuk data latih dan data validasi.



Gambar 4. 5 Grafik MSE Pelatihan Model pada Studi Simulasi

Gambar 4.5 menunjukkan grafik nilai MSE data latih dan data validasi untuk setiap *epoch* pada satu kali pelatihan model *deep learning*. Gambar 4.5 memperlihatkan bahwa grafik nilai MSE (*loss function*) antara data latih dan data validasi saling berimpit, sehingga dapat disimpulkan bahwa model tidak mengalami *overfitting*. Pelatihan model juga berjalan dengan baik karena nilai MSE pada data latih dan data validasi mengalami tren penurunan seiring dengan bertambahnya jumlah *epoch*.

Setelah pelatihan selesai dilakukan, maka model dipakai untuk melakukan prediksi pada dataset $\tilde{\mathcal{D}}_{sim_{latih}}$, $\tilde{\mathcal{D}}_{sim_{val}}$, dan $\tilde{\mathcal{D}}_{sim}$. Kemudian, nilai MSE pada ketiga prediksi tersebut dicatat dalam berkas “.csv”. Selain itu, dihitung pula nilai RMSE antara hasil estimasi $\hat{\theta}_{sim} = \{\hat{\mu}, \hat{\alpha}, \hat{\sigma}, \hat{\phi}, \hat{\psi}\}$ dan $\theta_{sim} = \{\mu, \alpha, \sigma, \phi, \psi\}$ untuk mengetahui kebaikan hasil estimasi model *probabilistic deep learning*. Proses pelatihan, estimasi, dan pencatatan tersebut diulang sebanyak 100 kali.

Proses pelatihan, estimasi, dan pencatatan tersebut juga diterapkan kepada model Mateu & Jalilian (2022) dan diulang sebanyak 100 kali percobaan. Selanjutnya, hasil pencatatan nilai MSE dan RMSE antara kedua metode dibandingkan untuk analisis lebih lanjut.

4.2.4 Hasil Evaluasi Model

Model *probabilistic deep learning* dan model Mateu & Jalilian (2022) mengeluarkan estimasi $\hat{\Lambda}_{ij}^{(s)}$, yaitu estimasi jumlah titik pada setiap *grid* B_{ij} , dan $\hat{\theta}_{sim} = \{\hat{\mu}, \hat{\alpha}, \hat{\sigma}, \hat{\phi}, \hat{\psi}\}$ pada *hidden layer*-nya. Estimasi $\hat{\Lambda}_{ij}^{(s)}$ dievaluasi menggunakan MSE sesuai dengan persamaan (3.12), sedangkan estimasi $\hat{\theta}_{sim}$ dievaluasi menggunakan RMSE pada persamaan (3.22) hingga persamaan (3.30).

Proses pelatihan model dan estimasi diulang sebanyak $n_{trial} = 100$ iterasi, sehingga dilakukan penghitungan rata-rata dan rentang kepercayaan untuk mendapatkan sebuah nilai akhir sebagai nilai pembanding. Nilai rata-rata dan rentang kepercayaan MSE dengan tingkat kepercayaan 95% dihitung menggunakan persamaan (3.14) dan persamaan (3.15). Tabel 4.3 menunjukkan perbandingan nilai tersebut antara model *probabilistic deep learning* dan model

Mateu & Jalilian (2022) dari 200 kali iterasi. Selain itu, Tabel 4.3 juga menunjukkan rata-rata waktu pelatihan model dan rentang kepercayaannya antara kedua metode.

Tabel 4.3 Perbandingan Waktu Pelatihan dan MSE antara Model Mateu & Jalilian (2022) dan *Probabilistic Deep Learning* dari 200 Iterasi

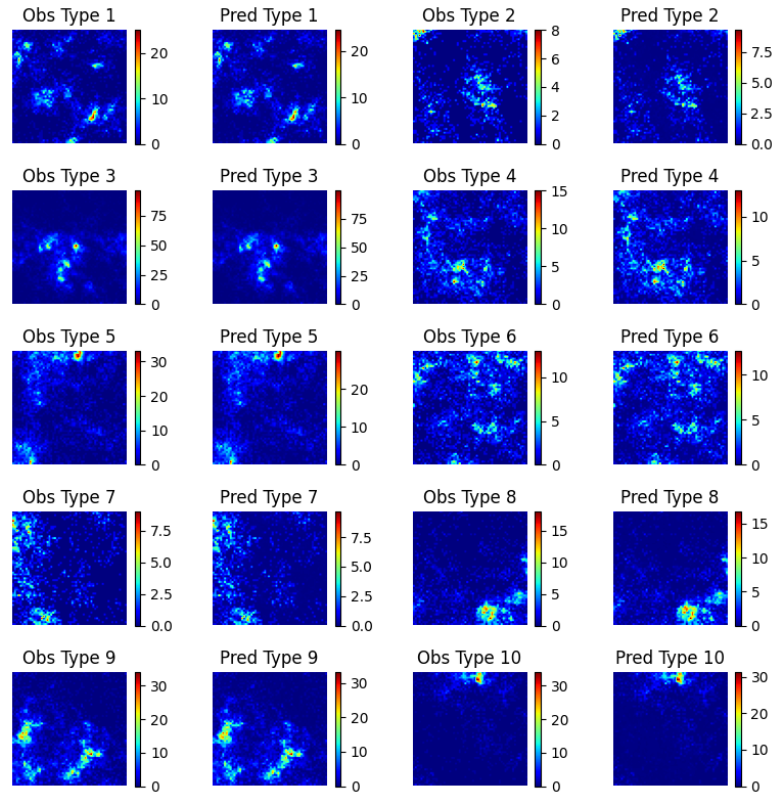
Pengukuran	Mateu & Jalilian (2022)	Probabilistic Deep Learning
Waktu Pelatihan (Dalam Detik)	92.474 ± 2.275	29.261 ± 0.863
MSE Data Latih	0.212 ± 0.038	0.095 ± 0.010
MSE Data Validasi	0.279 ± 0.049	0.135 ± 0.021
MSE Seluruh Data	0.241 ± 0.050	0.075 ± 0.014

Tabel 4.3 menunjukkan bahwa waktu pelatihan pada model *probabilistic deep learning* jauh lebih singkat daripada waktu pelatihan Mateu & Jalilian (2022). Model *probabilistic deep learning* juga mendapatkan nilai MSE dan *error* yang lebih kecil pada keseluruhan pengukuran yang menunjukkan bahwa *probabilistic deep learning* lebih konsisten dalam memberikan estimasi yang lebih baik dan lebih *robust* dalam memberikan estimasi di berbagai variasi pola *point pattern*.

Hasil estimasi $\widehat{\Lambda}_{ij}^{(s)}$ oleh salah satu iterasi beserta nilai aslinya $\Lambda_{ij}^{(s)}$ dapat divisualisasikan pada Gambar 4.6. Gambar 4.6 menunjukkan bahwa hasil estimasi $\widehat{\Lambda}_{ij}^{(s)}$ memiliki pola yang mirip dengan nilai $\Lambda_{ij}^{(s)}$ sebenarnya yang ditunjukkan pada Gambar 4.4. Hal tersebut mengindikasikan bahwa *probabilistic deep learning* memberikan estimasi $\widehat{\Lambda}_{ij}^{(s)}$ yang baik.

Pengukuran kebaikan estimasi $\widehat{\theta}_{sim}$ menggunakan RMSE dilakukan pada setiap parameter. Selain kedua metode berbasis *deep learning*, pengukuran RMSE juga melibatkan metode parametrik yang diperkenalkan oleh Choiruddin dkk (2020) sebagai pembandingan tambahan. Choiruddin dkk (2020) menggunakan teknik estimasi *least square* yang diregularisasi, sehingga *estimator* menjadi lebih stabil secara numerik. Penghitungan RMSE, rata-rata, dan rentang kepercayaan pada keseluruhan iterasi percobaan dilakukan kepada setiap parameter yang

diestimasi menggunakan persamaan (3.22) hingga persamaan (3.30). Hasil pengukuran tersebut pada ketiga metode ditunjukkan pada Tabel 4.4.



Gambar 4. 6 Perbandingan Hasil Estimasi $\hat{\Lambda}_{ij}^{(s)}$ pada Salah Satu Iterasi Model di Studi Simulasi

Tabel 4. 4 Perbandingan RMSE pada Metode Choiruddin dkk (2020), Mateu & Jalilian (2022) dan *Probabilistic Deep Learning* dari 200 Iterasi

Parameter	Choiruddin dkk (2020)	Mateu & Jalilian (2022)	Probabilistic Deep Learning
$\hat{\mu}$	3451.367 ± 426.956	3391.628 ± 279.971	3386.625 ± 297.207
$\hat{\alpha}$	0.246 ± 0.050	0.470 ± 0.076	0.317 ± 0.102
$\hat{\phi}$	0.085 ± 0.033	0.325 ± 0.107	0.237 ± 0.105
$\hat{\sigma}$	0.345 ± 0.057	0.979 ± 0.176	0.974 ± 0.176
$\hat{\psi}$	0.104 ± 0.016	0.400 ± 0.118	0.217 ± 0.073

Ketiga metode perbandingan dijalankan sebanyak 200 kali iterasi, namun terdapat 63% dari keseluruhan hasil estimasi Choiruddin dkk (2020) yang mendapatkan nilai RMSE sangat besar. Estimasi yang memiliki RMSE besar tersebut dianggap sebagai *outlier* dan tidak diikutsertakan pada perbandingan RMSE pada Tabel 4.4. Oleh sebab itu, dapat disimpulkan bahwa metode parametrik tidak menjamin hasil yang konvergen.

Secara umum, Tabel 4.4 menunjukkan bahwa metode Choiruddin dkk (2020) mendapatkan nilai RMSE terkecil secara signifikan pada mayoritas parameter. Namun demikian, proses estimasi pada metode Choiruddin dkk (2020) terdiri dari dua tahap estimasi, yaitu (1) estimasi *pair correlation function* (PCF) yang membutuhkan waktu sekitar 20 menit, dan (2) estimasi parameter menggunakan *least square estimator* yang membutuhkan waktu sekitar 4 menit, sehingga total waktu yang dibutuhkan adalah sekitar 24 menit untuk satu kali proses estimasi. Apabila jumlah tipe *multivariate* semakin bertambah, maka total waktu yang dibutuhkan oleh Choiruddin (2020) juga akan semakin lama. Dengan demikian, metode berbasis *deep learning* menawarkan fleksibilitas yang lebih baik dalam menangani jumlah tipe *multivariate* yang lebih besar karena memiliki waktu estimasi yang jauh lebih singkat daripada metode parametrik. Detail perbandingan rata-rata dan rentang kepercayaan waktu estimasi pada metode Choiruddin dkk (2020), Mateu & Jalilian (2022), dan *probabilistic deep learning* dirangkum pada Tabel 4.5.

Tabel 4. 5 Perbandingan Waktu Estimasi pada Choiruddin dkk (2020), Mateu & Jalilian (2022), dan Probabilistic Deep Learning

Metode	Waktu Estimasi / Pelatihan (Dalam Detik)
Choiruddin dkk (2020)	1372.346 ± 271.656
Mateu & Jalilian (2022)	92.474 ± 2.275
Probabilistic Deep Learning	29.261 ± 0.863

Sementara itu, ketika nilai RMSE antara kedua metode berbasis *deep learning* dibandingkan (Tabel 4.4), model *probabilistic deep learning* mendapatkan nilai RMSE yang lebih kecil pada parameter $\hat{\alpha}$, $\hat{\phi}$, dan $\hat{\psi}$, sedangkan nilai RMSE pada $\hat{\mu}$, dan $\hat{\sigma}$ tidak berbeda secara signifikan. Hal tersebut mengindikasikan bahwa

model *probabilistic deep learning* cenderung memberikan estimasi $\hat{\theta}_{sim}$ yang lebih baik daripada model Mateu & Jalilian (2022) dengan waktu estimasi yang lebih singkat (Tabel 4.5).

Perbandingan rata-rata parameter sebenarnya θ_{sim} dengan rata-rata hasil estimasi parameter pada metode Choiruddin dkk (2020), Mateu & Jalilian (2022), dan *probabilistic deep learning* ditunjukkan pada Lampiran 8. Tabel tersebut menunjukkan bahwa estimasi $\hat{\phi}$ dan $\hat{\sigma}$ oleh model Mateu & Jalilian (2022) dan *probabilistic deep learning* memiliki rata-rata nilai estimasi yang cukup berbeda dengan nilai sebenarnya. Hal ini juga didukung dengan nilai RMSE yang cukup besar pada kedua parameter tersebut (Tabel 4.4). Sementara pada parameter-parameter lain, perbedaan antara nilai estimasi dan nilai sebenarnya tidak signifikan, sehingga dapat disimpulkan bahwa model *probabilistic deep learning* memiliki kemampuan estimasi yang baik pada parameter-parameter tersebut.

4.3 Studi Terapan

Subbab 4.3.1 hingga 4.3.4 membahas tentang (1) pengumpulan *dataset*, (2) proses diskretisasi, (3) pelatihan model, dan (4) evaluasi model pada studi terapan. Model hanya dievaluasi berdasarkan MSE pada $\hat{\Lambda}_{ij}^{(s,t)}$ (tanpa evaluasi RMSE pada setiap parameter) karena *dataset* BCI tidak menyediakan nilai parameter sebenarnya θ_{bci} . Selain itu, evaluasi pada studi terapan tidak mengikutsertakan metode parametrik Choiruddin dkk (2020) karena tingginya dimensi data pada *dataset* BCI (ratusan tipe *multivariate* dengan T repetisi waktu), sehingga menyebabkan waktu estimasinya sangat lama.

4.3.1 Pengumpulan Dataset

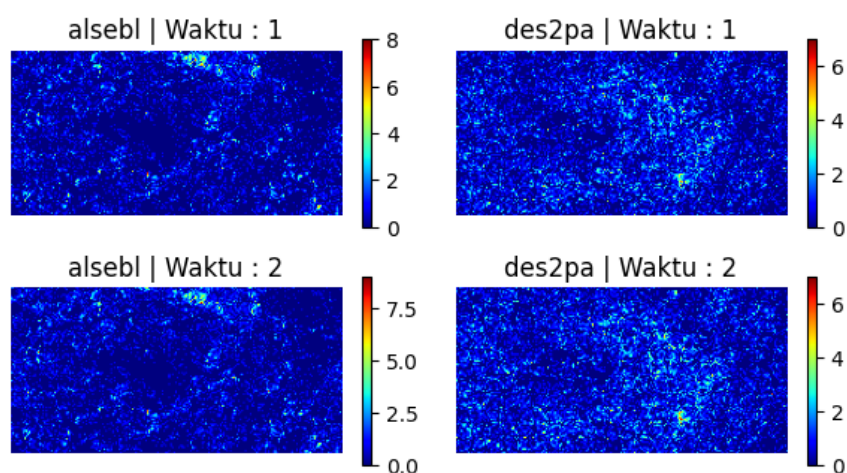
Studi terapan menggunakan *dataset* sekunder Barro Colorado Island (BCI) yang berisi lokasi koordinat ratusan spesies pohon pada hutan hujan tropis di Pulau Barro Colorado yang diobservasi pada 8 waktu berbeda. Setiap lokasi pohon pada masing-masing spesies dan waktu direpresentasikan sebagai sebuah titik pada area hutan seluas $500 \times 1000 \text{ m}^2$, sehingga kumpulan titik-titik tersebut membentuk data *point pattern* dengan banyak tipe *multivariate* dan waktu, atau disebut pula

sebagai *multivariate spatio-temporal point pattern*. Dataset BCI juga mengandung variabel *covariates* yang menggambarkan kondisi tanah pada area hutan tersebut.

Dataset BCI bersifat *open source* sehingga dapat diunduh secara bebas melalui Internet. Kemudian, diterapkan data *cleaning* untuk membuang data-data yang tidak perlu dan *missing values*. Hasil pembersihan data tersebut disimpan dalam bentuk *dataframe* yang memuat informasi waktu, spesies, dan koordinat lokasi pohon, seperti yang ditunjukkan pada Tabel 3.4. Library “spatstat” dipakai untuk mengubah informasi pada *dataframe* tersebut menjadi data *multivariate spatio-temporal point pattern*. Proses serupa juga dilakukan terhadap variabel *covariates*. Contoh *point pattern* dari 4 spesies pohon yang diobservasi sebanyak 8 kali ditunjukkan pada Gambar 3.6, sedangkan variabel *covariates* digambarkan pada Gambar 3.7 dan detail masing-masing variabel dijelaskan pada Tabel 3.3.

4.3.2 Diskretisasi Point Pattern dan Variabel Covariate

Diskretisasi terhadap *point pattern* memiliki proses yang serupa seperti pada studi simulasi. *Observation window* W dibagi ke dalam *grid-grid* B_{ij} dengan $i = 1, 2, \dots, 100$ dan $j = 1, 2, \dots, 200$, sehingga sebuah *point pattern* terdiri dari 20.000 *grid* berukuran kecil. Jumlah titik pada setiap *grid* B_{ij} dinotasikan sebagai $N(\mathbf{X}^{(s,t)} \cap B_{ij})$ dan diasumsikan sebagai $\Lambda_{ij}^{(s,t)}$. Hasil diskretisasi dinotasikan sebagai $\tilde{\mathcal{D}}_{bci}$. Gambar 4.7 menunjukkan visualisasi hasil diskretisasi pada 4 spesies yang diobservasi sebanyak 5 kali.



Gambar 4. 7 Contoh Plot Hasil Diskretisasi pada 2 Spesies dan 2 Waktu Observasi

Setiap kolom pada Gambar 4.7 menunjukkan spesies, sedangkan setiap baris menunjukkan urutan waktu observasi. Masing-masing plot dilengkapi dengan *heatmap* yang menunjukkan besaran jumlah titik, sehingga daerah berwarna terang menunjukkan jumlah titik yang banyak pada daerah tersebut. Variabel *covariate* terdiri dari $p = 13$ variabel dan didiskretisasi menggunakan persamaan (3.2). Hasil diskretisasi terhadap variabel *covariate* dinotasikan sebagai $\tilde{\mathcal{Z}}_{bci}$.

4.3.3 Pembagian Dataset dan Pelatihan Model

Sebelum proses pelatihan model *deep learning* dimulai, *dataset* $\tilde{\mathcal{D}}_{bci}, \tilde{\mathcal{Z}}_{bci}, \mathcal{J}_{bci}$ dibagi ke dalam data latih dan data validasi. Variabel $\tilde{\mathcal{D}}_{bci}$ merupakan data *point pattern* yang telah didiskretisasi, $\tilde{\mathcal{Z}}_{bci}$ merupakan variabel *covariate* yang telah didiskretisasi, dan \mathcal{J}_{bci} merupakan vektor kolom yang merepresentasikan waktu observasi dengan $t = 1, 2, \dots, T$. *Dataset* dibagi sesuai dengan penjelasan pada subbab 3.1.3 sehingga didapatkan $\tilde{\mathcal{D}}_{bci_{latih}}, \tilde{\mathcal{D}}_{bci_{val}}, \tilde{\mathcal{Z}}_{bci_{latih}}, \tilde{\mathcal{Z}}_{bci_{val}}, \mathcal{J}_{bci_{latih}},$ dan $\mathcal{J}_{bci_{val}}$.

Arsitektur model *probabilistic deep learning* ditunjukkan pada Tabel 4.1 dan *loss function* yang dipakai adalah MSE. Model *deep learning* pada studi terapan dilatih menggunakan Google Colaboratory dengan *processor dual core* Intel Xeon CPU @ 2.00GHz, RAM berukuran 13GB, dan Tesla T4 sebagai GPU *accelerator*.

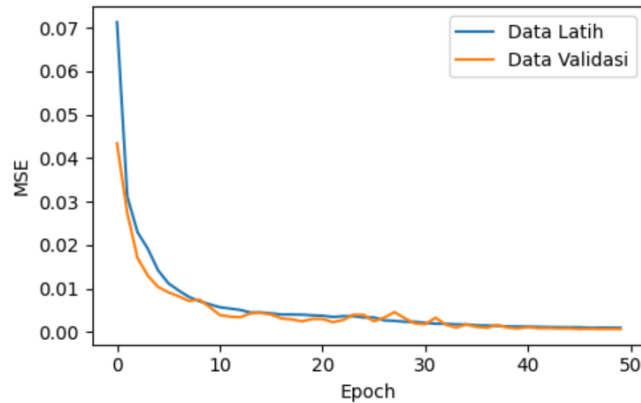
Proses pembelajaran berlangsung dengan cara mengubah bobot berdasarkan galat data latih. Model *deep learning* yang telah mengalami perubahan bobot kemudian dievaluasi menggunakan data validasi. Proses evaluasi tersebut berlangsung pada setiap *epoch* sehingga perubahan nilai MSE pada setiap *epoch* dapat divisualisasikan dalam bentuk grafik. Grafik tersebut seringkali disebut sebagai *training plot*.

Analisis pada studi terapan dibagi menjadi 2 bagian, yaitu analisis pada:

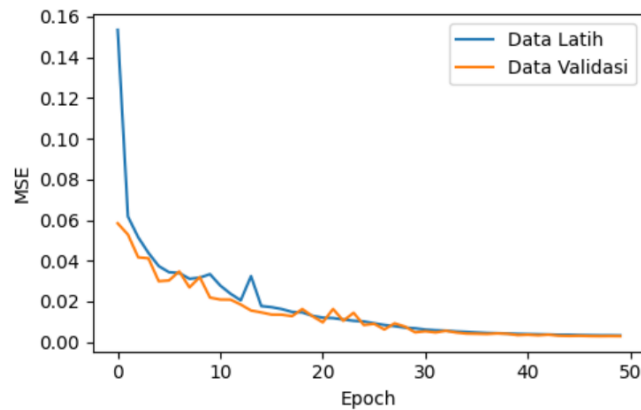
- 1) 25 spesies dengan 4 waktu observasi,
- 2) 100 spesies dengan 8 waktu observasi.

Evaluasi model dilakukan terhadap setiap jenis analisis untuk mengetahui bahwa model dapat memberikan estimasi yang baik pada beragam jumlah tipe *multivariate*

dan waktu. Oleh sebab itu, terdapat 2 model pula dengan struktur arsitektur serupa, namun memiliki jumlah *neuron* yang berbeda. Model untuk analisis pertama memiliki jumlah *neuron* lebih sedikit daripada model untuk analisis kedua yang memiliki lebih banyak spesies dan waktu observasi. Grafik *training plot* pada kedua model ditunjukkan pada Gambar 4.8 dan Gambar 4.9.



Gambar 4. 8 Grafik MSE Pelatihan Model dengan Jumlah Spesies 25 dan Waktu Observasi 4 pada Studi Terapan



Gambar 4. 9 Grafik MSE Pelatihan Model dengan Jumlah Spesies 100 dan Waktu Observasi 8 pada Studi Terapan

Kedua *training plot* pada Gambar 4.8 dan Gambar 4.9 tidak mengindikasikan terjadinya *overfitting* karena tren MSE antara data latih dan data validasi saling berimpit. Selain itu, nilai MSE juga mengalami tren penurunan pada kedua *training plot* yang menandakan bahwa pelatihan kedua model berlangsung dengan baik. Hal tersebut juga menunjukkan bahwa model *probabilistic deep*

learning cukup *robust* untuk menganalisis jumlah spesies dan waktu observasi yang sedikit maupun banyak.

Kedua model *probabilistic deep learning* yang telah dilatih terhadap masing-masing jenis analisis lalu digunakan untuk melakukan prediksi pada data latih, data validasi, dan keseluruhan dataset. Nilai MSE yang didapatkan pada ketiga prediksi tersebut lalu dicatat ke dalam berkas “.csv” untuk analisis selanjutnya. Proses pelatihan, estimasi, dan pencatatan tersebut diulang sebanyak $n_{trial} = 100$ kali pada analisis pertama dan $n_{trial} = 5$ kali pada analisis kedua.

Proses pelatihan, estimasi, dan pencatatan yang serupa juga diterapkan terhadap model Mateu & Jalilian (2022). Model tersebut dilatih untuk menganalisis jumlah spesies dan waktu observasi yang serupa dengan model *probabilistic deep learning*. Analisis pertama dilakukan sebanyak $n_{trial} = 100$ kali iterasi, sedangkan analisis kedua dilakukan sebanyak $n_{trial} = 5$ kali iterasi. Hasil pencatatan MSE berdasarkan data latih, data validasi, dan keseluruhan *dataset* disimpan ke dalam berkas “.csv”.

4.3.4 Hasil Evaluasi Model

Model *probabilistic deep learning* dan model Mateu & Jalilian (2022) memberikan estimasi jumlah titik pada setiap *grid* B_{ij} pada setiap spesies dan waktu observasi yang dinotasikan sebagai $\widehat{\Lambda}_{ij}^{(s,t)}$. Selain itu, model juga mengeluarkan estimasi $\widehat{\theta}_{bci} = \{\widehat{\mu}, \widehat{\alpha}, \widehat{\sigma}, \widehat{\phi}, \widehat{\psi}, \widehat{\varphi}, \widehat{\rho}, \widehat{\zeta}\}$ pada *hidden layer*-nya. Namun, evaluasi kebaikan kedua model hanya dilakukan terhadap estimasi $\widehat{\Lambda}_{ij}^{(s,t)}$ karena *dataset* BCI tidak menyediakan nilai parameter sebenarnya $\theta_{bci} = \{\mu, \alpha, \sigma, \phi, \psi, \varphi, \rho, \zeta\}$. Estimasi $\widehat{\Lambda}_{ij}^{(s,t)}$ dievaluasi menggunakan MSE pada persamaan (3.13).

1) Analisis terhadap 25 spesies dengan 4 waktu observasi

Pelatihan model dan estimasi dilakukan sebanyak $n_{trial} = 100$ iterasi, sehingga didapatkan 100 data yang mengandung nilai MSE berdasarkan prediksi pada data latih, data validasi, dan keseluruhan dataset seperti yang ditunjukkan pada

Tabel 4.6. Selain itu, waktu pelatihan model pada setiap iterasi juga disimpan sebagai bahan perbandingan antara kedua metode.

Tabel 4. 6 Struktur Data Pencatatan Nilai MSE dan Waktu Pelatihan pada Setiap Iterasi

No.	Waktu Pelatihan	MSE Data Latih	MSE Data Validasi	MSE Keseluruhan Data
1	$Waktu_{latih_1}$	MSE_{latih_1}	MSE_{val_1}	MSE_{all_1}
2	$Waktu_{latih_2}$	MSE_{latih_2}	MSE_{val_2}	MSE_{all_2}
...
j	$Waktu_{latih_j}$	MSE_{latih_j}	MSE_{val_j}	MSE_{all_j}
...
n_{trial}	$Waktu_{latih_{ntrial}}$	$MSE_{latih_{ntrial}}$	$MSE_{val_{ntrial}}$	$MSE_{all_{ntrial}}$

Kemudian dilakukan penghitungan rata-rata dan rentang kepercayaan dengan tingkat kepercayaan 95% terhadap data kumpulan MSE dan waktu pelatihan tersebut menggunakan persamaan (3.14) dan persamaan (3.15).

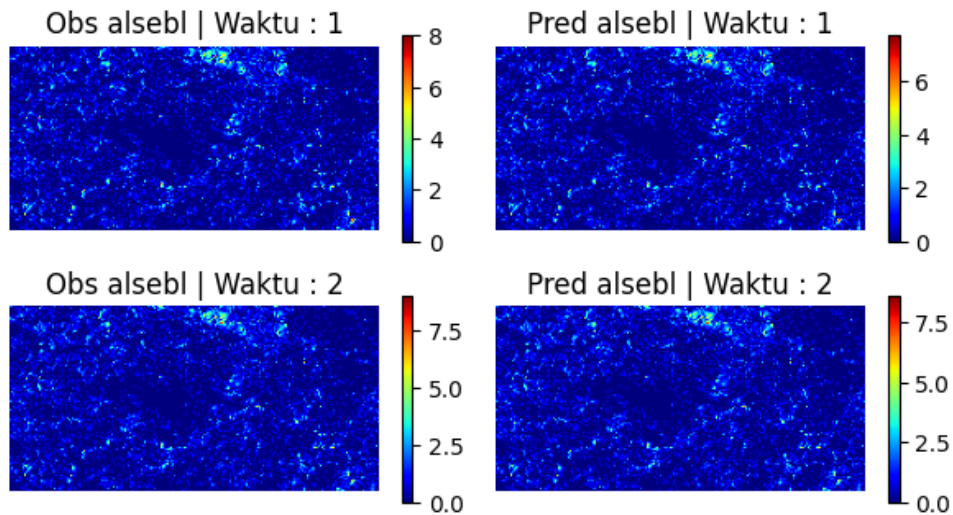
Tabel 4. 7 Perbandingan Waktu Pelatihan dan MSE antara Model Mateu & Jalilian (2022) dan *Probabilistic Deep Learning* dari 100 Iterasi dengan Jumlah Spesies 25 dan Waktu Observasi 4

Pengukuran	Mateu & Jalilian (2022)	Probabilistic Deep Learning
Waktu Pelatihan (Dalam Detik)	549.570 ± 1.981	194.711 ± 3.601
MSE Data Latih	0.002 ± 0.000	0.001 ± 0.000
MSE Data Validasi	0.003 ± 0.000	0.001 ± 0.000
MSE Seluruh Data	0.002 ± 0.000	0.001 ± 0.000

Tabel 4.7 menunjukkan perbandingan nilai rata-rata MSE dan waktu pelatihan beserta rentang kepercayaannya antara model *probabilistic deep learning* dan model Mateu & Jalilian (2022). Serupa dengan hasil perbandingan pada studi simulasi, waktu pelatihan pada model *probabilistic deep learning* jauh lebih singkat daripada model Mateu & Jalilian (2022). Selain itu, model *probabilistic deep*

learning mendapatkan nilai MSE dan *error* yang lebih kecil. Hal tersebut menandakan bahwa *probabilistic deep learning* mampu memberikan estimasi yang lebih baik dan lebih konsisten daripada model Mateu & Jalilian (2022).

Gambar 4.10 menunjukkan visualisasi dari hasil estimasi $\hat{\Lambda}_{ij}^{(s,t)}$ pada model *probabilistic deep learning*. Gambar 4.10 terdiri dari 2 kolom, dengan kolom sebelah kiri merupakan nilai $\Lambda_{ij}^{(s,t)}$ sebenarnya, sedangkan kolom kanan menunjukkan hasil estimasi untuk spesies “alsebl”. Gambar 4.10 memperlihatkan bahwa estimasi $\hat{\Lambda}_{ij}^{(s,t)}$ dari model *probabilistic deep learning* memiliki pola persebaran titik yang sangat mirip dengan nilai sebenarnya $\Lambda_{ij}^{(s,t)}$.



Gambar 4.10 Hasil Estimasi $\hat{\Lambda}_{ij}^{(s,t)}$ dengan Jumlah Spesies 25 dan Waktu Observasi 4 pada Salah Satu Iterasi

2) Analisis terhadap 100 spesies dengan 8 waktu observasi

Analisis ini menggunakan $n_{trial} = 5$ iterasi karena jumlah spesies dan waktu observasi yang dianalisis lebih banyak daripada analisis pertama, sehingga waktu pelatihan kedua model menjadi lebih lama. Pencatatan waktu pelatihan dan nilai MSE dilakukan pada setiap iterasi seperti yang ditunjukkan pada Tabel 4.6. Kemudian dilakukan penghitungan rata-rata dan rentang kepercayaan dengan tingkat kepercayaan 95% terhadap data kumpulan MSE dan waktu pelatihan menggunakan persamaan (3.14) dan (3.15). Tabel 4.8 menunjukkan perbandingan

nilai rata-rata MSE dan waktu pelatihan, beserta rentang kepercayaannya antara model *probabilistic deep learning* dan Mateu & Jalilian (2022).

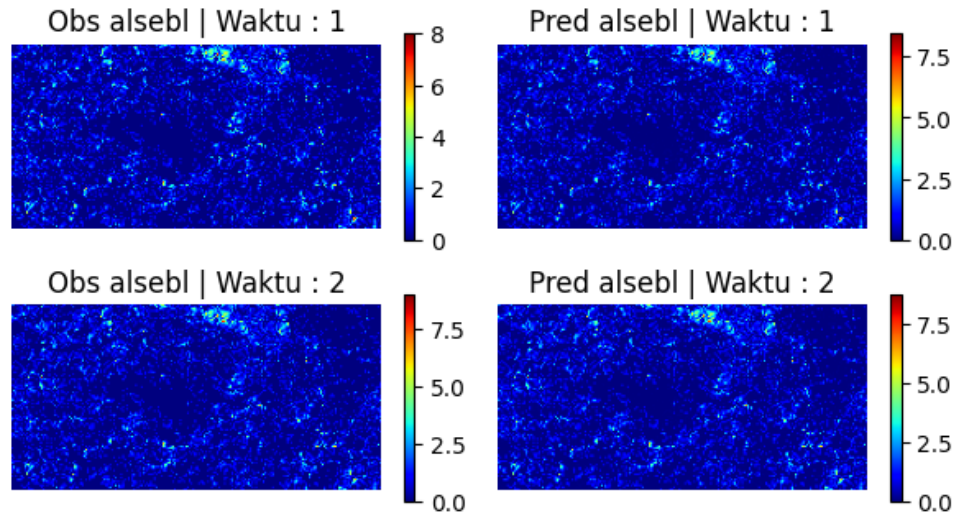
Tabel 4.8 menunjukkan bahwa waktu pelatihan antara kedua metode berbeda sangat signifikan. Model *probabilistic deep learning* memiliki waktu pelatihan yang lebih singkat hingga lebih dari 10 kali lipat daripada model Mateu & Jalilian (2022). Model *probabilistic deep learning* juga mendapatkan nilai MSE dan *error* yang lebih kecil pada seluruh pengukuran. Hal tersebut mengindikasikan bahwa *probabilistic deep learning* memberikan estimasi yang lebih baik dan lebih konsisten pada kasus analisis dengan ratusan jumlah tipe *multivariate* dan banyak waktu observasi. Hasil estimasi $\widehat{\Lambda}_{ij}^{(s,t)}$ dari model *probabilistic deep learning* dapat divisualisasikan dalam bentuk plot pada Gambar 4.11.

Tabel 4. 8 Perbandingan Waktu Pelatihan dan MSE antara Model Mateu & Jalilian (2022) dan *Probabilistic Deep Learning* dari 5 Iterasi dengan Jumlah Spesies 100 dan Waktu Observasi 8

Pengukuran	Mateu & Jalilian (2022)	Probabilistic Deep Learning
Waktu Pelatihan (Dalam Detik)	3663.352 ± 10.792	321.274 ± 4.643
MSE Data Latih	0.004 ± 0.000	0.003 ± 0.000
MSE Data Validasi	0.005 ± 0.000	0.003 0.000
MSE Seluruh Data	0.004 ± 0.001	0.002 ± 0.000

Kolom sebelah kiri pada Gambar 4.11 menunjukkan nilai sebenarnya pada spesies “alsebl”, sedangkan kolom sebelah kanan merupakan hasil estimasi $\widehat{\Lambda}_{ij}^{(s,t)}$ untuk spesies tersebut. Sementara itu, setiap baris menunjukkan indeks waktu terjadinya sensus. Serupa dengan analisis sebelumnya, estimasi $\widehat{\Lambda}_{ij}^{(s,t)}$ dari model *probabilistic deep learning* memiliki pola yang sangat mirip dengan nilai sebenarnya $\Lambda_{ij}^{(s,t)}$. Oleh sebab itu, berdasarkan nilai MSE dan plot secara visual, maka dapat disimpulkan bahwa model *probabilistic deep learning* cukup *robust*

untuk menganalisis puluhan hingga ratusan tipe *multivariate* yang mengandung unsur *temporal*.



Gambar 4.11 Hasil Estimasi $\hat{\Lambda}_{ij}^{(s,t)}$ dengan Jumlah Spesies 100 dan Waktu Observasi 8 pada Salah Satu Iterasi

4.3.5 Interpretasi Model

Model *probabilistic deep learning* yang telah dilatih kemudian digunakan untuk estimasi parameter $\hat{\theta}_{bci}$ pada kedua jenis analisis, yaitu (1) analisis terhadap 25 species dengan 4 waktu observasi, dan (2) analisis terhadap 100 spesies dengan 8 waktu observasi. Hasil estimasi tersebut kemudian dipakai untuk mengetahui pola persebaran pohon pada dataset BCI. Penjelasan terkait metodologi interpretasi model dijelaskan pada subbab 3.3.4.

1) Analisis terhadap 25 spesies dengan 4 waktu observasi

a. Interpretasi untuk hasil estimasi $\hat{\Lambda}$

Estimasi $\hat{\Lambda} = [\hat{\Lambda}^{(s,t)}]$ menunjukkan estimasi jumlah titik pada *point pattern* untuk spesies ke- s dan waktu observasi ke- t . Hasil estimasi $\hat{\Lambda}$ untuk 25 spesies yang diobservasi mulai tahun 1981 hingga 1995 ditunjukkan pada Tabel 4.9. Estimasi $\hat{\Lambda}$ pada Tabel 4.9 mengindikasikan adanya variasi ekspektasi jumlah titik yang cukup menonjol pada seluruh spesies. Variasi ini mencerminkan beragam pola persebaran pohon yang unik pada setiap spesies. Meskipun demikian, ekspektasi jumlah titik untuk masing-masing spesies pada keseluruhan tahun

observasi tidak mengalami perubahan yang cukup signifikan. Hal ini menunjukkan bahwa terdapat korelasi temporal cukup kuat yang mengindikasikan stabilitas jumlah titik dari waktu ke waktu.

Tabel 4. 9 Hasil Estimasi $\hat{\Lambda}$ untuk 25 Spesies dengan 4 Waktu Observasi

Nama Spesies	1981	1985	1990	1995
acaldi	1729.366	1442.021	1182.077	859.62
alibed	372.592	414.869	470.85	475.065
alsebl	7760.904	8481.343	9118.68	9190.738
anaxpa	719.147	734.915	886.379	993.671
andiin	418.669	427.408	436.339	428.51
annoac	605.094	645.469	710.62	705.051
apeime	474.767	435.572	427.452	392.827
aspicr	549.407	587.423	618.726	621.565
beilpe	2648.804	3052.932	3249.871	3183.955
brosal	970.219	1011.07	1054.124	1045.5
calolo	780.625	862.935	1041.559	1151.078
cappfr	3795.39	4091.876	4097.997	4090.595
caseac	562.115	596.771	632.294	632.827
cassel	873.64	974.483	1081.496	1131.811
cecrin	607.324	554.409	517.776	497.745
cha2sc	291.474	343.038	405.02	440.358
chr1ec	554.234	553.347	538.701	537.813
chr2ar	531.683	604.163	816.649	865.032
coccma	543.678	564.644	612.885	625.703
cordbi	872.322	939.756	1230.826	1182.771
cordla	1805.377	1835.622	1875.982	1740.586
cou2cu	1631.595	1824.046	2128.741	2243.756
crothi	806.722	819.409	1221.059	773.102
cupasy	1080.041	1214.807	1359.249	1464.773
des2pa	11628.81	12272.93	12433.65	12125.7

b. Interpretasi untuk parameter $\hat{\mu}$

Tabel 4.10 menunjukkan hasil estimasi $\hat{\mu}$ untuk 25 spesies yang diobservasi pada 4 waktu berbeda. Parameter $\mu = [\mu^{(s,t)}]$ menunjukkan ekspektasi jumlah titik pada *point pattern* untuk spesies s dan waktu observasi t tanpa

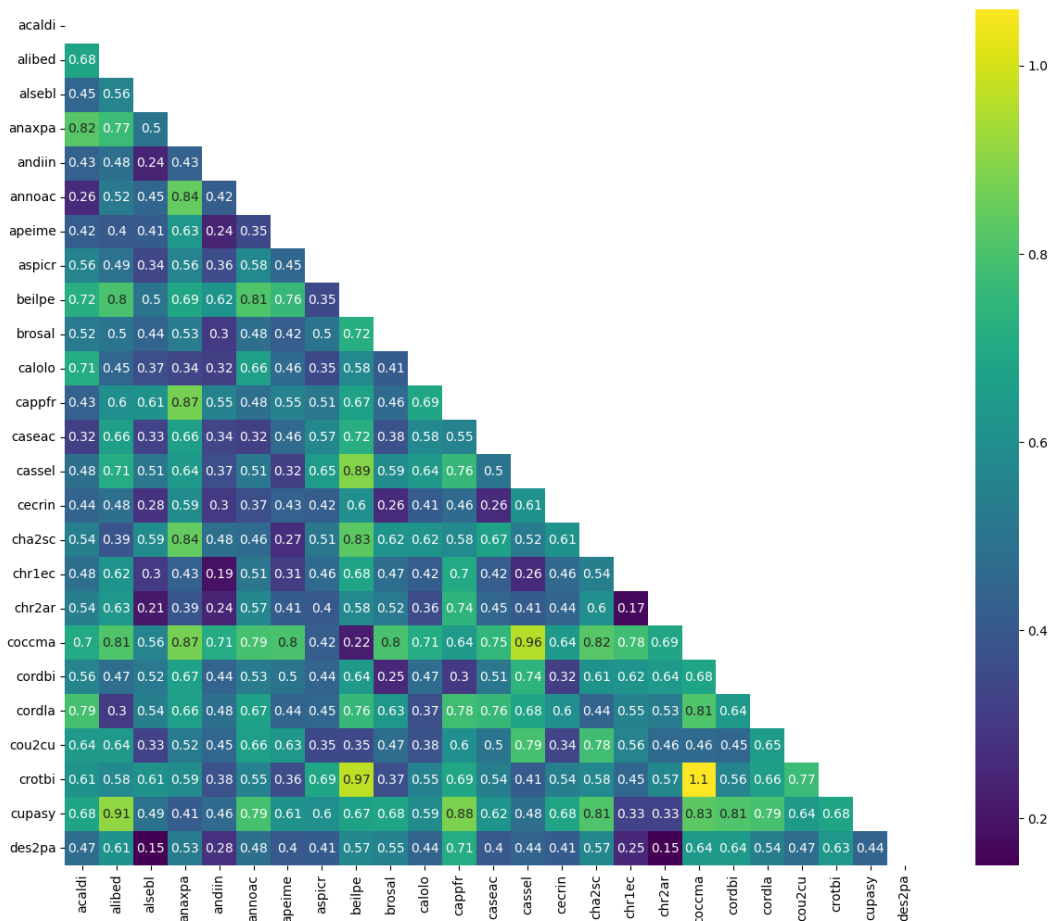
dipengaruhi oleh interaksi antar spesies dan interaksi antar titik pada spesies yang sama. Estimasi $\hat{\mu}$ pada Tabel 4.10 menunjukkan bahwa terdapat perbedaan ekspektasi jumlah titik yang cukup signifikan pada beberapa spesies. Hal tersebut menunjukkan keberagaman pola persebaran lokasi pohon setiap spesies. Selain itu, ekspektasi jumlah titik setiap spesies pada keseluruhan tahun observasi tidak mengalami perbedaan yang signifikan, sehingga menunjukkan bahwa korelasi antar waktu observasi tergolong kuat.

Tabel 4. 10 Hasil Estimasi $\hat{\mu}$ untuk 25 Spesies dengan 4 Waktu Observasi

Nama Spesies	1981	1985	1990	1995
acaldi	1729.352	1442.026	1182.083	859.623
alibed	372.59	414.877	470.827	475.063
alsebl	7760.904	8481.345	9118.68	9190.737
anaxpa	719.15	734.918	886.385	993.667
andiin	418.662	427.41	436.342	428.515
annoac	605.087	645.469	710.619	705.039
apeime	474.769	435.582	427.453	392.826
aspicr	549.423	587.422	618.721	621.557
beilpe	2648.81	3052.93	3249.874	3183.951
brosal	970.218	1011.079	1054.129	1045.506
calolo	780.625	862.929	1041.554	1151.084
cappfr	3795.39	4091.875	4097.995	4090.595
caseac	562.118	596.781	632.29	632.833
cassel	873.64	974.486	1081.501	1131.806
cecrin	607.325	554.409	517.778	497.745
cha2sc	291.471	343.015	405.019	440.36
chr1ec	554.236	553.348	538.7	537.814
chr2ar	531.686	604.16	816.649	865.019
cocma	543.673	564.641	612.908	625.701
cordbi	872.321	939.755	1230.832	1182.774
cordla	1805.368	1835.619	1875.982	1740.587
cou2cu	1631.593	1824.047	2128.743	2243.76
crotbi	806.72	819.411	1221.06	773.107
cupasy	1080.036	1214.802	1359.254	1464.768
des2pa	11628.81	12272.93	12433.65	12125.7

c. Interpretasi untuk $\hat{\alpha}$

Parameter $\hat{\alpha}$ merupakan matriks berukuran $m \times a$ yang menunjukkan pola interaksi antar tipe *multivariate* (Waagepetersen dkk, 2016). Dalam penerapannya, estimasi jarak antar tipe *multivariate* (spesies) s dan s' dapat diketahui dengan menghitung jarak Euclid $\|\hat{\alpha}_s - \hat{\alpha}_{s'}\|$. Semakin jauh estimasi jarak antara kedua spesies, maka kedua spesies memiliki pola persebaran titik yang semakin berbeda. Gambar 4.12 menunjukkan estimasi jarak antar 25 spesies. Warna kuning terang menunjukkan jarak Euclid yang saling berjauhan antara dua spesies, sedangkan warna biru gelap menunjukkan jarak Euclid yang berdekatan.



Gambar 4.12 Heatmap Plot yang Menunjukkan Jarak Euclid Antar 25 Spesies

d. Interpretasi untuk $\hat{\sigma}$

Parameter $\sigma = [\sigma^{(s,t)}]$ menunjukkan besarnya deviasi ekspektasi jumlah titik μ pada *point pattern* untuk spesies ke- s dan waktu observasi ke- t . Nilai σ yang

besar mengindikasikan bahwa nilai intensitas μ sangat bervariasi pada seluruh wilayah. Pada kasus seperti ini, titik cenderung berjumlah banyak pada suatu area tertentu, dan berjumlah sedikit di area-area lain. Sementara itu, nilai σ kecil menunjukkan persebaran nilai intensitas μ yang cenderung rata (*uniform*) pada keseluruhan wilayah *point pattern*.

Tabel 4. 11 Hasil Estimasi $\hat{\sigma}$ untuk 25 Spesies dengan 4 Waktu Observasi

Nama Spesies	1981	1985	1990	1995
acaldi	0.046358	0.043874	0.022353	0.078601
alibed	0.005031	0.135856	0.31465	0.004747
alsebl	0.005292	0.002671	0.000741	0.000424
anaxpa	0.080697	0.007595	0.019639	0.012693
andiin	0.130678	0.002895	0.00097	0.003061
annoac	0.227123	0.000839	0.012003	0.123594
apeime	0.001028	0.162702	0.197378	0.001941
aspicr	0.150276	0.004961	0.002224	0.009175
beilpe	0.013181	0.007347	0.009294	0.012587
brosal	0.067431	0.009268	0.086881	0.000377
calolo	0.046794	0.000583	0.015448	0.010922
cappfr	0.001149	0.004984	0.003606	0.001863
caseac	0.058167	0.133993	0.097734	0.07988
cassel	0.005153	0.123852	0.00088	0.015763
cecrin	0.00232	0.004157	0.251082	0.038081
cha2sc	0.007605	0.305167	0.007768	0.005084
chr1ec	0.053262	0.001562	0.065475	0.242136
chr2ar	0.249726	0.001128	0.010496	0.091702
cocma	0.092741	0.01046	0.168372	0.002268
cordbi	0.124353	0.022165	0.093021	0.002598
cordla	0.060374	0.006131	0.005521	0.0174
cou2cu	0.003221	0.009281	0.013511	0.008722
crotbi	3.29E-05	0.088302	0.045138	0.007062
cupasy	0.001396	0.0049	0.113632	0.009008
des2pa	0.00259	0.00257	0.003686	0.004843

Tabel 4.11 menunjukkan hasil estimasi $\hat{\sigma}$ untuk 25 spesies yang diobservasi pada 4 waktu berbeda. Berdasarkan Tabel 4.11, estimasi $\hat{\sigma}$ sangat bervariasi pada setiap spesies dan waktu observasi. Hal tersebut mengindikasikan

keberagaman pola interaksi yang terjadi pada setiap spesies. Keberagaman tersebut dapat dipengaruhi oleh variabel-variabel lingkungan ketika sensus berlangsung.

e. Interpretasi untuk *Proportion of Variance (PV)*

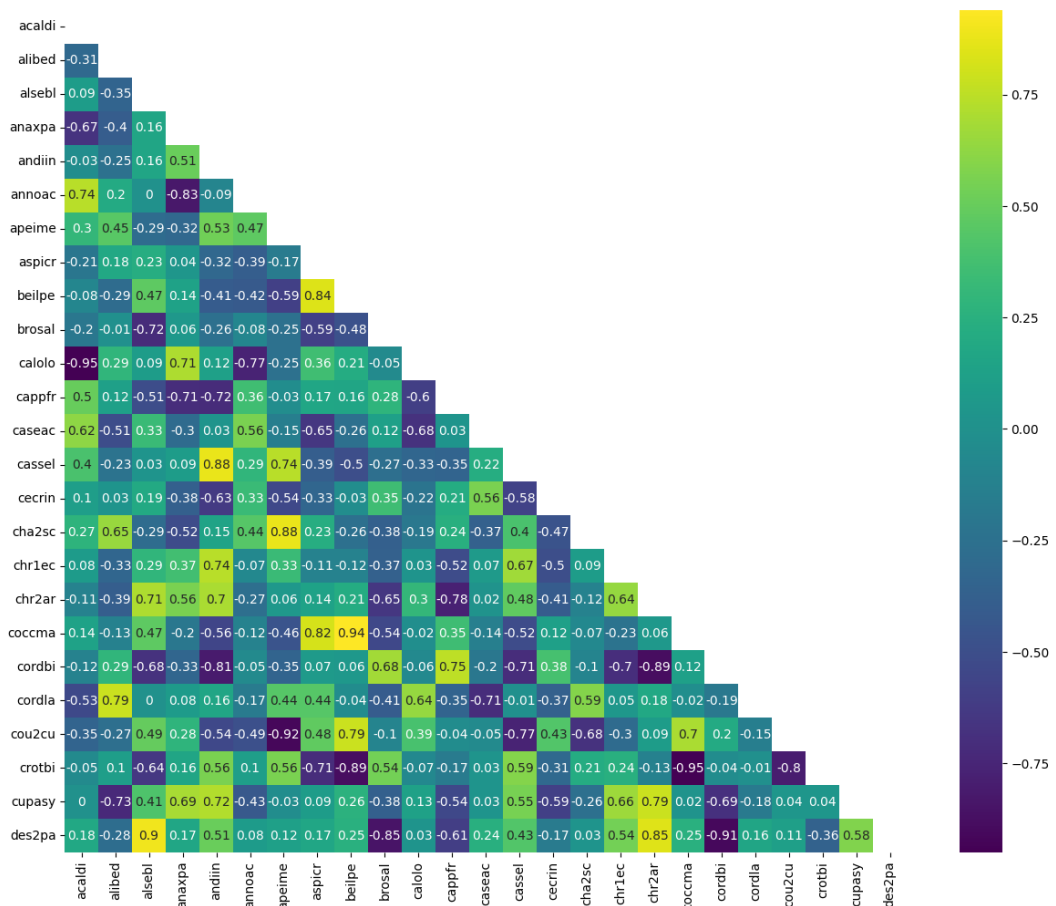
Proportion of variance (PV) merupakan pengukuran yang menunjukkan proporsi antara komponen interaksi antar spesies dengan total variansi keseluruhan (Choiruddin dkk, 2020). PV dihitung menggunakan persamaan (2.26). Semakin tinggi nilai PV, maka semakin besar pula pengaruh interaksi antar spesies dibandingkan dengan pengaruh internal spesies itu sendiri.

Tabel 4. 12 Hasil Estimasi PV untuk 25 Spesies dengan 4 Waktu Observasi

Nama Spesies	1981	1985	1990	1995
acaldi	0.987	0.989	0.997	0.964
alibed	1	0.911	0.657	1
alsebl	0.999	1	1	1
anaxpa	0.973	1	0.998	0.999
andiin	0.451	0.999	1	0.999
annoac	0.745	1	0.999	0.908
apeime	1	0.745	0.664	1
aspicr	0.805	1	1	0.999
beilpe	0.999	1	1	0.999
brosal	0.932	0.999	0.892	1
calolo	0.978	1	0.998	0.999
cappfr	1	1	1	1
caseac	0.969	0.854	0.916	0.943
cassel	1	0.934	1	0.999
cecrin	1	1	0.404	0.967
cha2sc	1	0.712	1	1
chr1ec	0.969	1	0.955	0.606
chr2ar	0.613	1	0.999	0.921
coccma	0.979	1	0.934	1
cordbi	0.878	0.996	0.928	1
cordla	0.985	1	1	0.999
cou2cu	1	0.999	0.999	0.999
crotdi	1	0.961	0.989	1
cupasy	1	1	0.958	1
des2pa	1	1	1	1

Tabel 4.12 menunjukkan estimasi PV pada 25 spesies yang diobservasi pada 4 waktu berbeda. Tabel 4.12 menunjukkan bahwa hampir seluruh spesies memiliki estimasi PV yang mendekati 1, sehingga dapat dikatakan apabila persebaran pohon lebih dipengaruhi oleh interaksi antar spesies daripada interaksi akibat faktor internal dari suatu spesies.

f. Korelasi antar Spesies dan Waktu



Gambar 4.13 Heatmap Plot yang Menunjukkan Korelasi Antar 25 Spesies dengan 4 Waktu Observasi pada Tahun 1995

Nilai korelasi antar spesies dihitung menggunakan persamaan (2.27) dan persamaan (2.28). Hasil penghitungan nilai korelasi pada observasi tahun 1995 divisualisasikan menggunakan *heatmap plot* pada Gambar 4.13. Warna kuning terang menunjukkan korelasi positif yang kuat antara dua spesies, sedangkan warna biru gelap menunjukkan korelasi negatif yang kuat antara dua spesies. Apabila kedua spesies s dan s' memiliki korelasi positif kuat, maka kedua spesies memiliki

kecenderungan kenaikan atau penurunan jumlah titik yang sama pada setiap koordinat u . Secara umum, mayoritas nilai korelasi antar spesies berada di rentang -0.3 hingga 0.3.

Distribusi estimasi nilai korelasi antar spesies pada setiap koordinat u dari observasi tahun 1981 hingga 1995 ditunjukkan pada Tabel 4.13. Tabel 4.13 menunjukkan bahwa distribusi korelasi antar spesies tidak mengalami perubahan yang signifikan dari tahun 1981 hingga 1995. Hal tersebut menunjukkan bahwa korelasi antar waktu observasi sangat kuat. Keseluruhan tahun didominasi oleh nilai korelasi antar spesies yang lemah pada rentang -0.3 hingga 0.3, sedangkan proporsi nilai korelasi kuat (kurang dari -0.6 dan lebih dari 0.6) adalah yang paling sedikit diantara keseluruhan interval.

Tabel 4. 13 Distribusi (Dalam Persen) Estimasi Nilai Korelasi Antar 25 Spesies dengan Waktu Observasi 4

Interval		1981	1985	1990	1995
-1	-0.6	9.67	11.0	10.0	10.33
-0.6	-0.3	17.33	15.33	14.33	17.0
-0.3	0	22.33	23.0	25.0	22.0
0	0.3	27.33	27.33	28.67	28.0
0.3	0.6	15.33	14.67	13.0	13.67
0.6	1	8.0	8.67	9.0	9.0

2) Analisis terhadap 100 spesies dengan 8 waktu observasi

a. Interpretasi untuk hasil estimasi $\hat{\Lambda}$

Tabel 4. 14 Distribusi Estimasi $\hat{\Lambda}$ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		1981	1985	1990	1995
0	2000	73	72	68	70
2000	4000	18	16	21	19
4000	6000	2	5	4	5
6000	8000	3	1	1	0
8000	10000	0	2	2	2
> 10000		4	4	4	4

Tabel 4. 14 Lanjutan Distribusi Estimasi $\hat{\Lambda}$ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		2000	2005	2010	2015
0	2000	75	74	72	70
2000	4000	15	17	19	21
4000	6000	4	3	3	3
6000	8000	1	1	1	1
8000	10000	1	1	1	1
> 10000		4	4	4	4

Distribusi hasil estimasi $\hat{\Lambda}$ untuk 100 spesies dan 8 waktu observasi ditunjukkan pada Tabel 4.14. Tabel 4.14 menunjukkan bahwa sebagian besar spesies memiliki estimasi ekspektasi jumlah titik yang berkisar diantara 0 hingga 4000. Sementara itu, hanya sejumlah kecil spesies yang memiliki ekspektasi jumlah titik lebih dari 4000. Pada keseluruhan tahun, fluktuasi distribusi $\hat{\Lambda}$ umumnya terjadi pada rentang dibawah 6000 yang mengindikasikan ketatnya persaingan antar spesies pada rentang tersebut.

b. Interpretasi untuk parameter $\hat{\mu}$

Tabel 4.15 menunjukkan distribusi hasil estimasi $\hat{\mu}$ untuk 100 spesies yang diobservasi sebanyak 8 kali. Mayoritas spesies memiliki ekspektasi jumlah titik pada rentang 0 hingga 4000, dan hanya sedikit spesies yang memiliki ekspektasi jumlah titik diatas 4000.

Tabel 4. 15 Distribusi Estimasi $\hat{\mu}$ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		1981	1985	1990	1995
0	2000	73	72	68	70
2000	4000	18	16	21	19
4000	6000	2	5	4	5
6000	8000	3	1	1	0
8000	10000	0	2	2	2
>10000		4	4	4	4

Tabel 4. 15 Lanjutan Distribusi Estimasi $\hat{\mu}$ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		2000	2005	2010	2015
0	2000	75	74	72	70
2000	4000	15	17	19	21
4000	6000	4	3	3	3
6000	8000	1	1	1	1
8000	10000	1	1	1	1
>10000		4	4	4	4

Pada keseluruhan tahun, distribusi jumlah spesies cenderung berubah-ubah pada interval kurang dari 4000. Hal tersebut menunjukkan bahwa kompetisi persebaran pohon pada rentang tersebut cukup ketat.

c. Interpretasi untuk parameter $\hat{\alpha}$

Tabel 4. 16 Distribusi Jarak Euclid (Dalam Persen) antara 100 Spesies

Interval		Prosentase
0	0.1	0.83
0.1	0.2	13.35
0.2	0.3	36.48
0.3	0.4	33.25
0.4	0.5	13.56
> 0.5		2.53

Tabel 4.16 menunjukkan distribusi jarak Euclid $\|\hat{\alpha}_s - \hat{\alpha}_{s'}\|$ antar 100 spesies dalam satuan persentase. Tabel 4.16 mengindikasikan bahwa mayoritas spesies memiliki jarak pada rentang 0.2 hingga 0.4, sedangkan jarak antar spesies yang kecil (kurang dari 0.1) dan jarak antar spesies yang besar (lebih dari 0.5) hanya dimiliki oleh sebagian kecil spesies. Hal ini menunjukkan bahwa hanya sedikit spesies yang saling berinteraksi sangat kuat atau sangat lemah, dan sebagian besar spesies memiliki interaksi moderat.

d. Interpretasi untuk $\hat{\sigma}$

Tabel 4. 17 Distribusi Estimasi $\hat{\sigma}$ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		1981	1985	1990	1995
0	0.02	66	64	70	58
0.02	0.04	10	9	9	16
0.04	0.06	6	9	5	6
0.06	0.08	1	5	3	5
0.08	0.1	6	1	3	6
> 0.1		11	12	10	9

Tabel 4. 17 Lanjutan Distribusi Estimasi $\hat{\sigma}$ (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		2000	2005	2010	2015
0	0.02	61	59	60	61
0.02	0.04	7	12	3	8
0.04	0.06	4	6	8	9
0.06	0.08	10	7	7	5
0.08	0.1	3	3	7	4
> 0.1		15	13	15	13

Distribusi estimasi $\hat{\sigma}$ untuk 100 spesies yang diobservasi pada tahun 1981 hingga 2015 ditunjukkan pada Tabel 4.17. Sebagian besar spesies memiliki estimasi $\hat{\sigma}$ bernilai kecil, sehingga mayoritas spesies tersebut memiliki persebaran individu pohon yang merata pada keseluruhan area hutan. Pada estimasi $\hat{\sigma}$ yang sangat kecil (kurang dari 0.02) dan sangat besar (lebih dari 0.1), nilai persentasenya cenderung tidak mengalami perubahan yang signifikan pada keseluruhan tahun. Hal tersebut mengindikasikan bahwa kelompok spesies tersebut cenderung memiliki persebaran titik yang stabil pada keseluruhan waktu observasi.

e. Interpretasi untuk *Proportion of Variance (PV)*

Tabel 4.18 menunjukkan distribusi estimasi PV pada beberapa interval untuk 100 spesies yang diobservasi pada 8 tahun berbeda. Pada keseluruhan tahun,

mayoritas nilai PV berada pada rentang 0.8 hingga 1, sehingga dapat disimpulkan bahwa interaksi antar spesies lebih berpengaruh dominan daripada interaksi akibat faktor internal setiap spesies.

Tabel 4. 18 Distribusi Estimasi **PV** (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		1981	1985	1990	1995
< 0.5		3	2	1	2
0.5	0.6	1	1	2	1
0.6	0.7	1	1	5	2
0.7	0.8	4	6	2	3
0.8	0.9	7	6	6	9
0.9	1.0	84	84	84	82

Tabel 4. 18 Lanjutan Distribusi Estimasi PV (Dalam Persen) untuk 100 Spesies dengan Waktu Observasi 8

Interval		2000	2005	2010	2015
< 0.5		3	3	6	4
0.5	0.6	3	2	2	0
0.6	0.7	10	3	1	4
0.7	0.8	1	5	4	7
0.8	0.9	6	6	10	4
0.9	1.0	77	81	77	81

f. Korelasi antar Spesies dan Waktu

Penghitungan korelasi menggunakan persamaan (2.27) dan persamaan (2.28) juga diterapkan untuk menganalisis 100 spesies yang diobservasi pada 8 waktu berbeda. Distribusi estimasi nilai korelasi antar spesies pada setiap koordinat u dari tahun 1981 hingga 2015 ditunjukkan pada Tabel 4.19. Distribusi korelasi cenderung tidak mengalami perubahan yang signifikan pada keseluruhan tahun, sehingga dapat dikatakan bahwa korelasi antar waktu observasi sangat kuat. Pada keseluruhan tahun, mayoritas spesies memiliki estimasi nilai korelasi diantara -0.3 hingga 0.3 sehingga banyak spesies yang berkorelasi kurang kuat pada setiap

koordinat u . Proporsi jumlah spesies yang saling berkorelasi kuat positif (lebih dari 0.6) dan berkorelasi kuat negatif (kurang dari -0.6) adalah yang terkecil dibandingkan dengan interval lainnya. Selain itu, jumlah spesies yang berkorelasi positif dan negatif cenderung berimbang.

Tabel 4. 19 Distribusi (Dalam Persen) Estimasi Nilai Korelasi Antar 100 Spesies dengan Waktu Observasi 8

Interval		1981	1985	1990	1995
-1	-0.6	9.33	9.37	9.29	8.14
-0.6	-0.3	17.13	17.25	17.39	17.9
-0.3	0	23.56	23.39	23.33	23.98
0	0.3	23.7	23.86	23.47	24.57
0.3	0.6	17.47	17.58	18.06	18.04
0.6	1	8.81	8.55	8.44	7.37

Tabel 4. 19 Lanjutan Distribusi (Dalam Persen) Estimasi Nilai Korelasi Antar 100 Spesies dengan Waktu Observasi 8

Interval		2000	2005	2010	2015
-1	-0.6	9.54	8.61	10.22	8.97
-0.6	-0.3	17.72	17.8	17.68	17.62
-0.3	0	22.77	23.62	22.12	23.43
0	0.3	23.15	24.12	22.69	23.82
0.3	0.6	18.04	17.82	18.16	17.92
0.6	1	8.79	8.04	9.13	8.24

(Halaman ini sengaja dikosongkan)

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis pada penelitian ini, maka diperoleh kesimpulan sebagai berikut.

1. Pelatihan model *probabilistic deep learning* pada studi simulasi dan studi terapan tidak menunjukkan terjadinya *overfitting* karena tren MSE setiap *epoch* antara data latih dan data validasi saling berimpit. Selain itu, secara umum pelatihan model pada studi simulasi dan studi terapan berjalan baik karena nilai MSE pada setiap *epoch* menunjukkan tren penurunan.
2. Evaluasi kebaikan model pada studi simulasi terdiri dari 2 pengukuran, yaitu pengukuran MSE untuk mengetahui kebaikan hasil estimasi $\hat{\Lambda}_{ij}^{(s)}$, dan pengukuran RMSE untuk mengetahui kebaikan hasil estimasi $\hat{\theta}_{sim}$. Model *probabilistic deep learning* mendapatkan nilai MSE dan *error* yang lebih kecil daripada model Mateu & Jalilian (2022). Hal tersebut mengindikasikan bahwa *probabilistic deep learning* lebih konsisten dalam memberikan estimasi $\hat{\Lambda}_{ij}^{(s)}$ yang lebih baik dan lebih *robust* dalam memberikan estimasi di berbagai variasi pola *point pattern*. Selain itu, model *probabilistic deep learning* juga memiliki waktu pelatihan yang lebih singkat daripada model Mateu & Jalilian (2022). Pada pengukuran RMSE, Choiruddin dkk (2020) mendapatkan nilai RMSE lebih kecil pada mayoritas parameter. Meskipun demikian, proses estimasi Choiruddin dkk (2020) membutuhkan waktu yang jauh lebih lama daripada metode berbasis *deep learning*. Sementara itu, perbandingan RMSE antara kedua metode berbasis *deep learning* menunjukkan bahwa *probabilistic deep learning* mendapatkan nilai RMSE yang lebih kecil pada parameter $\hat{\alpha}$, $\hat{\phi}$, dan $\hat{\psi}$. Hal tersebut mengindikasikan bahwa model *probabilistic deep learning* cenderung memberikan estimasi $\hat{\theta}_{sim}$ yang lebih baik daripada model Mateu & Jalilian (2022).

3. Analisis pada studi terapan dibagi menjadi 2 bagian, yaitu analisis untuk jumlah spesies 25 dengan waktu observasi 4, dan analisis untuk jumlah spesies 100 dengan waktu observasi 8. Keباikan hasil estimasi $\widehat{\Lambda}_{ij}^{(s,t)}$ dievaluasi menggunakan MSE pada kedua jenis analisis. Pada kedua jenis analisis, model *probabilistic deep learning* mendapatkan nilai MSE dan *error* yang lebih kecil daripada model Mateu & Jalilian (2022). Hal tersebut menandakan bahwa *probabilistic deep learning* mampu memberikan estimasi yang lebih baik dan lebih konsisten daripada model Mateu & Jalilian (2022) pada berbagai jumlah tipe *multivariate* dan waktu observasi. Selain itu, waktu pelatihan pada model *probabilistic deep learning* jauh lebih singkat daripada Mateu & Jalilian (2022). Perbedaan tersebut semakin signifikan ketika analisis dilakukan pada tipe *multivariate* berukuran sangat besar.

5.2 Saran

Saran yang diberikan pada penelitian selanjutnya adalah:

1. Studi simulasi pada penelitian ini tidak menggunakan unsur *temporal* dan variabel *covariate* sehingga tergolong sebagai *multivariate point pattern* stasioner. Penelitian selanjutnya dapat menggunakan studi simulasi yang mengandung variabel waktu dan *covariate* (*multivariate spatio-temporal point pattern* non-stasioner) yang terdiri dari puluhan hingga ratusan tipe *multivariate*. Tujuannya adalah untuk menguji performa model dalam memberikan estimasi parameter pada berbagai jumlah tipe *multivariate spatio-temporal point pattern* non-stasioner sehingga hasil studi simulasi semakin komprehensif.
2. Estimasi parameter $\widehat{\sigma}$ oleh *probabilistic deep learning* mendapatkan nilai RMSE yang cukup besar dan cenderung *underestimate*. Pada penelitian ini, $\widehat{\sigma}$ diestimasi sebagai bobot pada suatu *hidden layer* dengan nilai inisialisasi diambil secara *uniform* pada rentang 0 hingga 0.739. Dengan demikian, model sulit untuk memberikan estimasi $\widehat{\sigma}^2$ yang baik apabila nilai σ^2 sebenarnya bernilai besar. Penelitian selanjutnya dapat menggunakan rentang pengambilan

nilai inisialisasi bobot yang lebih besar sehingga model dapat memberikan estimasi $\hat{\sigma}^2$ yang lebih baik.

3. Parameter ϕ dan ψ diasumsikan mengikuti distribusi Normal sehingga model perlu memberikan estimasi untuk parameter lokasi dan skala untuk estimasi $\hat{\phi}$ dan $\hat{\psi}$. Penelitian ini menggunakan fungsi aktivasi *softplus* untuk estimasi parameter skala karena fungsi aktivasi ini memiliki domain $(0, \infty)$. Namun demikian, evaluasi pada studi simulasi menunjukkan bahwa estimasi $\hat{\phi}$ dan $\hat{\psi}$ mendapatkan RMSE yang cukup besar. Penelitian berikutnya dapat menggunakan fungsi aktivasi Rectified Linear Unit (ReLU) untuk estimasi parameter skala karena fungsi aktivasi ReLU memiliki komputasi yang lebih efisien dan memiliki performa yang bagus pada kasus-kasus *deep learning* kompleks daripada fungsi aktivasi *softplus* (Nair & Hinton, 2010).

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

- Aggarwal, C. C. (2019). *Neural Networks and Deep Learning*. Springer Cham.
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., . . . Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*.
- Amrit, C., Paauw, T., Aly, R., & Lavric, M. (2017). Identifying child abuse through text mining and machine learning. *Expert Systems with Applications*, 402-418.
- Baddeley, A. (2007). Spatial Point Processes and their Applications. Dalam *Stochastic Geometry: Lectures given at the C.I.M.E. Summer School held in Martina Franca, Italy, September 13-18, 2004* (hal. 1-75). Berlin: Springer Berlin Heidelberg.
- Baddeley, A., Rubak, E., & Turner, R. (2015). *Spatial Point Patterns Methodology and Applications with R*. New York: Chapman and Hall/CRC.
- Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 859–877.
- Chang, D. T. (2021). Probabilistic Deep Learning with Probabilistic Neural Networks and Deep Probabilistic Models. *arXiv*.
- Choiruddin, A., Cuevas-Pacheco, F., Coeurjolly, J.-F., & Waagepetersen, R. (2020). Regularized estimation for highly multivariate log Gaussian Cox processes. *Statistics and Computing*, 649–662.
- Choiruddin, A., Susanto, T. Y., & Metrikasari, R. (2021). Two-step estimation for modeling the earthquake occurrences in Sumatra by Neyman–Scott Cox point processes. *Soft Computing in Data Science: 6th International Conference, SCDS 2021*, (hal. 146-159).
- Chollet, F. (2015). Keras. Diambil kembali dari <https://github.com/fchollet/keras>
- Condit, R., Pérez, R., Aguilar, S., Lao, S., Foster, R., & Hubbell, S. (2019). Complete data from the Barro Colorado 50-ha plot: 423617 trees, 35 years. *Dryad*.
- Cox, D. R. (1955). Some Statistical Methods Connected with Series of Events. *Journal of the Royal Statistical Society. Series B (Methodological)*, 129-164.
- Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Najada, H. A. (2015). Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 23.
- Cressie, N., & Moores, M. T. (2021). Spatial Statistics. *arXiv*.

- Deldjoo, Y., Elahi, M., Cremonesi, P., Garzotto, F., Piazzolla, P., & Quadrana, M. (2016). Content-Based Video Recommendation System Based on Stylistic Visual Features. *Journal on Data Semantics*, 99-113.
- Doersch, C. (2016). Tutorial on Variational Autoencoders. *arXiv*.
- Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2000). Incorporating Second-Order Functional Knowledge for Better Option Pricing. *Advances in Neural Information Processing Systems*. MIT Press.
- Dürr, O., Sick, B., & Murina, E. (2020). *Probabilistic Deep Learning With Python, Keras and TensorFlow Probability*. New York: Manning Publications.
- Eckardt, M., González, J. A., & Mateu, J. (2021). Graphical modelling and partial characteristics for multitype and multivariate-marked spatio-temporal point processes. *Computational Statistics & Data Analysis*, 107139.
- Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 115-118.
- Germain, M., Gregor, K., Murray, I., & Larochelle, H. (2015). MADE: Masked Autoencoder for Distribution Estimation. *Proceedings of the 32nd International Conference on Machine Learning*, (hal. 881-889).
- Ghojogh, B., & Crowley, M. (2019). The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. *ArXiv*.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Guan, Y., Jalilian, A., & Waagepetersen, R. (2015). Quasi-Likelihood for Spatial Point Processes. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 77(3), 677–697.
- Hessellund, K. B., Xu, G., Guan, Y., & Waagepetersen, R. (2022). Second-order semi-parametric inference for multivariate log Gaussian Cox processes. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*.
- Hinton, G., Srivastava, N., & Swersky, K. (2017). *Neural Networks for Machine Learning: Lecture 6e*.
- Hodson, T. O., Over, T. M., & Foks, S. S. (2021). Mean Squared Error, Deconstructed. *Journal of Advances in Modeling Earth Systems*, 13.
- Husain, A., & Choiruddin, A. (2021). Poisson and logistic regressions for inhomogeneous multivariate point processes: a case study in the Barro Colorado Island plot. *Soft Computing in Data Science: 6th International Conference, SCDS 2021*, (hal. 301-311).
- Jun, M., Schumacher, C., & Saravanan, R. (2019). Global multivariate point pattern models for rain type occurrence. *Spatial Statistics vol. 31*, 100355.

- Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1×1 Convolutions. *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*. Montreal.
- Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *arXiv*.
- Kusumadewi, S. (2004). *Membangun Jaringan Saraf Tiruan Menggunakan MATLAB & EXCEL LINK*. Yogyakarta: Graha Ilmu.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 436-444.
- Lederer, J. (2021). Activation Functions in Artificial Neural Networks: A Systematic Overview. *arXiv*. doi:<https://doi.org/10.48550/arXiv.2101.09957>
- Lenzi, A., Bessac, J., Rudi, J., & Stein, M. L. (2023). Neural networks for parameter estimation in intractable models. *Computational Statistics & Data Analysis*, 107762.
- Lim, Z.-Y., Ong, L.-Y., & Leow, M.-C. (2021). A Review on Clustering Techniques: Creating Better User Experience for Online Roadshow. *Future Internet*, 233.
- Møller, J., & Waagepetersen, R. P. (2007). Modern Statistics for Spatial Point Processes. *21st Nordic Conference on Mathematical Statistics, Rebild, Denmark, June 2006 (NordStat 2006)*.
- Nagpal, K., Foote, D., & Liu, Y. (2019). Development and validation of a deep learning algorithm for improving Gleason scoring of prostate cancer. *npj Digital Medicine*, 48.
- Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on International Conference on Machine Learning* (hal. 807–814). Madison: Omnipress.
- Oord, A. v., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., . . . Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *Arxiv*.
- Rajala, T., Murrell, D. J., & Olhede, S. C. (2018). Detecting multivariate interactions in spatial point patterns with Gibbs models and variable selection. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 1237–1273.
- Ripley, B. D. (1977). Modelling Spatial Patterns. *Journal of the Royal Statistical Society. Series B (Methodological)*, 172-212.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 533–536.
- Salimans, T., Karpathy, A., Chen, X., & Kingma, D. P. (2017). Pixel{CNN}++: Improving the Pixel{CNN} with Discretized Logistic Mixture Likelihood and Other Modifications. *International Conference on Learning Representations*.
- Singh, A., & Ogunfunmi, T. (2022). An Overview of Variational Autoencoders for Source Separation, Finance, and Bio-Signal Applications. *Entropy*, 55.

- Singh, J., & Banerjee, R. (2019). A Study on Single and Multi-layer Perceptron Neural Network. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, (hal. 35-40).
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 1929–1958.
- Syahfitra, F. D., Syahputra, R., & Putra, K. T. (2017). Implementation of Backpropagation Artificial Neural Network as a Forecasting System of Power Transformer Peak Load at Bumiayu Substation. *Journal of Electrical Technology UMY*, 118-125.
- Tama, B. A., Vania, M., Lee, S., & Lim, S. (2023). Recent advances in the application of deep learning for fault diagnosis of rotating machinery using vibration signals. *Artificial Intelligence Review*, 4667–4709.
- Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent Advances in Stochastic Gradient Descent in Deep Learning. *Mathematics*, 682.
- Uzair, M., & Jamil, N. (2020). Effects of Hidden Layers on the Efficiency of Neural networks. *2020 IEEE 23rd International Multitopic Conference (INMIC)*, (hal. 1-6).
- Vakili, M., Ghamsari, M., & Rezaei, M. (2020). Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification. *arXiv*.
- Valente, F., & Laurini, M. (2023). A spatio-temporal analysis of fire occurrence patterns in the Brazilian Amazon. *Scientific Reports vol. 13*, 12727.
- Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 236–244.
- Zhang, S., Zhai, J., Chen, J., & He, Q. (2019). Autoencoder and its various variants. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Miyazaki: IEEE.
- Zhao, K., He, T., Wu, S., Wang, S., Dai, B., Yang, Q., & Lei, Y. (2018). Application research of image recognition technology based on CNN in image location of environmental monitoring UAV. *EURASIP Journal on Image and Video Processing*, 150.

LAMPIRAN

Lampiran 1 Model untuk Studi Simulasi

No.	Jenis Layer	Nama Layer	Dimensi Output	Fungsi Aktivasi	Layer Sebelumnya
1	Input	xi	$m = 10$	-	-
2	Input	ui	2	-	-
3	Input	di	1	-	-
4	Custom	udist	50 × 50	-	ui
5	Dense	h1	64	tanh	xi
6	Dense	h2	64	tanh	udist
7	Dense	h_udist	m	tanh	h2
8	Dense	h_udist2	$a = 4$	tanh	h2
9	Dense	h4	128	relu	h1
10	Dropout	dr1	128	-	h4
11	Dense	h5	128	relu	h1
12	Dropout	dr2	128	-	h5
13	Dense	mu_branch	64	tanh	dr1
14	Dense	par_branch	64	tanh	dr2
15	Dropout	dr_par_branch	64	-	par_branch
16	Dense	par_branch_2	64	tanh	dr2
17	Dropout	dr_par_branch_2	64	-	par_branch_2
18	Dense	mu_params	m	exp	mu_branch
19	Probabilistic Layer	mu_dist	m	-	mu_params
20	Dense	phi_params	$a \times 2$	sigmoid, softplus	dr_par_branch
21	Probabilistic Layer	phi_dist	a	-	phi_params
22	Dense	psi_params	$m \times 2$	sigmoid, softplus	dr_par_branch_2
23	Probabilistic Layer	psi_dist	m	-	psi_params
24	Multiply	pre_Emat	a	-	h_udist2, phi_dist
25	Dense	Emat	$a \times$ 50×50	-	pre_Emat

Lampiran 1 Lanjutan Model untuk Studi Simulasi

No.	Jenis Layer	Nama Layer	Dimensi Output	Fungsi Aktivasi	Layer Sebelumnya
26	Multiply	pre_Vmat	m	-	h_udist, psi_dist
27	Dense	Vmat	$m \times 50 \times 50$	-	pre_Vmat
28	Probabilistic Layer	e_dist	a	-	Emat
29	Dense	Elin	m	-	e_dist
30	Probabilistic Layer	v_dist	m	-	Vmat
31	Dense	sigma_h	m	-	di
32	Multiply	sigma_v_hat	m	-	sigma_h, v_dist
33	Add	lin	m	-	Elin, sigma_v_hat
34	Custom	exp_lin	m	exp	lin
35	Multiply	xihat	m	-	mu_dist, exp_lin

Berdasarkan arsitektur pada Lampiran 1, *layer* terakhir adalah “xihat” (baris 35 pada Lampiran 1) yang merepresentasikan nilai $\hat{\Lambda}_{ij}^{(s)}$, yaitu estimasi jumlah titik pada *grid* B_{ij} dengan tipe *multivariate* ke- s . Layer “udist” merupakan *layer* untuk menghitung jarak Euclid antar titik $\mathbf{d}(u, u')$, sehingga keluaran dari *layer* tersebut adalah matriks berukuran 50×50 . Jarak antar titik $\mathbf{d}(u, u')$ direpresentasikan dalam sebuah *hidden layer* dengan persamaan:

$$\begin{aligned} \hat{\mathbf{d}}(u, u')^{(l)} &= f_{\tanh}(\mathbf{b}_7 + \mathbf{w}_7^T \times \mathbf{d}(u, u')) \\ \hat{\mathbf{d}}(u, u')^{(s)} &= f_{\tanh}(\mathbf{b}_8 + \mathbf{w}_8^T \times \mathbf{d}(u, u')) \end{aligned} \quad (1)$$

dimana \mathbf{w}_7 dan \mathbf{w}_8 merupakan matriks bobot berukuran $50 \times a$ dan $50 \times m$, \mathbf{b}_7 dan \mathbf{b}_8 merupakan vektor bias berukuran a dan m , sedangkan $\mathbf{d}(u, u')$ adalah jarak Euclid antar titik berukuran 50×50 . Persamaan (3.9) menunjukkan bahwa $\mu_{ij}^{(s)} \sim \text{Poisson}(\lambda_{ij}^{(s)})$. Layer “mu_params” pada *layer* ke-18 mencerminkan nilai estimasi

parameter $\hat{\lambda} = [\hat{\lambda}_{ij}^{(s)}]$ untuk parameter $\hat{\mu} = [\hat{\mu}_{ij}^{(s)}]$. Distribusi Poisson membutuhkan parameter $\hat{\lambda}$ yang bernilai positif, sehingga diterapkan fungsi aktivasi eksponensial terhadap nilai keluaran dari “mu_params”. Dengan kata lain, nilai $\hat{\lambda}$ didapatkan menggunakan persamaan:

$$\hat{\lambda} = f_{\text{exp}} \left(\mathbf{b}_{18} + \mathbf{w}_{18}^T \times \hat{\mathbf{h}}^{(13)} \right) \quad (2)$$

dimana $\mathcal{L}_{13} = 64$ merupakan jumlah *neuron* pada *layer* ke-13 (lihat baris 13 kolom 4 pada Lampiran 1), \mathbf{b}_{18} merupakan vektor bias berukuran m , $\hat{\mathbf{h}}^{(13)}$ adalah matriks keluaran dari *layer* ke-13 berukuran 64×50 , dan \mathbf{w}_{18} adalah matriks bobot berukuran $64 \times m$. Nilai $\hat{\mu}_{ij}^{(s)}$ diasumsikan merupakan ekspektasi dari distribusi Poisson, sehingga:

$$\mathbb{E} \left(\hat{\mu}_{ij}^{(s)} \right) = \hat{\lambda}_{ij}^{(s)} \quad (3)$$

Parameter $\boldsymbol{\phi} = [\phi_{ij}^{(l)}]$ dan $\boldsymbol{\psi} = [\psi_{ij}^{(s)}]$ diasumsikan mengikuti distribusi Normal dengan parameter:

$$\boldsymbol{\mu}_{\phi} = [\mu_{\phi_{ij}}^{(l)}], (\boldsymbol{\sigma}_{\phi})^2 = [(\sigma_{\phi_{ij}}^{(l)})^2]$$

$$\boldsymbol{\mu}_{\psi} = [\mu_{\psi_{ij}}^{(s)}], (\boldsymbol{\sigma}_{\psi})^2 = [(\sigma_{\psi_{ij}}^{(s)})^2]$$

Oleh sebab itu, *layer* “phi_params” dan “psi_params” (baris 20 dan 22 pada Lampiran 1) memiliki dimensi keluaran $a \times 2$ dan $m \times 2$ yang mencerminkan nilai estimasi parameter untuk setiap distribusi Normal. Nilai estimasi $\boldsymbol{\mu}_{\phi}$ dan $(\boldsymbol{\sigma}_{\phi})^2$ didapatkan melalui persamaan:

$$\hat{\boldsymbol{\mu}}_{\phi} = f_{\text{sig}} \left(\mathbf{b}_{20}^{(1)} + (\mathbf{w}_{20}^{(1)})^T \times \hat{\mathbf{h}}^{(15)} \right) \quad (4)$$

$$(\hat{\sigma}_\phi)^2 = f_{\text{splus}} \left(\mathbf{b}_{20}^{(2)} + \left(\mathbf{w}_{20}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(15)} \right)$$

dimana $\mathcal{L}_{15} = 64$ merupakan jumlah *neuron* pada *layer* ke-15 (lihat baris 15 kolom 4 pada Lampiran 1), $\mathbf{b}_{20}^{(1)}$ dan $\mathbf{b}_{20}^{(2)}$ merupakan vektor bias berukuran a , $\hat{\mathbf{h}}^{(15)}$ adalah matriks keluaran dari *layer* ke-15 berukuran 64×50 , sedangkan $\mathbf{w}_{20}^{(1)}$ dan $\mathbf{w}_{20}^{(2)}$ adalah matriks bobot berukuran $64 \times a$. Dengan demikian, $\hat{\boldsymbol{\mu}}_\phi$ dan $(\hat{\sigma}_\phi)^2$ adalah matriks berukuran $a \times 50$. Sementara itu, nilai estimasi $\boldsymbol{\mu}_\psi$ dan $(\sigma_\psi)^2$ didapatkan melalui persamaan:

$$\hat{\boldsymbol{\mu}}_\psi = f_{\text{sig}} \left(\mathbf{b}_{22}^{(1)} + \left(\mathbf{w}_{22}^{(1)} \right)^T \times \hat{\mathbf{h}}^{(17)} \right) \quad (5)$$

$$(\hat{\sigma}_\psi)^2 = f_{\text{splus}} \left(\mathbf{b}_{22}^{(2)} + \left(\mathbf{w}_{22}^{(2)} \right)^T \times \hat{\mathbf{h}}^{(17)} \right) \quad (6)$$

dimana $\mathcal{L}_{17} = 64$ merupakan jumlah *neuron* pada *layer* ke-17 (lihat baris 17 kolom 4 pada Lampiran 1), $\mathbf{b}_{22}^{(1)}$ dan $\mathbf{b}_{22}^{(2)}$ merupakan vektor bias berukuran m , $\hat{\mathbf{h}}^{(17)}$ adalah matriks keluaran dari *layer* ke-17 berukuran 64×50 , sedangkan $\mathbf{w}_{22}^{(1)}$ dan $\mathbf{w}_{22}^{(2)}$ merupakan matriks bobot berukuran $64 \times m$. Dengan demikian, $\hat{\boldsymbol{\mu}}_\psi$ dan $(\hat{\sigma}_\psi)^2$ adalah matriks berukuran $m \times 50$.

Nilai estimasi parameter $\hat{\boldsymbol{\phi}}$ dan $\hat{\boldsymbol{\psi}}$ didapatkan melalui proses *sampling* menggunakan *reparameterization trick* (Kingma & Welling, 2014) yang ditunjukkan pada persamaan:

$$\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\mu}}_\phi + \hat{\sigma}_\phi \odot \varepsilon_\phi \quad (7)$$

$$\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{\mu}}_\psi + \hat{\sigma}_\psi \odot \varepsilon_\psi \quad (8)$$

dimana \odot merupakan operasi perkalian elemen demi elemen (*element-wise multiplication*), sedangkan $\varepsilon_\phi \sim \text{Normal}(0,1)$ dan $\varepsilon_\psi \sim \text{Normal}(0,1)$.

Pada penelitian Mateu & Jalilian (2022), Gaussian *random field* \mathbf{E} dan \mathbf{V} didapatkan dengan membangkitkan distribusi *multivariate normal* berdimensi ukuran *batch* b' dengan *mean* $\mathbf{0}$. Suatu distribusi *multivariate normal* dapat

dibangkitkan berdasarkan nilai *mean* dan hasil Cholesky *decomposition* terhadap matriks *covariance*-nya. Dengan demikian, untuk mendapatkan Gaussian *random field* \mathbf{E} dan \mathbf{V} , maka pembentukan distribusi *multivariate normal* didasarkan pada nilai *mean* dan:

$$\mathbf{\Sigma}_E = \mathbf{A}\mathbf{A}^T \quad (9)$$

$$\mathbf{\Sigma}_V = \mathbf{B}\mathbf{B}^T \quad (10)$$

dimana $\mathbf{\Sigma}_E$ dan $\mathbf{\Sigma}_V$ merupakan matriks *covariance* untuk \mathbf{E} dan \mathbf{V} , sedangkan \mathbf{A} dan \mathbf{B} adalah matriks *lower triangular* yang didapatkan dari proses Cholesky *decomposition* terhadap kedua matriks tersebut (Mateu & Jalilian, 2022).

Model *probabilistic deep learning* memberikan estimasi matriks *lower triangular* \mathbf{A} dan \mathbf{B} pada *hidden layer*-nya, yaitu *layer* “Emat” dan “Vmat” (baris 25 dan 27 pada Lampiran 1). *Layer* “Emat” mengeluarkan estimasi matriks \mathbf{A} berdasarkan input dari *layer* “pre_Emat” (baris 24 pada Lampiran 1) yang berisi representasi kalkulasi jarak Euclid antar titik $\widehat{\mathbf{d}}(u, u')^{(l)}$ dan estimasi $\widehat{\boldsymbol{\phi}}$:

$$\begin{aligned} \widehat{\mathbf{h}}^{(25)} &= \left(\mathbf{w}_{25} (\widehat{\boldsymbol{\phi}} \times \widehat{\mathbf{d}}(u, u')^{(l)}) \right) \\ \widehat{\mathbf{h}}^{(25)} &= \left[\widehat{h}_{i'j'}^{(25)} \right] = \begin{cases} \left| \widehat{h}_{i'j'}^{(25)} \right| & \text{jika } i' = j' \\ \widehat{h}_{i'j'}^{(25)} & \text{jika } i' \neq j' \end{cases} \\ \mathbf{A} &= \begin{cases} \widehat{h}_{i'j'}^{(25)} & \text{jika } i' \geq j' \\ 0 & \text{jika } i' < j' \end{cases} \end{aligned} \quad (11)$$

dimana $i' = 1, 2, \dots, 50$, $j' = 1, 2, \dots, 50$, $\widehat{\mathbf{h}}^{(25)}$ merupakan matriks output *hidden layer* berukuran $a \times 50 \times 50$, \mathbf{w}_{25} merupakan matriks bobot berukuran $a \times 50 \times a$ pada *layer* ke-25, dan $\widehat{\mathbf{d}}(u, u')^{(l)}$ adalah representasi dari jarak antar titik (Persamaan 1). Semua elemen diagonal pada matriks $\widehat{\mathbf{h}}^{(25)}$ diubah ke bentuk positif dan semua elemen *upper diagonal* diubah ke 0 untuk mendapatkan *lower triangular* matriks \mathbf{A} . Dengan demikian, matriks \mathbf{A} dibentuk berdasarkan variabel penyusun $\mathbf{\Sigma}_E$ pada persamaan (2.16).

Proses serupa juga dilakukan pada *layer* “Vmat” (baris 27 pada Lampiran 1). Sesuai dengan persamaan (2.14) dan persamaan (2.21), *layer* “Vmat” menerima input dari *layer* “pre_Vmat” (baris 26 pada Lampiran 1) yang mengandung informasi kalkulasi jarak Euclid antar titik $\mathbf{d}(u, u')$ dan estimasi $\hat{\boldsymbol{\psi}}$:

$$\begin{aligned}\hat{\mathbf{h}}^{(27)} &= \left(\mathbf{w}_{27} (\hat{\boldsymbol{\psi}} \times \hat{\mathbf{d}}(u, u')^{(s)}) \right) \\ \hat{\mathbf{h}}^{(27)} = [\hat{h}_{i'j'}^{(27)}] &= \begin{cases} |\hat{h}_{i'j'}^{(27)}| & \text{jika } i' = j' \\ \hat{h}_{i'j'}^{(27)} & \text{jika } i' \neq j' \end{cases} \\ \mathbf{B} &= \begin{cases} \hat{h}_{i'j'}^{(27)} & \text{jika } i' \geq j' \\ 0 & \text{jika } i' < j' \end{cases}\end{aligned}\quad (12)$$

dimana $\hat{\mathbf{h}}^{(27)}$ merupakan matriks output *hidden layer* berukuran $m \times 50 \times 50$, \mathbf{w}_{27} merupakan matriks bobot berukuran $m \times 50 \times m$, dan $\hat{\mathbf{d}}(u, u')^{(s)}$ adalah representasi dari jarak antar titik (Persamaan 4.1). Semua elemen diagonal pada matriks $\hat{\mathbf{h}}^{(27)}$ diubah ke bentuk positif dan semua elemen *upper diagonal* diubah ke 0 untuk mendapatkan *lower triangular* matriks \mathbf{B} .

Dengan kata lain, meskipun \mathbf{A} dan \mathbf{B} dihitung menggunakan *neural network*, penghitungan estimasi kedua matriks *lower triangular* tersebut tetap dipengaruhi oleh variabel-variabel pada persamaan parametrik. Nilai estimasi $\hat{\mathbf{E}}_{ij}^{(l)}$ dan $\hat{\mathbf{V}}_{ij}^{(s)}$ didapatkan melalui proses *reparameterization trick* pada persamaan:

$$\hat{\mathbf{E}}_{ij}^{(l)} = \mathbf{A} \odot \boldsymbol{\varepsilon}_E \quad (13)$$

$$\hat{\mathbf{V}}_{ij}^{(s)} = \mathbf{B} \odot \boldsymbol{\varepsilon}_V \quad (14)$$

dimana $\boldsymbol{\varepsilon}_E \sim \text{Multivariate Normal}(\mathbf{0}, \mathbf{I}_{a \times b'})$ sedangkan $\boldsymbol{\varepsilon}_V \sim \text{Multivariate Normal}(\mathbf{0}, \mathbf{I}_{m \times b'})$.

Persamaan *multivariate* LGCP pada persamaan (3.7) mengandung komponen:

$$\sum_{l=1}^a \alpha_{sl} \mathbf{E}_{ij}^{(l)}$$

yang dapat direpresentasikan sebagai proses kalkulasi pada sebuah *hidden layer* tanpa fungsi aktivasi. Oleh sebab itu, nilai estimasi $\hat{\alpha}$ dapat dimodelkan sebagai bobot pada *layer* “Elin” (baris 29 pada Lampiran 1) yang menerima input $\hat{\mathbf{E}}_{ij}^{(l)}$. Proses estimasi $\hat{\alpha}$ ditunjukkan pada persamaan:

$$\hat{\alpha} \hat{\mathbf{E}}_{ij}^{(s)} = \sum_{l=1}^a w_{29,l,s} \hat{\mathbf{E}}_{ij}^{(l)} \quad (15)$$

dimana \mathbf{w}_{29}^T merupakan matriks bobot berukuran $m \times a$ yang mencerminkan hasil estimasi $\hat{\alpha}$.

Parameter $\hat{\sigma}^{(s)}$ dimodelkan sebagai bobot karena parameter ini tidak tergantung kepada *grid* B_{ij} . Proses estimasi $\hat{\sigma}^{(s)}$ terjadi pada sebuah *hidden layer* tanpa fungsi aktivasi yang menerima input data berupa vektor kolom dengan semua elemen bernilai 1. Pada Lampiran 1, estimasi $\hat{\sigma}^{(s)}$ terjadi pada *layer* “sigma_h” (baris 31) yang menerima input berupa vektor kolom yang semua elemennya bernilai 1. Estimasi pada *layer* “sigma_h” ditunjukkan pada persamaan:

$$\begin{aligned} \hat{\sigma}^{(s)} &= \sum_{k'=1}^1 w_{31,k',s} 1 \\ &= w_{31,1,s} \times \mathbf{1} \end{aligned} \quad (16)$$

Dimana \mathbf{w}_{31} merupakan matriks bobot berukuran $1 \times m$. Sesuai dengan persamaan (3.7), hasil estimasi $\hat{\sigma}^{(s)}$ dikalikan dengan $\hat{\mathbf{V}}_{ij}^{(s)}$ seperti yang ditunjukkan pada persamaan:

$$\hat{\sigma} \hat{\mathbf{V}}_{ij}^{(s)} = \hat{\sigma}^{(s)} \times \hat{\mathbf{V}}_{ij}^{(s)} \quad (17)$$

Proses perkalian dilakukan pada *layer* “sigma_v_hat” (baris 32 pada Lampiran 1).

Hasil estimasi $\widehat{\Lambda}_{ij}^{(s)}$ didapatkan melalui proses rekonstruksi sesuai dengan persamaan (3.7), sehingga:

$$\widehat{\Lambda}_{ij}^{(s)} = \hat{\mu}_{ij}^{(s)} f_{\text{exp}}(\widehat{\alpha} \widehat{\mathbf{E}}_{ij}^{(s)} + \widehat{\sigma} \widehat{\mathbf{V}}_{ij}^{(s)}) \quad (18)$$

Fungsi eksponensial telah diterapkan kepada $\hat{\mu}_{ij}^{(s)}$ pada persamaan (2), sehingga fungsi eksponensial tidak diterapkan kembali pada parameter tersebut, seperti yang ditunjukkan pada Persamaan (18).

Lampiran 2 Kode untuk Pembangkitan Data *Point Pattern* dan Diskretisasi pada Studi Simulasi

```
setwd("D:/Master/Thesis/thesis_point_pattern/simulasi/discretization")
library(spatstat)
library(units)
require("RandomFields")

# true parameter
a = 4
m = 10
xrange = c(0,2)
yrange = c(0,1)
xdim = 1000
ydim = 500

vec_alpha = c(
  sqrt(0.5), 0.10, -1, 0,
  0, 0, -0.70, 1,
  0, -0.15, sqrt(0.5), 0.10,
  -1, 0, 0, 0,
  -0.70, 1, 0, -0.15,
  sqrt(0.5), 0.10, -1, 0,
  0, 0, -0.70, 1,
  0, -0.15, sqrt(0.5), 0.10,
  -1, 0, 0, 0,
  -0.70, 1, 0, -0.15
)
alpha = matrix(vec_alpha, nrow = m, ncol = a, byrow=T)
dim(alpha)

sigmas = c(1, 1, 1.5, 1, 0.2, 0.2, 1, 1.5, 1.5, 1.5)
phi = c(0.02, 0.03, 0.03, 0.05)
psi = c(0.01, 0.02, 0.02, 0.03, 0.04, 0.04, 0.05, 0.06, 0.06, 0.07)
n = rep(5000, 10)
print(n)
```

Lampiran 2 Lanjutan Kode untuk Pembangkitan Data *Point Pattern* dan Diskretisasi pada Studi Simulasi

```
# discretization
nx <- 200
ny <- 100
spChosen <- paste0("type_", seq(1, m)); spChosen

par(mfrow=c(1, 1), mar=c(0, 0, 1, 0))

# simulation
simulate.lmc2<-
function(alpha, sigmas, n, phi=c(0.05), psi=c(0.05), xrange=c(0,1), yrange=c(0
,1), xdim=100, ydim=100, fits=NULL, fields=F){

  xdiscr=(xrange[2]-xrange[1])/xdim
  ydiscr=(yrange[2]-yrange[1])/ydim

  xx=xrange[1]+xdiscr*(c(1:xdim)-0.5)
  yy=yrange[1]+ydiscr*(c(1:ydim)-0.5)
  griddx=length(xx)
  griddy=length(yy)

  nspecies=dim(alpha)[1]
  nlatent=dim(alpha)[2]

  loglambda=matrix(0, nspecies, griddx*griddy)

  for (j in 1:nlatent){
    Y1 = GaussRF(x=xx, y=yy, model="exponential",
grid=TRUE, param=c(mean=0, variance=1, nugget=0, scale=phi[j]))
    Y1=c(Y1)
    for (l in 1:nspecies)
      loglambda[l,]=loglambda[l,]+alpha[l,j]*Y1
  }

  for (l in 1:nspecies){
    U = GaussRF(x=xx, y=yy, model="exponential",
grid=TRUE, param=c(mean=0, variance=1, nugget=0, scale=psi[l]))
    loglambda[l,]=loglambda[l,]+sigmas[l]*c(U)
  }

  maxnpoints=nspecies*(sum(n)+10*sqrt(sum(n)))
  x0 <- y0 <- m0 <- double(maxnpoints)
  idx=0
  for (l in 1:nspecies){
    if (is.null(fits)){
      mu=log(n[l])-sum(alpha[l,]^2)/2-sigmas[l]^2/2
      Lamb <- as.im(list(x=xx,y=yy,z=matrix(exp(mu +
loglambda[l,]), ncol=griddy, byrow=F)), W=owin(xrange=xrange, yrange=yrange)
)
```

Lampiran 2 Lanjutan Kode untuk Pembangkitan Data *Point Pattern* dan Diskretisasi pada Studi Simulasi

```

    } else {
      xygrid=as.data.frame(expand.grid(xx,yy))
      names(xygrid)=c("x","y")
Lamb <-
as.im(list(x=xx,y=yy,z=matrix(predict.ppm(fits[[1]],locations=xygrid)*exp(
loglambda[1,]-sum(alpha[1,]^2)/2-
sigmas[1]^2/2),ncol=griddy,byrow=F)),W=owin(xrange=xrange,yrange=yrange)
)
    }

print(summary(Lamb))
X0 <- rpoispp(Lamb,W=owin(xrange=xrange,yrange=yrange))
if (idx+X0$n>maxnpoints){
  x0=c(x0,double(X0$n))
  y0=c(y0,double(X0$n))
  m0=c(m0,double(X0$n))
}
x0[(idx+1):(idx+X0$n)] <- X0$x
y0[(idx+1):(idx+X0$n)] <- X0$y
m0[(idx+1):(idx+X0$n)] <- rep(1, X0$n)
idx=idx+X0$n
}
print(c(nspecies,nlatent,phi))
X <- ppp(x0[1:idx], y0[1:idx],
window=owin(xrange=xrange,yrange=yrange), marks=as.factor(m0[1:idx]))
if (fields)
  return(list(X=X,field=Y1))
else
  return(X)
}

dfun <- function(t, ppp_split)
{
  df <- data.frame(time=rep(t, nx * ny), loc=1:(nx * ny))
  for (s in 1:m)
  {
    X <- ppp_split[[s]]
    df <- data.frame(df, c(quadratcount(X, nx=nx, ny=ny)))
  }
  names(df) <- c("time", "loc", spChosen)
  return(df)
}

for(i in 1:120) {
  set.seed(i)
  ppp = simulate.lmc2(alpha=alpha, sigmas=sigmas, n, phi=phi, psi=psi,
                    xrange=xrange, yrange=yrange, xdim=xdim,
ydim=ydim,
                    fits=NULL, fields=F)
  ppp_split = split(ppp)

dfout <- rbind.data.frame(dfun(1, ppp_split))

```

Lampiran 2 Lanjutan Kode untuk Pembangkitan Data *Point Pattern* dan Diskretisasi pada Studi Simulasi

```
write.csv(dfout, file=paste0("dataset/simulasi_", i, ".csv"),
          row.names=FALSE, quote=FALSE)
saveRDS(ppp, file = paste0("point_pattern/point_pattern_", i, ".rds"))
print(i)
}
```

Lampiran 3 Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Simulasi

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import os
import random

is_training_new = True
trial_type = 'trial_4h_v6.1'
save_path = f'result/{trial_type}'

try:
    metric_rec_df = pd.read_csv(f'{save_path}/mse_{trial_type}.csv')
except FileNotFoundError:
    metric_rec_df = pd.DataFrame({
        'training_time': [],
        'mse_train': [],
        'mse_test': [],
        'mse_all': [],
        'acc_train': [],
        'acc_test': [],
        'acc_all': [],
    })

metric_rec_df.tail()
sim_number = metric_rec_df.shape[0] + 1

sim_data =
pd.read_csv(f"discretization/dataset/simulasi_{sim_number}.csv")
coord_data = pd.read_csv("discretization/dataset/coord.csv")

# slicing dataset
n_species = 10
n_time = 1
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
sim_data = sim_data.iloc[:, :2 + n_species]
sim_data = sim_data.loc[sim_data['time'] <= n_time]

species_names = sim_data.columns[2:].tolist()
len(species_names)

# number of observations: grid cells
n_obs = sim_data.shape[0]

# number of species
m_species = len(species_names)
# dimension of discretization grid
grid_dimyx = (100, 200)

# grid coordinates of the cells
cell_centers = coord_data.iloc[:, [1, 2]]
plt.scatter(cell_centers.iloc[:, 0], cell_centers.iloc[:, 1], s=0.05)
plt.show()
plt.clf()

# split to training and test samples
np.random.seed(8)
test_idx = np.random.choice(n_obs, size=int(0.3 * n_obs), replace=False)

plt.scatter(cell_centers.iloc[sim_data.iloc[test_idx, 1] - 1, 0],
            cell_centers.iloc[sim_data.iloc[test_idx, 1] - 1, 1], s=0.01)
plt.show()
plt.clf()

# split spatio-temporal counts
sim_train = np.delete(sim_data.values, test_idx, axis=0)
sim_test = sim_data.iloc[test_idx, :].values

# split grid cells
cell_centers_train = cell_centers.iloc[sim_train[:, 1] - 1, :]
cell_centers_test = cell_centers.iloc[sim_test[:, 1] - 1, :]

dummy_train = np.ones(sim_train.shape[0])
dummy_test = np.ones(sim_test.shape[0])

sim_train = np.delete(sim_train, [0, 1], axis=1)
sim_test = np.delete(sim_test, [0, 1], axis=1)
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
# # Constructing the Model
from tensorflow.keras.layers import Input, Lambda, Dense, add,
concatenate, BatchNormalization, Dropout, Concatenate, Multiply
from tensorflow.keras.models import Model
from tensorflow.keras.constraints import Constraint, NonNeg
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K
from tensorflow.keras import layers, optimizers
import tensorflow_probability as tfp
import tensorflow as tf
import time
from tensorflow.python.framework.ops import disable_eager_execution,
enable_eager_execution

# encoder network parameters
batch_size = 200
latent_dim = 4 #
epochs = 50

### Input layers

# spatio-temporal counts and regional covariates
xi = Input(batch_shape=(batch_size, m_species), name="counts")
# spatial coordinates of cells
ui = Input(batch_shape=(batch_size, 2), name="coordinates")
# dummy input
di = Input(batch_shape=(batch_size, 1), name="dummy_input")

# Network architecture: from observations to latent field parameters

import tensorflow as tf
from tensorflow.keras.layers import Layer

class CovarianceMatrixPredictionLayer(Layer):
    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        self.jitter = 1e-5
        super(CovarianceMatrixPredictionLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        # Assuming input_shape is (batch_size, input_dim)
        input_dim = input_shape[-1]
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
batch_size = input_shape[0]
    self.lower_triangular_weights =
self.add_weight(name='lower_triangular_weights',
    shape=(self.output_dim, batch_size, self.output_dim),
    initializer='glorot_uniform',
    trainable=True)

    super(CovarianceMatrixPredictionLayer, self).build(input_shape)

def call(self, x):
    # Assuming x has shape (batch_size, input_dim)
    batch_size = tf.shape(x)[0]
    input_dim = tf.shape(x)[-1]

    # Use a dense layer to predict the lower triangular part
    lower_triangular = tf.matmul(self.lower_triangular_weights,
tf.reshape(x, [-1, batch_size]))

    # Create a new matrix with absolute diagonal values
    lower_triangular = tf.linalg.set_diag(lower_triangular,
tf.abs(tf.linalg.diag_part(lower_triangular)))

    # Precompute the lower triangular mask
    mask = tf.linalg.band_part(tf.ones([batch_size, batch_size]), -
1, 0)

    # Enforce the lower triangular structure
    lower_triangular *= mask

    return lower_triangular

def compute_output_shape(self, input_shape):
    return (input_shape[0], self.output_dim, self.output_dim)

# convert cell coordinates to pairwise spatial distances
def udistfun(coord_data):
    # computes the pairwise distances between points
    dist2 = K.sum(tf.square(coord_data[None, :] - coord_data[:, None]),
axis=-1)
    return tf.sqrt(dist2)

udist = Lambda(udistfun, name="spatialdist")(ui)
```


Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
def spatialcorrf2(r):
    out = tf.exp(-r) # Exponential correlation function
    return out

def normalize_dist(args):
    val, mean, std = args
    return (val - mean) / std

def concat_layers(args):
    return K.concatenate(args, axis=1)

def concat_sigma(args):
    return K.mean(args, axis=1)

tfd = tfp.distributions

def mu_poisson(params):
    rate = params
    return tfd.Poisson(rate=rate)

def mu_poisson_2(params):
    rate = tf.math.exp(params)
    return tfd.Poisson(rate=rate)

def mu_zero_inf(out, n):
    un_rate, un_s = tf.split(out, num_or_size_splits=n, axis=-1)

    rate = tf.squeeze(tf.math.exp(un_rate))
    s = tf.math.sigmoid(un_s)
    probs = tf.stack([1-s, s], -1)

    mix = tfd.Mixture(
        cat=tfd.Categorical(probs=probs),
        components=[
            tfd.Deterministic(loc=tf.zeros_like(rate)),
            tfd.Poisson(rate=rate)
        ])
    return mix

def phi_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.nn.sigmoid(un_loc)
    scale = 1e-3 + tf.math.softplus(0.05 * un_scale)
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
return tfd.Normal(loc=loc, scale=scale)

def psi_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.nn.sigmoid(un_loc)
    scale = 1e-3 + tf.math.softplus(0.05 * un_scale)
    return tfd.Normal(loc=loc, scale=scale)

def E_normal(params):
    return tfd.MultivariateNormalTril(loc = 0, scale_tril = params)

def V_normal(params):
    return tfd.MultivariateNormalTril(loc = 0, scale_tril = params)

def to_tensor_param_tanh(s):
    mean_s = s.sample()
    mean_s = tf.clip_by_value(mean_s, clip_value_min=-1,
clip_value_max=1)
    return mean_s

def to_tensor_param_sigmoid(s):
    mean_s = s.sample()
    mean_s = tf.clip_by_value(mean_s, clip_value_min=0,
clip_value_max=1)
    return mean_s

def to_tensor_param_softplus(s):
    mean_s = s.sample()
    mean_s = tf.clip_by_value(mean_s, clip_value_min=0,
clip_value_max=None)
    return mean_s

def to_tensor_E(s):
    sample = s.sample()
    return tf.reshape(sample, [sample.shape[1], sample.shape[0]])

def to_tensor_V(s):
    sample = s.sample()
    return tf.reshape(sample, [sample.shape[1], sample.shape[0]])

def lr_scheduler(epoch, lr):
    if epoch < 25:
        return lr
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
else:
    return max(lr * tf.math.exp(-0.05), 0.0005)

lr_callback = tf.keras.callbacks.LearningRateScheduler(lr_scheduler)

h1 = Dense(units=(128), activation='tanh', use_bias=True, name="h1")(xi)
h3 = Dense(units=(128), activation='tanh', use_bias=True,
name="h3")(udist)

h_udist = Dense(units=(m_species), activation='tanh', use_bias=True,
name="h_udist")(h3)
h_udist2 = Dense(units=(latent_dim), activation='tanh', use_bias=True,
name="h_udist2")(h3)

h4 = Dense(units=256, activation='relu', use_bias=True, name="h4")(h1)
dr1 = Dropout(0.1)(h4)

h5 = Dense(units=256, activation='relu', use_bias=True, name="h5")(h1)
dr2 = Dropout(0.1)(h5)

mu_branch = Dense(units=128, activation='tanh', use_bias=True,
name="mu_branch")(dr1)

variational_branch = Dense(units=128, activation='tanh', use_bias=True,
name="variational_branch")(dr2)
variational_branch = Dropout(0.1)(variational_branch)

variational_branch_2 = Dense(units=128, activation='tanh',
use_bias=True, name="variational_branch_2")(dr2)
variational_branch_2 = Dropout(0.1)(variational_branch_2)

mu_params = Dense(m_species, name="mu_params",
activation='linear')(mu_branch)
mu_dist = tfp.layers.DistributionLambda(mu_poisson_2, name="mu_dist",
convert_to_tensor_fn=lambda s: s.mean()(mu_params)

phi_params = Dense(latent_dim * 2, name="phi_params",
activation='linear')(variational_branch)
phi_dist = tfp.layers.DistributionLambda(phi_normal, arguments={'n': 2},
name="phi_dist",
convert_to_tensor_fn=to_tensor_param_sigmoid)(phi_params)
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
psi_params = Dense(m_species * 2, name="psi_params",
activation='linear')(variational_branch_2)
psi_dist = tfp.layers.DistributionLambda(psi_normal, arguments={'n': 2},
name="psi_dist",
convert_to_tensor_fn=to_tensor_param_sigmoid)(psi_params)

pre_Rmat = layers.Multiply()([h_udist2, phi_dist])
Rmat = CovarianceMatrixPredictionLayer(name="Rmat",
output_dim=latent_dim)(pre_Rmat)

pre_Cmat = layers.Multiply()([h_udist, psi_dist])
Cmat = CovarianceMatrixPredictionLayer(name="Cmat",
output_dim=m_species)(pre_Cmat)

e_dist = tfp.layers.DistributionLambda(E_normal, name="e_dist",
convert_to_tensor_fn=to_tensor_E)(Rmat)
Elin = Dense(units=m_species, activation=None, use_bias=False,
name="Elin")(e_dist)

v_dist = tfp.layers.DistributionLambda(V_normal, name="v_dist",
convert_to_tensor_fn=to_tensor_V)(Cmat)

sigma_h = Dense(units=m_species, kernel_constraint=NonNeg(),
activation=None, use_bias=False, name="sigma_h")(di)
sigma_v_hat = Multiply(name="sigma_v_hat")([sigma_h, v_dist])

lin = layers.Add()([Elin, sigma_v_hat])

exp_lin = layers.Lambda(lambda x: K.exp(x), name="Lamb")(lin)
xihat = Multiply(name="xihat")([mu_dist, exp_lin])

model = Model(inputs=[xi, ui, di], outputs=xihat,
name=f'model_{trial_type}')
model.compile(loss='mse',
optimizer=optimizers.RMSprop(learning_rate=0.0015),
metrics=['accuracy'])
model.summary()

# Fitting the model
start_time = time.time()
```

Lampiran 3 Lanjutan Kode Pelatihan Model Probabilistic Deep Learning pada Studi Simulasi

```
if is_training_new:
    history = model.fit(x=[sim_train, cell_centers_train, dummy_train],
                       y=sim_train, shuffle=True, epochs=epochs,
batch_size=batch_size, callbacks=[lr_callback],
                       validation_data=(sim_test, cell_centers_test,
dummy_test],
                                   sim_test))

end_time = time.time()

training_time = round((end_time - start_time), 2)
print(f'Training Time: {training_time} seconds')

if is_training_new:
    plt.figure(figsize=(10,8))

    plt.subplot(2, 1, 1)
    plt.plot(history.history['loss'], label="Train data", linewidth=1)
    plt.plot(history.history['val_loss'], label="Validation data",
linewidth=1)
    plt.xlabel("Epoch")
    plt.ylabel("Loss (MSE)")
    plt.ylim(0, 0.5)
    plt.title(f"Model Loss per Epoch")
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.plot(history.history['accuracy'], label="Train data",
linewidth=1)
    plt.plot(history.history['val_accuracy'], label="Validation data",
linewidth=1)
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.title(f"Model Accuracy per Epoch")
    plt.legend()

    plt.tight_layout()

if is_training_new:
    model.save(f'model/{trial_type}/model_{sim_number}.h5')
else:
    model.load_weights(f'model/model_{trial_type}.h5')
```

Lampiran 4 Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Simulasi

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import os
import random
from sklearn.metrics import mean_squared_error, mean_absolute_error
import seaborn as sns
from scipy import stats
from tensorflow.keras.layers import Input, Lambda, Dense, add,
concatenate, BatchNormalization, Dropout, Concatenate, Multiply
from tensorflow.keras.models import Model
from tensorflow.keras.constraints import Constraint, NonNeg
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K
from tensorflow.keras import layers, optimizers
import tensorflow_probability as tfp
#from tensorflow.keras import objectives

import tensorflow as tf
import time
from tensorflow.python.framework.ops import disable_eager_execution,
enable_eager_execution

sim_data =
pd.read_csv(f"discretization/dataset/simulasi_{sim_number}.csv")
coord_data = pd.read_csv("discretization/dataset/coord.csv")

# slicing dataset
sim_data = sim_data.iloc[:, :2 + n_species]
sim_data = sim_data.loc[sim_data['time'] <= n_time]

species_names = sim_data.columns[2:].tolist()

# number of observations: grid cells
n_obs = sim_data.shape[0]

# grid coordinates of the cells
cell_centers = coord_data.iloc[:, [1, 2]]
cell_centers_0 = cell_centers.iloc[sim_data.iloc[:, 1] - 1, :]

dummy_0 = np.ones(sim_data.shape[0])
sim_data_0 = np.delete(sim_data.values, [0, 1], axis=1)
```

Lampiran 4 Lanjutan Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Simulasi

```
eval_metrics = model.evaluate(x=[sim_data_0, cell_centers_0, dummy_0],
y=sim_data_0, batch_size=batch_size)

yout = model.predict([sim_data_0, cell_centers_0, dummy_0],
batch_size=batch_size)
yout_df = pd.DataFrame(yout, columns=species_names)
loc_sim_data = sim_data.iloc[:, 1]

# ### Estimated Parameters
mu_model = Model(model.inputs, mu_dist)
e_model = Model(model.inputs, e_dist)
v_model = Model(model.inputs, v_dist)

est_mu_dist = mu_model.predict([sim_data_0, cell_centers_0, dummy_0],
batch_size=batch_size)
est_e_dist = e_model.predict([sim_data_0, cell_centers_0, dummy_0],
batch_size=batch_size)
est_v_dist = v_model.predict([sim_data_0, cell_centers_0, dummy_0],
batch_size=batch_size)

est_mu = est_mu_dist.copy()
est_e = est_e_dist.copy()
est_v = est_v_dist.copy()

est_mu_df = pd.DataFrame(est_mu, columns=species_names)
est_e_df = pd.DataFrame(est_e)
est_v_df = pd.DataFrame(est_v, columns=species_names)

est_alpha = model.get_layer('Elin').get_weights()[0].reshape(m_species,
latent_dim)
est_sigma = model.get_layer('sigma_h').get_weights()[0]

est_alpha_df = pd.DataFrame(est_alpha)
est_sigma_df = pd.DataFrame(est_sigma.reshape((n_time, m_species)),
columns=species_names)

# saving to csv
save_path = f'result/{trial_type}'
if is_training_new:
    yout_df.to_csv(f'{save_path}/est_lambda_besar/est_lambda_besar_{sim_
number}.csv', index=False)
    est_mu_df.to_csv(f'{save_path}/est_mu/est_mu_{sim_number}.csv',
index=False)
```

Lampiran 4 Lanjutan Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Simulasi

```
est_alpha_df.to_csv(f'{save_path}/est_alpha/est_alpha_{sim_number}.csv',
index=False)
    est_e_df.to_csv(f'{save_path}/est_e/est_e_{sim_number}.csv',
index=False)
    est_sigma_df.to_csv(f'{save_path}/est_sigma/est_sigma_{sim_number}.c
sv', index=False)
    est_v_df.to_csv(f'{save_path}/est_v/est_v_{sim_number}.csv',
index=False)

phi_model = Model(model.inputs, phi_dist)
psi_model = Model(model.inputs, psi_dist)

est_phi = phi_model.predict([sim_data_0, cell_centers_0, dummy_0],
batch_size=batch_size)
est_psi = psi_model.predict([sim_data_0, cell_centers_0, dummy_0],
batch_size=batch_size)

est_phi_df = pd.DataFrame(est_phi).mean(axis=0)
est_psi_df = pd.DataFrame(est_psi, columns=species_names).mean(axis=0)

# saving to csv
if is_training_new:
    est_phi_df.to_csv(f'{save_path}/est_phi/est_phi_{sim_number}.csv',
index=False)
    est_psi_df.to_csv(f'{save_path}/est_psi/est_psi_{sim_number}.csv',
index=False)
```

Lampiran 5 Kode Diskretisasi *Point Pattern* dan Variabel *Covariate* pada Studi Terapan

```
setwd("D:/Master/Thesis/Codes/Mateu Jalilian 22_Prioritas 2")

file_names <- paste("dataset/bci.tree/bci.tree", 1:8, ".rdata", sep="")
lapply(file_names, load, .GlobalEnv)

bci <- list(census1=bci.tree1, census2=bci.tree2, census3=bci.tree3,
           census4=bci.tree4, census5=bci.tree5, census6=bci.tree6,
           census7=bci.tree7, census8=bci.tree8)

prepfun <- function(obj)
{
  ok <- obj$status == "A"
  data.frame(species=obj$sp[ok], x=obj$gx[ok], y=obj$gy[ok])
}
```


Lampiran 5 Lanjutan Kode Diskretisasi *Point Pattern* dan Variabel *Covariate* pada Studi Terapan

```
dat <- rbind.data.frame(data.frame(time=1, prepfun(bci.tree1)),
                        data.frame(time=2, prepfun(bci.tree2)),
                        data.frame(time=3, prepfun(bci.tree3)),
                        data.frame(time=4, prepfun(bci.tree4)),
                        data.frame(time=5, prepfun(bci.tree5)),
                        data.frame(time=6, prepfun(bci.tree6)),
                        data.frame(time=7, prepfun(bci.tree7)),
                        data.frame(time=8, prepfun(bci.tree8)))

dat <- na.omit(dat)
table(dat$time)
table(dat$species)

# selecting 100 most abundant species
sum(table(dat$species) > 8 * 302)
spChosen <- names(table(dat$species)[table(dat$species) > 8 * 302])
ok <- dat$species %in% spChosen
dat <- dat[ok, ]

library(feather)
write_feather(dat, path="dataset/bci100sp8census.feather")

library(spatstat)
Win <- owin(c(0, 1000), c(0, 500))
par(mfrow=c(8, 4), mar=c(0, 0, 1, 0))
for (i in 1:8)
{
  for (sp in c("chr1ec", "cappfr", "des2pa", "hybapr"))
  {
    plot(0, xlim=c(0, 1000), ylim=c(0, 500), axes=F,
         main=paste0("species: ", sp, ", census: ", i), frame.plot=F,
type="n")
    plot(Win, add=TRUE)
    ok <- (dat$time == i) & (dat$species == sp)
    points(dat$x[ok], dat$y[ok], cex=0.075, col="grey20")
  }
}
dev.copy2pdf(file="dataset/bcidata.pdf", width=8, height=10)

win.graph()
par(mfrow=c(2, 2), mar=c(0, 0, 1, 0))
for (i in 1:1)
{
```

Lampiran 5 Lanjutan Kode Diskretisasi *Point Pattern* dan Variabel *Covariate* pada Studi Terapan

```
for (sp in c("chr1ec", "cappfr", "socrex", "guatdu"))
{
  plot(0, xlim=c(0, 1000), ylim=c(0, 500), axes=F,
       main=paste0("species: ", sp), frame.plot=F, type="n")
  plot(Win, add=TRUE)
  ok <- (dat$time == i) & (dat$species == sp)
  points(dat$x[ok], dat$y[ok], cex=1.0, pch = 19, col="grey20")
}
}

# discretization
nx <- 200
ny <- 100

dfun <- function(t)
{
  df <- data.frame(time=rep(t, nx * ny), loc=1:(nx * ny))
  for (s in spChosen)
  {
    ok <- (dat$time == t) & (dat$species == s)
    X <- ppp(x=dat$x[ok], y=dat$y[ok], window=Win)
    df <- data.frame(df, c(quadratcount(X, nx=nx, ny=ny)))
  }
  names(df) <- c("time", "loc", spChosen)
  return(df)
}

dfout <- rbind.data.frame(dfun(1), dfun(2), dfun(3), dfun(4),
                          dfun(5), dfun(6), dfun(7), dfun(8))

object.size(dfout) / 1024^2

write.csv(dfout, file="dataset/bci_full_temporal.csv",
          row.names=FALSE, quote=FALSE)

# BCI covariates
load("dataset/bci.covars.Rda")
bci.covars
dim(bci.covars)
midmean <- function(x)
{
  k <- length(x)
  out <- numeric(k - 1)
```

Lampiran 5 Lanjutan Kode Diskretisasi *Point Pattern* dan Variabel *Covariate* pada Studi Terapan

```
for (i in 1:(k - 1))
  out[i] <- mean(x[c(i, i + 1)])
  return(out)
}

Q <- quadratcount.ppp(rpoispp(0.001, win=Win), nx=nx, ny=ny)
xx <- midmean(attr(Q, "xbreaks"))
yy <- midmean(rev(attr(Q, "ybreaks")))
xx <- rep(xx, each=ny)
yy <- rep(yy, nx)
cdat <- data.frame(loc=1:(nx * ny), x=xx, y=yy)

for (i in 1:length(bci.covars))
{
  cdat <- data.frame(cdat, interp.im(bci.covars[[i]], x=xx, y=yy))
}
names(cdat) <- c("loc", "x", "y", names(bci.covars))

object.size(cdat) / 1024^2
write.csv(cdat, file="dataset/bci_covars.csv",
          row.names=FALSE, quote=FALSE)
par(mfrow=c(3, 5), mar=c(0, 0, 1, 0))
for (i in 1:13)
{
  plot(0, xlim=c(0, 1000), ylim=c(0, 500), axes=F,
       main=names(bci.covars)[i],
       frame.plot=F, type="n")
  plot(bci.covars[[i]], main="", add=TRUE)
  plot(Win, add=TRUE, main="")
}

plot(0, xlim=c(0, 1000), ylim=c(0, 500), axes=F,
     main="x", frame.plot=F, type="n")
plot(as.im(function(x,y){ x }, W=Win), main="", add=TRUE)
plot(Win, add=TRUE, main="")

plot(0, xlim=c(0, 1000), ylim=c(0, 500), axes=F,
     main="y", frame.plot=F, type="n")
plot(as.im(function(x,y){ y }, W=Win), main="", add=TRUE)
plot(Win, add=TRUE, main="")

dev.copy2pdf(file="dataset/bcicovars.pdf", width=10, height=4)
```

Lampiran 6 Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import os
import random

# path to the data directory
path = '/content/drive/My Drive/Thesis/my_work'
os.chdir(path)
# #!ls
# #os.listdir(path)

# True if we want to training new model. False if we want to use saved
model
is_training_new = True
trial_type = 'trial_4h_v6.1'

save_path = f'result/{trial_type}'
bci_data = pd.read_csv("dataset/bci_full_temporal.csv")
bci_covars = pd.read_csv("dataset/bci_covars.csv")

try:
    metric_rec_df = pd.read_csv(f'{save_path}/metric_{trial_type}.csv')
except FileNotFoundError:
    metric_rec_df = pd.DataFrame({
        'training_time': [],
        'mse_train': [],
        'mse_test': [],
        'mse_all': [],
        'acc_train': [],
        'acc_test': [],
        'acc_all': [],
    })

train_number = metric_rec_df.shape[0] + 1

bci_data = pd.read_csv("dataset/bci_full_temporal.csv")
bci_covars = pd.read_csv("dataset/bci_covars.csv")
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
# slicing dataset
n_species = 25
n_time = 4

bci_data = bci_data.iloc[:, :2 + n_species]
bci_data = bci_data.loc[bci_data['time'] <= n_time]

species_names = bci_data.columns[2:].tolist()
covars_names = bci_covars.columns[1:].tolist()

# number of observations: grid cells
n_obs = bci_data.shape[0]
# number of species
m_species = len(species_names)
# dimension of discretization grid
grid_dimyx = (100, 200)

# grid coordinates of the cells
cell_centers = bci_covars.iloc[:, [1, 2]]
plt.scatter(cell_centers.iloc[:, 0], cell_centers.iloc[:, 1], s=0.05)
plt.show()
plt.clf()

covars = bci_covars.iloc[:, 1:16]
# standardizing the spatial covariates
covars = (covars - covars.mean(axis=0)) / covars.std(axis=0)

""""# Splitting Data into Training and Testing Sets""""

# split to training and test samples
test_idx = np.random.choice(n_obs, size=int(0.3 * n_obs), replace=False)

plt.scatter(cell_centers.iloc[bci_data.iloc[test_idx, 1] - 1, 0],
cell_centers.iloc[bci_data.iloc[test_idx, 1] - 1, 1], s=0.01)
plt.show()
plt.clf()

# split spatio-temporal counts
bci_train = np.delete(bci_data.values, test_idx, axis=0)
bci_test = bci_data.iloc[test_idx, :].values
# split grid cells
cell_centers_train = cell_centers.iloc[bci_train[:, 1] - 1, :]
cell_centers_test = cell_centers.iloc[bci_test[:, 1] - 1, :]
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
# split spatial covariates
covars_train = covars.iloc[bci_train[:, 1] - 1, :]
covars_test = covars.iloc[bci_test[:, 1] - 1, :]

time_train = np.reshape(bci_train[:, 0], (bci_train.shape[0],))
time_test = np.reshape(bci_test[:, 0], (bci_test.shape[0],))

time_train.shape, time_test.shape

dummy_time_train = pd.get_dummies(time_train)
dummy_time_test = pd.get_dummies(time_test)

bci_train = np.delete(bci_train, [0, 1], axis=1)
bci_test = np.delete(bci_test, [0, 1], axis=1)

"""# Constructing the Model"""

from tensorflow.keras.layers import Input, Lambda, Dense, add,
concatenate, BatchNormalization, Dropout, Concatenate, Multiply
from tensorflow.keras.models import Model
from tensorflow.keras.constraints import Constraint, NonNeg
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K
from tensorflow.keras import layers, optimizers
import tensorflow_probability as tfp
#from tensorflow.keras import objectives

import tensorflow as tf

import time
from tensorflow.python.framework.ops import disable_eager_execution,
enable_eager_execution

# encoder network parameters
batch_size = 200
latent_dim = 5
epochs = 50

### Input layers

# spatio-temporal counts and regional covariates
xi = Input(batch_shape=(batch_size, m_species), name="counts")
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
# spatial covariates
zi = Input(batch_shape=(batch_size, covars_train.shape[1]),
name="covariates")
# spatial coordinates of cells
ui = Input(batch_shape=(batch_size, 2), name="coordinates")
# time instances
ti = Input(batch_shape=(batch_size, ), name="times")
# dummy time instance
dti = Input(batch_shape=(batch_size, n_time), name="dummy_times")

import tensorflow as tf
from tensorflow.keras.layers import Layer

class CovarianceMatrixPredictionLayer(Layer):
    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        self.jitter = 1e-5
        super(CovarianceMatrixPredictionLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        input_dim = input_shape[-1]
        batch_size = input_shape[0]

        self.lower_triangular_weights =
            self.add_weight(name='lower_triangular_weights',
                shape=(self.output_dim, batch_size, self.output_dim),
                initializer='glorot_uniform',
                trainable=True)
        super(CovarianceMatrixPredictionLayer, self).build(input_shape)

    def call(self, x):
        batch_size = tf.shape(x)[0]
        input_dim = tf.shape(x)[-1]

        lower_triangular = tf.matmul(self.lower_triangular_weights,
            tf.reshape(x, [-1, batch_size]))

        lower_triangular = tf.linalg.set_diag(lower_triangular,
            tf.abs(tf.linalg.diag_part(lower_triangular)))

        mask = tf.linalg.band_part(tf.ones([batch_size, batch_size]), -
1, 0)
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
        lower_triangular *= mask

    return lower_triangular

def compute_output_shape(self, input_shape):
    return (input_shape[0], self.output_dim, self.output_dim)

"""Network architecture: from observations to latent field parameters"""

# convert cell coordinates to pairwise spatial distances
def udistfun(coords):
    # computes the pairwise distances between points
    #dist2 = K.sum((coords[None, :] - coords[:, None]) * (coords[None,
:] - coords[:, None]), axis=-1)
    dist2 = K.sum(tf.square(coords[None, :] - coords[:, None]), axis=-1)
    return tf.sqrt(dist2)

# convert time instances to pairwise distances between times
def tdistfun(times):
    # computes the pairwise distances between times
    return tf.abs(times[:,None] - times)

udist = Lambda(udistfun, name="spatialdist")(ui)
tdist = Lambda(tdistfun, name="temporaldist")(ti)

def normalize_dist(args):
    val, mean, std = args
    return (val - mean) / std

def concat_layers(args):
    return K.concatenate(args, axis=1)

def concat_sigma(args):
    return K.mean(args, axis=1)

tfd = tfp.distributions

def mu_poisson_2(params):
    rate = tf.math.exp(params)
    return tfd.Poisson(rate=rate)

def sphi_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
```


Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
loc = tf.nn.sigmoid(un_loc)
scale = 1e-5 + tf.math.softplus(0.05 * un_scale)

return tfd.Normal(loc=loc, scale=scale)

def salf_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.math.sigmoid(un_loc)
    scale = 1e-5 + tf.math.softplus(0.05 * un_scale)

    return tfd.Normal(loc=loc, scale=scale)

def tphi_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.math.tanh(un_loc)
    scale = 1e-5 + tf.math.softplus(0.05 * un_scale)

    return tfd.Normal(loc=loc, scale=scale)

def spsi_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.nn.sigmoid(un_loc)
    scale = 1e-5 + tf.math.softplus(0.05 * un_scale)

    return tfd.Normal(loc=loc, scale=scale)

def tpsi_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.math.tanh(un_loc)
    scale = 1e-5 + tf.math.softplus(0.05 * un_scale)

    return tfd.Normal(loc=loc, scale=scale)

def sigma_normal(params, n):
    un_loc, un_scale = tf.split(params, num_or_size_splits=n, axis=1)
    loc = tf.math.softplus(un_loc)
    scale = 1e-5 + tf.math.softplus(0.05 * un_scale)

    return tfd.Normal(loc=loc, scale=scale)

def E_normal(params):
    return tfd.MultivariateNormalTriL(loc = 0, scale_tril = params)
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
def V_normal(params):
    return tfd.MultivariateNormalTril(loc = 0, scale_tril = params)

def to_tensor_param_tanh(s):
    mean_s = s.sample()
    mean_s = tf.clip_by_value(mean_s, clip_value_min=-1,
clip_value_max=1)
    return mean_s

def to_tensor_param_sigmoid(s):
    mean_s = s.sample()
    mean_s = tf.clip_by_value(mean_s, clip_value_min=0,
clip_value_max=1)
    return mean_s

def to_tensor_param_softplus(s):
    mean_s = s.sample()
    mean_s = tf.clip_by_value(mean_s, clip_value_min=0,
clip_value_max=np.inf)
    return mean_s

def to_tensor_E(s):
    sample = s.sample()
    return tf.reshape(sample, [sample.shape[1], sample.shape[0]])

def to_tensor_V(s):
    sample = s.sample()
    return tf.reshape(sample, [sample.shape[1], sample.shape[0]])

def lr_scheduler(epoch, lr):
    if epoch < 25:
        return lr
    else:
        return max(lr * tf.math.exp(-0.05), 0.0005)

lr_callback = tf.keras.callbacks.LearningRateScheduler(lr_scheduler)

n_node_1 = min((m_species * 5), 384)

h1 = Dense(units=(n_node_1), activation='tanh', use_bias=True,
name="h1")(xi)
h2 = Dense(units=(128), activation='tanh', use_bias=True, name="h2")(zi)
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
h3 = Dense(units=(n_node_1), activation='tanh', use_bias=True,
name="h3")(u_dist)
h_u_dist = Dense(units=(m_species), activation='tanh', use_bias=True,
name="h_u_dist")(h3)
h_u_dist2 = Dense(units=(latent_dim), activation='tanh', use_bias=True,
name="h_u_dist2")(h3)

ht = Dense(units=(128), activation='tanh', use_bias=True,
name="ht")(t_dist)
h_t_dist = Dense(units=(m_species), activation='tanh', use_bias=True,
name="h_t_dist")(ht)
h_t_dist2 = Dense(units=(latent_dim), activation='tanh', use_bias=True,
name="h_t_dist2")(ht)

h_concat_1 = concatenate([h1, h2], axis=1, name="h_concat_1")

h4 = Dense(units=n_node_1 * 2, activation='relu', use_bias=True,
name="h4")(h_concat_1)
dr1 = Dropout(0.1)(h4)

h5 = Dense(units=n_node_1 * 2, activation='relu', use_bias=True,
name="h5")(h_concat_1)
dr2 = Dropout(0.1)(h5)

mu_branch = Dense(units=n_node_1, activation='tanh', use_bias=True,
name="mu_branch")(dr1)

variational_branch = Dense(units=n_node_1, activation='tanh',
use_bias=True, name="variational_branch")(dr2)
variational_branch = Dropout(0.1)(variational_branch)

variational_branch_2 = Dense(units=n_node_1, activation='tanh',
use_bias=True, name="variational_branch_2")(dr2)
variational_branch_2 = Dropout(0.1)(variational_branch_2)

mu_params = Dense(m_species, name="mu_params",
activation='linear')(mu_branch)
mu_dist = tfp.layers.DistributionLambda(mu_poisson_2, name="mu_dist",
convert_to_tensor_fn=lambda s: s.mean())(mu_params)

sphi_params = Dense(latent_dim * 2, name="sphi_params",
activation='linear')(variational_branch)
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
sphi_dist = tfp.layers.DistributionLambda(sphi_normal, arguments={'n':
2}, name="sphi_dist",
convert_to_tensor_fn=to_tensor_param_sigmoid)(sphi_params)

salf_params = Dense(latent_dim * 2, name="salf_params",
activation='linear')(variational_branch)
salf_dist = tfp.layers.DistributionLambda(salf_normal, arguments={'n':
2}, name="salf_dist",
convert_to_tensor_fn=to_tensor_param_sigmoid)(salf_params)

tphi_params = Dense(latent_dim * 2, name="tphi_params",
activation='linear')(variational_branch)
tphi_dist = tfp.layers.DistributionLambda(tphi_normal, arguments={'n':
2}, name="tphi_dist",
convert_to_tensor_fn=to_tensor_param_tanh)(tphi_params)

spsi_params = Dense(m_species * 2, name="spsi_params",
activation='linear')(variational_branch_2)
spsi_dist = tfp.layers.DistributionLambda(spsi_normal, arguments={'n':
2}, name="spsi_dist",
convert_to_tensor_fn=to_tensor_param_sigmoid)(spsi_params)

tpsi_params = Dense(m_species * 2, name="tpsi_params",
activation='linear')(variational_branch_2)
tpsi_dist = tfp.layers.DistributionLambda(tpsi_normal, arguments={'n':
2}, name="tpsi_dist",
convert_to_tensor_fn=to_tensor_param_tanh)(tpsi_params)

pre_Rmat = layers.Multiply()([h_udist2, h_tdist2, sphi_dist, salf_dist,
tphi_dist])
Rmat = CovarianceMatrixPredictionLayer(name="Rmat",
output_dim=latent_dim)(pre_Rmat)

pre_Cmat = layers.Multiply()([h_udist, h_tdist, spsi_dist, tpsi_dist])
Cmat = CovarianceMatrixPredictionLayer(name="Cmat",
output_dim=m_species)(pre_Cmat)

e_dist = tfp.layers.DistributionLambda(E_normal, name="e_dist",
convert_to_tensor_fn=to_tensor_E)(Rmat)
Elin = Dense(units=m_species, activation=None, use_bias=False,
name="Elin")(e_dist)
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
v_dist = tfp.layers.DistributionLambda(V_normal, name="v_dist",
convert_to_tensor_fn=to_tensor_V)(Cmat)
sigma_h = Dense(units=m_species, kernel_constraint=NonNeg(),
activation=None, use_bias=False, name="sigma_h")(dti)
sigma_v_hat = Multiply(name="sigma_v_hat")([sigma_h, v_dist])

lin = layers.Add()([Elin, sigma_v_hat])

exp_lin = layers.Lambda(lambda x: K.exp(x), name="Lamb")(lin)
xihat = Multiply(name="xihat")([mu_dist, exp_lin])

model = Model(inputs=[xi, zi, ui, ti, dti], outputs=xihat,
name=f'model_{trial_type}')
model.compile(loss='mse',
optimizer=optimizers.RMSprop(learning_rate=0.0015),
metrics=['accuracy'])
model.summary()

# plot_model(model, show_shapes=True, show_layer_names=True,
to_file="metrics/model_3.1a.jpg")

# Fitting the model
start_time = time.time()

if is_training_new:
    history = model.fit(x=[bci_train, covars_train, cell_centers_train,
time_train, dummy_time_train],
y=bci_train, shuffle=True, epochs=epochs,
batch_size=batch_size, callbacks=[lr_callback],
validation_data=([bci_test, covars_test,
cell_centers_test, time_test, dummy_time_test],
bci_test))

end_time = time.time()

training_time = round((end_time - start_time), 2)
print(f'Training Time: {training_time} seconds')

if is_training_new:
    plt.figure(figsize=(10,8))

    plt.subplot(2, 1, 1)
    plt.plot(history.history['loss'], label="Train data", linewidth=1)
```

Lampiran 6 Lanjutan Kode Pelatihan Model *Probabilistic Deep Learning* pada Studi Terapan

```
plt.plot(history.history['val_loss'], label="Validation data",
linewidth=1)
    plt.xlabel("Epoch")
    plt.ylabel("Loss (MSE)")
    plt.ylim(0, 0.5)
    plt.title(f"Model Loss per Epoch")
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.plot(history.history['accuracy'], label="Train data",
linewidth=1)
    plt.plot(history.history['val_accuracy'], label="Validation data",
linewidth=1)
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.title(f"Model Accuracy per Epoch")
    plt.legend()

    plt.tight_layout()

if is_training_new:
    model.save(f'model/{trial_type}/model_{train_number}.h5')
else:
    model.load_weights(f'model/model_{trial_type}.h5')
```

Lampiran 7 Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Terapan

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import seaborn as sns
from scipy import stats
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import os
import random
from tensorflow.keras.layers import Input, Lambda, Dense, add,
concatenate, BatchNormalization, Dropout, Concatenate, Multiply
from tensorflow.keras.models import Model
```

Lampiran 7 Lanjutan Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Terapan

```
from tensorflow.keras.constraints import Constraint, NonNeg
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K
from tensorflow.keras import layers, optimizers
import tensorflow_probability as tfp
import tensorflow as tf
import time
from tensorflow.python.framework.ops import disable_eager_execution,
enable_eager_execution

bci_data = pd.read_csv("dataset/bci_full_temporal.csv")
bci_covars = pd.read_csv("dataset/bci_covars.csv")

# slicing dataset
bci_data = bci_data.iloc[:, :2 + n_species]
bci_data = bci_data.loc[bci_data['time'] <= n_time]

species_names = bci_data.columns[2:].tolist()
covars_names = bci_covars.columns[1:].tolist()

# number of observations: grid cells
n_obs = bci_data.shape[0]

# grid coordinates of the cells
cell_centers = bci_covars.iloc[:, [1, 2]]

covars = bci_covars.iloc[:, 1:16]
# standardizing the spatial covariates
covars = (covars - covars.mean(axis=0)) / covars.std(axis=0)

cell_centers_0 = cell_centers.iloc[bci_data.iloc[:, 1] - 1, :]
covars_0 = covars.iloc[bci_data.iloc[:, 1] - 1, :].values

time_0 = np.reshape(bci_data.iloc[:, 0], (bci_data.shape[0],))
dummy_time_0 = pd.get_dummies(time_0)
bci_data_0 = np.delete(bci_data.values, [0, 1], axis=1)

eval_metrics = model.evaluate(x=[bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], \
                               y=bci_data_0, batch_size=batch_size)

yout = model.predict([bci_data_0, covars_0, cell_centers_0, time_0,
dummy_time_0], batch_size=batch_size)
```

Lampiran 7 Lanjutan Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Terapan

```
yout_df = pd.DataFrame(yout, columns=species_names)

"""### Estimated Parameters"""

mu_model = Model(model.inputs, mu_dist)
e_model = Model(model.inputs, e_dist)
v_model = Model(model.inputs, v_dist)

est_mu_dist = mu_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)
est_e_dist = e_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)
est_v_dist = v_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)

est_mu = est_mu_dist.copy()
est_e = est_e_dist.copy()
est_v = est_v_dist.copy()

est_mu_df = pd.DataFrame(est_mu, columns=species_names)
est_e_df = pd.DataFrame(est_e)
est_v_df = pd.DataFrame(est_v, columns=species_names)

est_alpha = model.get_layer('Elin').get_weights()[0].reshape(m_species,
latent_dim)
est_sigma = model.get_layer('sigma_h').get_weights()[0]

est_alpha_df = pd.DataFrame(est_alpha)
est_sigma_df = pd.DataFrame(est_sigma.reshape((n_time, m_species)),
columns=species_names)

# saving to csv
if is_training_new:
    yout_df.to_csv(f'{save_path}/est_lambda_besar/est_lambda_besar_{train_
number}.csv', index=False)
    est_mu_df.to_csv(f'{save_path}/est_mu/est_mu_{train_number}.csv',
index=False)
    est_alpha_df.to_csv(f'{save_path}/est_alpha/est_alpha_{train_number}
.csv', index=False)
    est_e_df.to_csv(f'{save_path}/est_e/est_e_{train_number}.csv',
index=False)
    est_sigma_df.to_csv(f'{save_path}/est_sigma/est_sigma_{train_number}
.csv', index=False)
```


Lampiran 7 Lanjutan Kode Prediksi pada Keseluruhan Data menggunakan Model *Probabilistic Deep Learning* pada Studi Terapan

```
est_v_df.to_csv(f'{save_path}/est_v/est_v_{train_number}.csv',
index=False)
flatten_dim = grid_dimyx[0] * grid_dimyx[1]

"""### Getting Parameter Distribution"""

sphi_model = Model(model.inputs, sphi_dist)
salf_model = Model(model.inputs, salf_dist)
tphi_model = Model(model.inputs, tphi_dist)
spsi_model = Model(model.inputs, spsi_dist)
tpsi_model = Model(model.inputs, tpsi_dist)

est_sphi = sphi_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)
est_salf = salf_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)
est_tphi = tphi_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)
est_spsi = spsi_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)
est_tpsi = tpsi_model.predict([bci_data_0, covars_0, cell_centers_0,
time_0, dummy_time_0], batch_size=batch_size)

est_sphi_df = pd.DataFrame(est_sphi).mean(axis=0)
est_tphi_df = pd.DataFrame(est_tphi).mean(axis=0)
est_salf_df = pd.DataFrame(est_salf).mean(axis=0)
est_spsi_df = pd.DataFrame(est_spsi, columns=species_names).mean(axis=0)
est_tpsi_df = pd.DataFrame(est_tpsi, columns=species_names).mean(axis=0)

# saving to csv
if is_training_new:
    est_sphi_df.to_csv(f'{save_path}/est_sphi/est_sphi_{train_number}.cs
v', index=False)
    est_tphi_df.to_csv(f'{save_path}/est_tphi/est_tphi_{train_number}.cs
v', index=False)
    est_salf_df.to_csv(f'{save_path}/est_salf/est_salf_{train_number}.cs
v', index=False)
    est_spsi_df.to_csv(f'{save_path}/est_spsi/est_spsi_{train_number}.cs
v', index=False)
est_tpsi_df.to_csv(f'{save_path}/est_tpsi/est_tpsi_{train_number}.csv',
index=False)
```

Lampiran 8 Perbandingan Hasil Estimasi Parameter pada Studi Simulasi

Parameter	True Parameter	Choiruddin dkk (2020)	Mateu & Jalilian (2022)	Probabilistic Deep Learning
$\hat{\mu}$	5000	4769.554	4937.458	5087.121
$\hat{\alpha}$	-0.0043	-0.0015	-0.0001	-0.0001
$\hat{\phi}$	0.3250	0.0681	0.0001	0.0852
$\hat{\sigma}$	0.9793	0.9964	0.0001	0.0057
$\hat{\psi}$	0.40	0.1118	0.0001	0.2579
$\hat{\Lambda}$	-	-	4937.848	5087.090

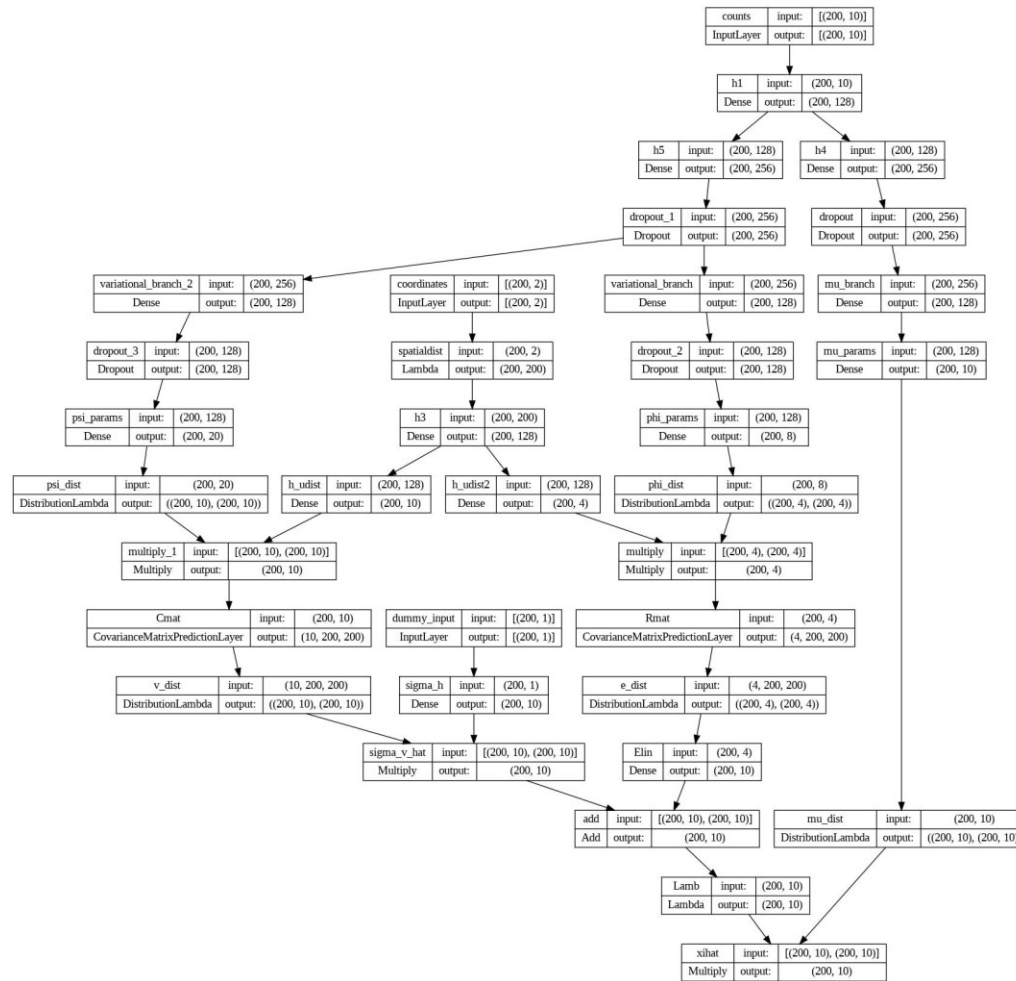
Lampiran 9 Perbandingan Hasil Estimasi Parameter pada Studi Terapan untuk 25 Spesies dan 4 Waktu Observasi

Parameter	Mateu & Jalilian (2022)	Probabilistic Deep Learning
$\hat{\mu}$	6831.788	7367.663
$\hat{\alpha}$	0.0001	0.0033
$\hat{\phi}$	0.6942	0.3367
$\hat{\varphi}$	0.5000	0.3345
$\hat{\rho}$	0.0003	0.0014
$\hat{\sigma}$	0.2230	0.4807
$\hat{\psi}$	-0.0002	0.0282
$\hat{\zeta}$	0.1538	0.0556
$\hat{\Lambda}$	6832.210	7367.662

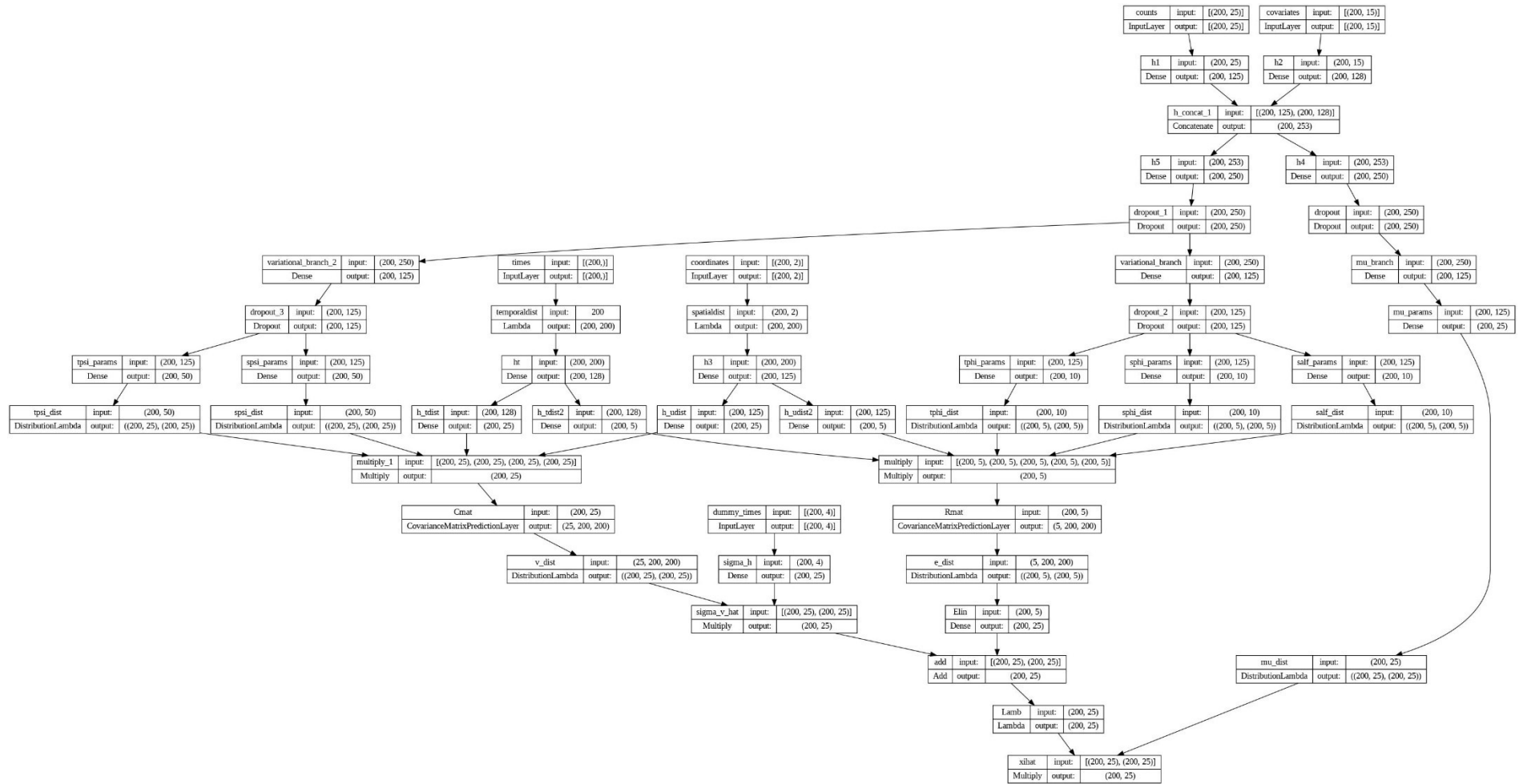
Lampiran 10 Perbandingan Hasil Estimasi Parameter pada Studi Terapan untuk 100 Spesies dan 8 Waktu Observasi

Parameter	Mateu & Jalilian (2022)	Probabilistic Deep Learning
$\hat{\mu}$	17277.268	19913.643
$\hat{\alpha}$	0.0001	-0.0001
$\hat{\phi}$	0.6968	0.2907
$\hat{\varphi}$	0.5004	0.3346
$\hat{\rho}$	-0.0015	-0.0115
$\hat{\sigma}$	0.2615	0.4822
$\hat{\psi}$	0.0002	-0.0132
$\hat{\zeta}$	0.0851	0.0329
$\hat{\Lambda}$	17277.696	19913.646

Lampiran 11 Arsitektur *Probabilistic Deep Learning* untuk Studi Simulasi



Lampiran 12 Arsitektur *Probabilistic Deep Learning* untuk Studi Terapan



BIOGRAFI PENULIS



Penulis bernama lengkap Ekky Rino Fajar Sakti, lahir di Malang, 24 Oktober 1997. Penulis merupakan anak pertama dari dua bersaudara. Penulis menempuh pendidikan formal di: SDN Buring Malang (2004 – 2009), SMPN 2 Malang (2009 – 2012), SMKN 4 Malang jurusan Rekayasa Perangkat Lunak (2012 – 2015), Pendidikan Sarjana dengan beasiswa tingkat satu pada jurusan Teknik Informatika di Universitas Ma Chung, Malang (2015 – 2019).

Setelah lulus sarjana, penulis menambah pengalaman penelitian dengan bekerja sebagai *research assistant* dalam penelitian tentang pengembangan teknologi pertanian menggunakan *computer vision* dari tahun 2019 hingga 2020. Pada tahun 2020 hingga sekarang, penulis aktif bekerja sebagai Data Scientist. Sembari bekerja, penulis melanjutkan pendidikan magister Statistika di Institut Teknologi Sepuluh Nopember, Surabaya dari Agustus 2022 hingga Agustus 2024. Pembaca yang ingin menyampaikan kritik, saran, atau diskusi tentang penelitian ini dapat menghubungi penulis melalui email ekky.rino@gmail.com.