TUGAS AKHIR – RF 184838

# OTOMATISASI ANALISIS FASE SEISMOGRAM GEMPA MIKRO DI LAPANGAN "X" MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK*

CHRISTOPHER SALIM
NRP 03411640000025

DOSEN PEMBIMBING 1 :
Dr. Widya Utama, DEA
NIP. 19760123 200003 1001

DOSEN PEMBIMBING 2 :
Mariyanto, S.Si, M.T.
NIP. 1991201711044

DEPARTEMEN TEKNIK GEOFISIKA
FAKULTAS TEKNIK SIPIL, PERENCANAAN, DAN KEBUMIAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

**TUGAS AKHIR – RF 184838**

# OTOMATISASI ANALISIS FASE SEISMOGRAM GEMPA MIKRO DI LAPANGAN "X" MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK*

CHRISTOPHER SALIM
NRP. 03411640000025

DOSEN PEMBIMBING 1 :
Dr. Widya Utama, DEA
NIP. 19760123 200003 1001

DOSEN PEMBIMBING 2 :
Mariyanto, S.Si, M.T.
NIP. 1991201711044

DEPARTEMEN TEKNIK GEOFISIKA
FAKULTAS TEKNIK SIPIL, PERENCAAN, DAN KEBUMIAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

*UNDERGRADUATE THESIS* – **RF 184838**

# AUTOMATION OF MICRO EARTHQUAKE SEISMOGRAM PHASE ANALYSIS IN "X" FIELD USING CONVOLUTIONAL NEURAL NETWORK ALGORITHM

CHRISTOPHER SALIM
NRP. 03411640000025

Supervisor 1 :
Dr. Widya Utama, DEA
NIP. 19760123 200003 1001

Supervisor 2 :
Mariyanto, S.Si, M.T.
NIP. 1991201711044

GEOPHYSICAL ENGINEERING DEPARTMENT
FACULTY OF CIVIL, PLANNING, AND GEO ENGINEERING
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2020

*Halaman ini sengaja dikosongkan*

# LEMBAR PENGESAHAN

Tugas akhir ini diajukan oleh :

| | |
|---|---|
| Nama | : Christopher Salim |
| NRP | : 03411640000025 |
| Departemen | : Teknik Geofisika |
| Judul Tugas Akhir | : Otomatisasi Analisis Fase Seismogram Gempa Mikro Di Lapangan "X" menggunakan Algoritma *Convolutional Neural Network* |

Telah berhasil dipertahankan di hadapan tim penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Departemen Teknik Geofisika, Fakultas Teknik Sipil, Perencanaan dan Kebumian, Institut Teknologi Sepuluh Nopember.

Surabaya, 9 Januari 2020

1. .....
   Dr. Widya Utama, DEA (Pembimbing I)
   NIP. 19611024 198803 1 001

2. .....
   Mariyanto, S.Si, M.T. (Pembimbing II)
   NIP. 199120171044

3. .....
   Anik Hilyah, S.Si, M.T (Penguji I)
   NIP. 19790813 200812 2002

4. .....
   Juan Pandu Gya Nur Rochman, S.Si., MT (Penguji II)
   NIP. 19800601 220150 1003

**Mengetahui,**
Kepala Laboratorium Petrofisika
Departemen Teknik Geofisika
Fakultas Teknik Sipil, Perencanaan dan Kebumian
Institut Teknologi Sepuluh Nopember Surabaya

Dr. Widya Utama, DEA
NIP. 19611024 198803 1 001

*Halaman ini sengaja dikosongkan*

# PERNYATAAN KEASLIAN TUGAS AKHIR

Saya menyatakan bahwa isi sebagian maupun keseluruh tugas akhir saya dengan judul "Otomatisasi Analisis Fase Seismogram Gempa Mikro untuk Evaluasi Sistem Panas Bumi di Lapangan "X" menggunakan Algoritma Convolutional Neural Network"adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 9 Januari 2020

Christopher Salim
NRP. 03411640000025

*Halaman ini sengaja dikosongkan*

# Otomatisasi Analisis Fase Seismogram Gempa Mikro di Lapangan "X" menggunakan Algoritma *Convolutional Neural Network*


Nama : Christopher Salim
NRP : 03411640000025
Departemen : Teknik Geofisika
Pembimbing I : Dr. Widya Utama, DEA.
Pembimbing II : Mariyanto, S.Si., M.T.


## ABSTRAK


Dalam menentukan fase seismogram gempa mikro dan menentukan waktu tiba gelombang P dan S, metode konvensional yang memerlukan *picking* manual seringkali digunakan oleh geofisikawan. Parameter waktu tiba ini kemudian digunakan untuk menghitung hiposenter dan melakukan *monitoring* pada manifestasi sumber panas bumi. Untuk meminimalkan subjektivitas pada proses *picking*, metode yang menggunakan kecerdasan buatan (*artificial intelligence/AI)* akan dilakukan pada penelitian ini. Algoritma yang akan digunakan adalah *convolutional neural network* (CNN), salah satu cabang dari *deep learning*. Algoritma ini seringkali digunakan untuk mengekstraksi fitur pada suatu gambar dan kemudian mempelajarinya untuk memberi label pada data lain yang memiliki fitur tersebut. CNN memerlukan data seismogram dalam jumlah banyak sebagai data *training* pada awalnya sebagai masukan. CNN yang sudah dilatih ini kemudian akan digunakan untuk menganalisis waktu tiba gelombang P dan S pada data seismogram di lapangan "X" secara otomatis. Fitur yang dipelajari dari data untuk membedakan antara gelombang P dan S adalah perbandingan amplitudo maksimum antara satu komponen dengan komponen pengukuran lainnya. Hasil *accuracy* dan *loss* dari proses *training* dan *testing* adalah rata-rata 0.95 dan 0.15. Sementara itu, ketika sudah diuji dengan data asli, hasil evaluasinya adalah nilai *precision* sebesar 0.80, nilai *recall* sebesar 0.26, dan nilai *accuracy* sebesar 0.91. Berdasarkan hasil tersebut, penelitian ini dapat berguna untuk mengotomatisasi proses awal evaluasi sumber panas bumi dan memberikan faktor objektivitas pada proses *picking* fase gelombang P dan S pada data seismogram.



Kata kunci : *deep learning, waktu tiba, seismogram, convolutional neural network, panas bumi*

*Halaman ini sengaja dikosongkan*

## *Automation Of Micro Earthquake Seismogram Phase Analysis In "X" Field Using Convolutional Neural Network Algorithm*


Name : Christopher Salim
Student ID : 03411640000025
Department : Teknik Geofisika
Advisor I : Dr. Widya Utama, DEA.
Advisor II : Mariyanto, S.Si., M.T.



***ABSTRACT***

*In determining the phase of a microseismic and determining the arrival time of P and S waves, conventional methods that require manual picking are often used by geophysicists. The arrival time parameter is then used to calculate the hypocenter and monitor the manifestation of geothermal sources. To minimize subjectivity in the picking process, methods that use artificial intelligence (AI) will be carried out in this study. The algorithm that will be used is convolutional neural network (CNN), one of the branches of deep learning. This algorithm is often used to extract features in an image and then learn them to label other data that have these features. CNN requires a large amount of seismogram data as training data initially as input. This trained CNN will then be used to analyze the P and S wave arrival time in the seismogram data in the "X" field automatically. The feature learned from the data to distinguish between P and S waves is the ratio of the maximum amplitude of one component to another measurement component. The results of accuracy and loss of the training and testing process are on average 0.95 and 0.15. Meanwhile, when tested with real data, the evaluation results are a precision value of 0.80, a recall value of 0.26, and an accuracy value of 0.91. Based on these results, this research can be useful to automate the initial process of evaluating geothermal sources and provide objectivity factors in the P and S wave phase picking process in seismogram data.*



*Keywords* : *deep learning, arrival time, seismogram, convolutional neural network, geothermal*

*Halaman ini sengaja dikosongkan*

# KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena atas berkat-Nya laporan tugas akhir yang berjudul "Otomatisasi Analisis Fase Seismogram Gempa Mikro untuk Evaluasi Sistem Panas Bumi di Lapangan "X" menggunakan Algoritma Convolutional Neural Network" ini dapat diselesaikan. Pelaksanaan dan penyusunan Laporan Tugas Akhir ini dapat terlaksanakan baik dengan bimbingan, bantuan, dan dukungan berbagai pihak. Pada kesempatan ini, penulis mengucapkan terima kasih kepada:

1. Papa, Mama, Ko Bobby, Ko David dan semua keluarga yang memberikan dukungan moril maupun materi selama penulis menjalani tugas akhir ini.
2. Dr. Widya Utama, DEA dan Mariyanto, S.Si., M.T. selaku pembimbing yang telah meluangkan banyak waktu untuk memberikan bimbingan dan arahan kepada penulis.
3. Anik Hilyah S.Si, M.T. dan Juan Pandu Gya Nur Rochman S.Si., M.T.. selaku dosen penguji tugas akhir yang senantiasa memberikan saran hingga terselesaikannya laporan tugas akhir ini.
4. Bapak dan Ibu dosen Departemen Teknik Geofisika ITS yang senantiasa membimbing dan mendidik penulis selama masa perkuliahan.
5. Jajaran Civitas Akademika Departemen Teknik Geofisika ITS yang senantiasa memberikan bimbingan dan izin melakukan berbagai kegiatan hingga terselesaikannya tugas akhir ini.
6. Mentor dalam pengerjaan tugas akhir ini, Mbak Sherly, yang telah meluangkan banyak waktu untuk membantu dalam memberikan saran untuk penyelesaian tugas akhir yang lebih baik.
7. Rekan-rekan tim tugas akhir MEQ (Tebo, Tyas, Gian) yang dengan setia bekerja sama dengan penulis dalam proses pengerjaan tugas akhir, serta menemani dalam hari-hari kurang tidur bersama.
8. Bagoes Idcha Mawardi, yang dengan tulus membantu pembuatan logo program *picking* otomatis ini.
9. Yuliana Puspa Dewi Rahardjo yang selalu menyemangati dan menemani saat penulis mengerjakan tugas akhir, baik di masa sulit maupun senang.
10. Semua teman-teman yang senantiasa yang menemani dan membantu penulis saat mengerjakan tugas akhir di Lab. Komputasi.
11. Semua pihak yang telah membantu yang tidak dapat dituliskan satu per satu.

Penulis menyadari bahwa pengembangan keilmuan tidak berhenti pada satu hasil. Kritik dan saran maupun diskusi diperlukan agar kebermanfaatan laporan tugas akhir ini semakin dirasakan bagi penulis pribadi maupun bagi pembaca.

Surabaya 9 Januari 2020

Christopher Salim
NRP. 03411640000025

*Halaman ini sengaja dikosongkan*

# DAFTAR ISI

*Halaman ini sengaja dikosongkan*

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR LAMPIRAN

# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang

Gempa mikro atau *microearthquake* (MEQ) adalah gempa yang terukur oleh seismogram dengan magnitudo yang kecil dan relatif tidak terlalu dapat dirasakan oleh manusia. Magnitudo pada umumnya untuk MEQ adalah lebih kecil atau sama dengan 3 SR (Isroi, Singarimbun, & Herdiansyah, 2015). Pada sistem panas bumi, terdapat perubahan tekanan, temperatur, dan fase fluida pada reservoir. Karena perubahan tersebut, terbentuk rekahan-rekahan pada batuan di sekitar reservoir yang mengakibatkan MEQ tersebut terjadi (Kamah, 2016). Dengan demikian, data seismogram yang mengukur MEQ dapat digunakan untuk monitoring sistem panas bumi yang ada di suatu lapangan.

Pada tahap awal pengolahan data MEQ, diperlukan analisis yang objektif agar data selanjutnya dapat dihasilkan dengan tepat. Pada mulanya, penapisan atau filter berguna untuk meminimalkan *noise* yang bukan berasal dari aktivitas reservoir. Setelah itu, dapat dilakukan penentuan waktu tiba dengan *picking*. Dengan data waktu tiba gelombang P dan S karena MEQ, dapat dilakukan perhitungan selanjutnya yaitu relokasi hiposenter MEQ. Cara untuk *picking* ini terdapat dalam berbagai bentuk. Cara konvensional yang paling sering dilakukan adalah dengan memperhatikan secara manual pola gelombang P yang kemudian disusul oleh gelombang S, seperti dideskripsikan oleh metodologi untuk relokasi hiposenter berikut (Hamzah, Makhrani, & Hasni), (Fauzi, 2010), (Nainggolan & Santosa, 2013). Akan tetapi, metode ini sangat rentan terhadap subjektivitas karena pemahaman seorang geofisikawan mengenai fase seismogram dapat berbeda terhadap geofisikawan yang lain. Dengan demikian, hasil hiposenter dapat berbeda-beda.

Selain *picking* secara manual, beberapa metode untuk melakukan *picking* secara otomatis telah dikembangkan. Salah satu metode yang terkenal adalah *short term average/long term average* (STA/LTA) yang dikembangkan oleh Allen (1982) Akan tetapi, metode ini dapat tidak efektif jika terdapat sedikit saja lonjakan amplitudo karena aktivitas manusia. Dengan berkembangnya ilmu statistik, matematika, dan kecerdasan buatan, metode *picking* otomatis terus dikembangkan. Baillard, Crawford, Ballu, Hibert, & Mangeney (2014) mencoba membuat algoritma *picking* otomatis berdasarkan kurtosis data. Akan tetapi, algoritma tersebut hanya didesain untuk jaringan seismik lokal dan memerlukan panjang *window* yang cukup spesifik. Metode lain yang menggunakan AI terus dikembangkan seperti pada metode transformasi *wavelet* (Colak, Destici, Özen, Arman, & Çerezci, 2009), *machine learning* dan *deep learning* (Perol, Gharbi,

& Denolle, 2018) (Zhu & Beroza, 2019) (Zhang, et al., 2018) (Rouet-Leduc, et al., 2017) (Ross, Meier, Hauksson, & Heaton, 2018) (Mousavi, Zhu, Sheng, & Beroza, 2019) (Meier, et al., 2018).

Penelitian ini akan menggunakan dan mengembangkan metode *deep learning* yang sudah ada, dan kali ini akan diterapkan ke data panas bumi. Hal ini membuat penelitian ini berbeda dengan penelitian sebelumnya, karena penelitian sebelumnya hanya ditujukan untuk kegempaan secara bencana. Dengan penelitian ini, diharapkan algoritma ini dapat menjawab kebutuhan eksplorasi panas bumi yang semakin meningkat.

## 1.2. Rumusan Masalah

Rumusan masalah dari penelitian pada tugas akhir ini adalah

1. Bagaimana desain program untuk menghasilkan parameter *picking* seismogram secara otomatis?
2. Bagaimana penerapan program pada data seismogram gempa mikro di Lapangan "X"?
3. Bagaimana tingkat akurasi dan efisiensi algoritma CNN untuk *picking* dibandingkan dengan hasil *picking* secara manual?

## 1.3. Batasan Masalah

Berdasarkan topik permasalahan dalam penelitian pada tugas akhir ini, penulis memberikan batasan masalah yang meliputi :

1. pembuatan program dilakukan dengan menggunakan perangkat lunak Python;
2. metode optimisasi dan desain arsitektur CNN dibuat dengan modul Python bernama TensorFlow;
3. data yang digunakan berupa data seismogram dari Lapangan "X".
4. Tidak dilakukan *filter* sinyal pada data.

## 1.4. Tujuan Penelitian

Penelitian ini dilakukan dengan tujuan untuk

1. membuat program untuk *picking* seismogram secara otomatis sehingga dapat diperoleh parameter waktu tiba gelombang dan magnitudonya.
2. mengetahui hasil algoritma pada data seismogram gempa mikro di Lapangan "X" sehingga dapat digunakan untuk tahap penelitian selanjutnya.
3. menganalisis kemampuan algoritma dalam mengotomatisasi proses *picking* pada data seismogram.

## 1.5. Manfaat Penelitian

Adapun manfaat yang diperoleh dari penelitian ini adalah

1. menghasilkan program *open-source* yang diunggah ke *repository* dan dapat dimanfaatkan serta dikembangkan lebih lanjut oleh mahasiswa maupun akademisi Departemen Teknik Geofisika ITS.
2. menawarkan kolaborasi antara pihak akademisi dan industri untuk meningkatkan efisiensi dalam evaluasi sumber geotermal, khususnya pada tahap awal pengolahan data.
3. berkolaborasi dengan penelitian pemodelan geofisika yang memerlukan data *picking* sebagai masukan.
4. dapat digunakan sebagai referensi dalam penerapan kecerdasan buatan dan *big data* pada dunia geofisika baik oleh kalangan akademisi, industri, dan masyarakat.

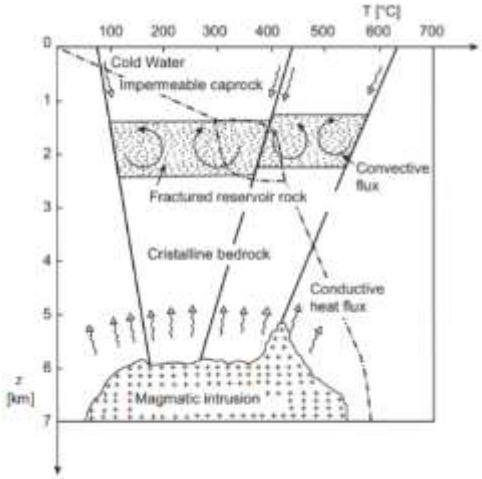
*Halaman ini sengaja dikosongkan*

# BAB II
# TINJAUAN PUSTAKA

## 2.1. Sistem Panas Bumi

Banyak sistem geotermal terkait erat dengan intrusi dan vulkanisme di sepanjang batas lempeng konvergen, *rift*, dan di *hotspot*. Area deformasi struktural aktif, terutama ekstensi dan transtension, melokalisasi intrusi dangkal. Karena intrusi tersebut, terbentuk patahan-patahan kecil yang membantu terjadinya konveksi fluida yang bersirkulasi di dalam sistem. Sistem tersebut kemudian menghantarkan panas dan aliran massa dari lokasi intrusi pada kedalaman> 3 km ke hidrosfer (Stimac, Goff, & Goff, 2015).

Agar konveksi termal pada sistem tersebut terjadi, kondisi tertentu harus dipenuhi. Aliran fluida, yang semata-mata disebabkan oleh perbedaan kerapatan yang ditimbulkan oleh gradien suhu, disebut konveksi termal atau alami. Meningkatnya suhu di sepanjang kedalaman mengurangi densitas fluida di bawah. Dengan perbedaan tersebut, persamaan hidrostatik harus dapat dipenuhi agar dapat terjadi keseimbangan antara gaya angkat fluida dan gaya gravitasi. Gambar 2.1 menjelaskan bagaimana aliran konveksi tetap terjaga di dalam batuan reservoir yang mengandung fluida. Gradien panas bumi yang tinggi juga diperlukan agar konveksi termal dapat terjadi. Nilai yang diperlukan biasanya lebih tinggi dari 0,2C/m. (Toth & Bobok, 2015)




**Gambar 2.1.** Sistem panas bumi pada umumnya (Toth & Bobok, 2015)

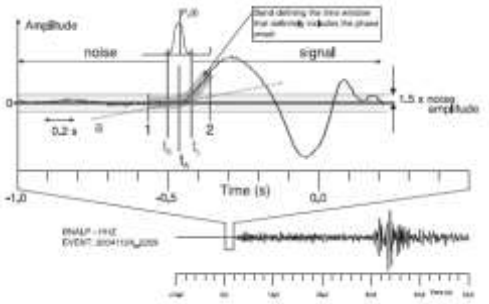
## 2.2. Teori Pengolahan Data Seismik Pasif

Metode seismic pasif menggunakan gelombang seismik alami. Gelombang frekuensi rendah ini bisa digunakan untuk pemantauan aktivitas gunung api, pemantauan patahan aktif, strategi mitigasi bencana dalam gempa bumi dan perkiraan bencana gempabumi, dan untuk pemantauan sistem panas bumi. Pada umumnya, data mikroseismik terekam dan terproses hampir sama dengan sinyal gempa tektonik. Namun, gempa tektonik umumnya memiliki magnitude di atas 5 Mw, sedangkan magnitudo MEQ di area panas bumi hanya kurang dari 3 Mw (Julian and Foulger, 2009).

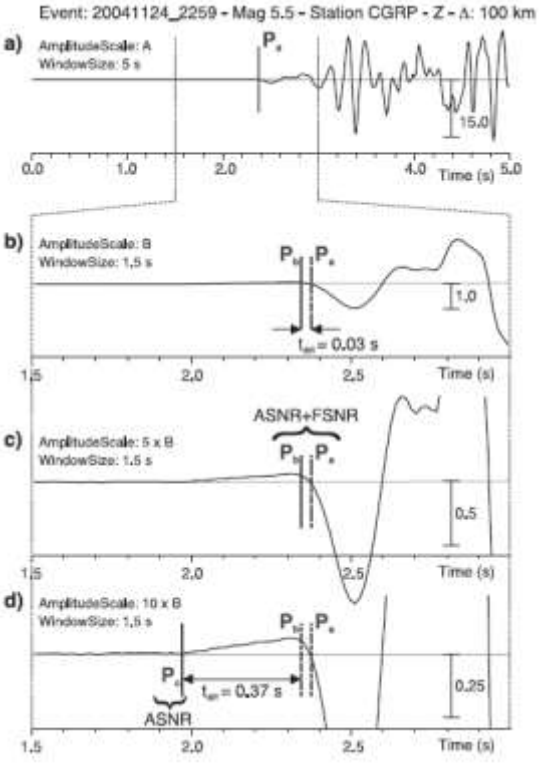
### 2.2.1. Penentuan *Picking* berdasarkan Amplitudo dan Frekuensi

Pada tahap pengolahan data MEQ lokal maupun regional, penentuan fase dan waktu tiba dapat menjadi inkonsisten dan ambigu karena pola gelombang yang kompleks dan waktu tiba yang sangat singkat antara gelombang P dan S. Pada umumnya, fase seismik didefinisikan oleh dua pengamatan visual berupa perubahan amplitudo dan/atau frekuensi. Contoh pada gambar 2.2 menjelaskan bagaimana suatu gelombang dapat ditentukan fasenya sebagai gelombang P. Gelombang P (*compressional*) yang merupakan *direct wave* maupun gelombang yang mengalami refraksi di Moho cenderung memiliki komponen dominan di arah vertikal, sedangkan gelombang S (*shear*) memiliki komponen utara-selatan atau timur-barat yang lebih dominan. Selain itu, pada umumnya, frekuensi gelombang P lebih besar daripada frekuensi gelombang S. Dengan memperhatikan parameter-parameter tersebut, dapat diketahui kapan suatu gelombang pertama kali tiba dan terekam. (Diehl & Kissling, 2007)

Kemudian, hal lain yang harus diperhatikan adalah skala amplitudo dan ukuran jendela (*window*) dalam interpretasi. Oleh karena itu, untuk meminimalkan subjektivitas, pilihan panjang *window* disesuaikan dengan frekuensi dominan sinyal dan juga *amplitude-based signal to noise ratio* (ASNR) yang ada. Jika terdapat banyak *noise* yang beramplitudo kuat atau sinyal yang lemah, pada umumnya lebih baik menggunakan *window* yang lebih lebar agar dapat menentukan perubahan frekuensi yang drastis tersebut. Penentuan skala amplitudo juga bergantung pada ASNR. Semakin banyak *noise* yang ada, interpretasi dengan menggunakan skala amplitudo kecil akan lebih baik. Hal ini mengurangi keambiguan *noise* menjadi sinyal, karena amplitudonya yang diskala lebih besar.




**Gambar 2.2.** Contoh cara *picking* waktu tiba pada suatu seismogram. (Diehl & Kissling, 2007)




**Gambar 2.3**. Perbedaan interpretasi *picking* karena perbedaan *window* dan skala amplitudo yang digunakan. (Diehl & Kissling, 2007)

### 2.2.2. *Filtering*

Menurut teori sinyal, representasi digital suatu sinyal kontinu bergantung pada *sampling frequency*-nya. Jika tidak, akan terjadi *aliasing* atau fenomena ketika suatu gelombang kehilangan karakternya dan menjadi mirip dengan gelombang lain. Untuk menghindari *aliasing*, sinyal dengan frekuensi $f$ harus di-*sampling* dengan *sampling frequency* sebesar:

$$\Delta f > 2f \tag{2.1}$$

sehingga frekuensi Nyquist-nya adalah:

$$fn = \frac{\Delta f}{2} \tag{2.2}$$

Untuk menghilangkan aliasing, data seismik biasanya difilter dengan analog filter anti-aliasing untuk menghapus frekuensi di atas frekuensi Nyquist. Filter biasanya diterapkan untuk meningkatkan SNR jika terdapat *noise*, sehingga diharapkan data akan lebih mudah diinterpretasi. Namun, pemfilteran dapat secara signifikan mengubah bentuk wavelet dan juga dapat menyebabkan pergeseran fase sinyal. Untuk menghindari hal tersebut, filter yang dilakukan secara dua arah pada sinyal diperlukan agar tidak terjadi pergeseran fase. Filter ini dapat berupa filter fase-minimum atau filter Butterworth orde kedua. Jika rentang frekuensi *noise* diketahui, maka dapat diterapkan filter *low-pass* atau filter *high-pass*. Kemudian dapat dibandingkan antara sinyal asli dan sinyal yang sudah di-*filter* sebelum dilakukan *picking*.

## 2.3. *Deep Learning*

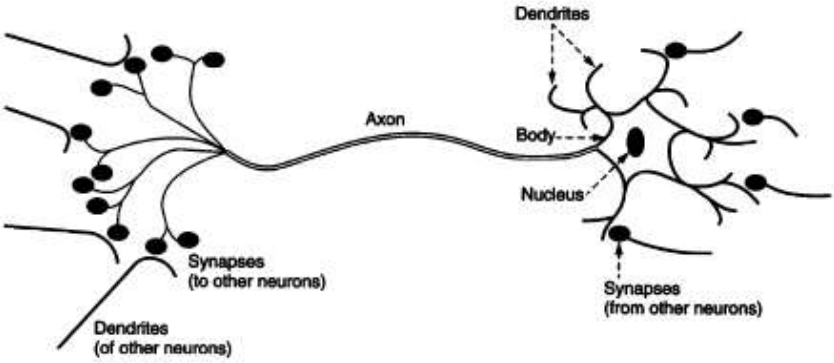
*Deep learning* adalah cabang dari *machine learning* dan *artificial intelligence* (AI) yang memprogram komputer untuk mempelajari struktur dan pola suatu data tanpa perlu diberi instruksi kode secara eksplisit. Dengan berkembangnya teknologi penyimpanan data, data yang dapat digunakan menjadi semakin banyak. Dari sinilah *deep learning* lebih dipilih untuk beberapa kasus, karena *deep learning* dapat mempelajari data yang berukuran besar dan jika kapasitas komputasi mencukupi.

Terdapat dua jenis *deep learning* berdasarkan ketersediaan informasi sebelumnya pada data: *unsupervised learning* dan *supervised learning*. *Unsupervised learning* berguna dalam mengkategorikan data berdasarkan pola yang mungkin ada, tapi belum diklasifikasi oleh manusia. Sedangkan *supervised learning* berarti mesin melakukan prediksi berdasarkan pilihan luaran yang diinginkan dan sudah disediakan pemrogram.

### 2.3.1. Arsitektur *Neural Network*

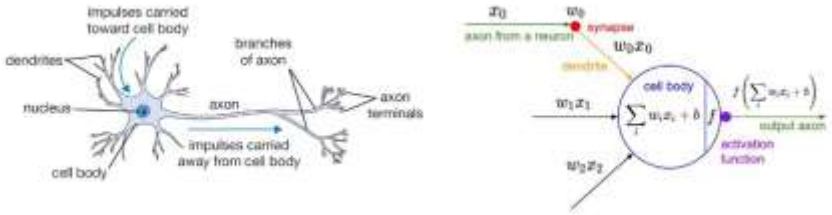
Sistem *deep learning* yang paling banyak digunakan adalah teknik yang mensimulasikan bagaimana otak manusia bekerja dalam memproses suatu informasi. Teknik ini dinamakan *artificial neural network* (ANN). Jaringan saraf manusia memiliki sel yang disebut juga sebagai neuron. Penghubung antarneuron tersebut adalah akson dan dendrit. Akson dan dendrit tersebut memiliki suatu ruang di antaranya yang disebut sebagai sinapsis. Di sinapsis

itulah informasi dijalarkan. Kemudian akson juga berkoneksi dengan neuron-neuron lainnya. Respon sinapsis ini berubah jika stimulus eksternal ini diubah-ubah (Mehrotra, Mohan, & Ranka, 1997) . Gambar 2.4 menunjukkan bagaimana skema jaringan neural otak manusia bekerja dan menjadi dasar bagi sistem otomatis yang akan digunakan.




**Gambar 2.4.** Jaringan neuron secara biologis (Mehrotra, Mohan, & Ranka, 1997)

Berdasarkan jaringan tersebut, dikembangkanlah ANN. Jika dianalogikan dengan sistem pada Gambar 2.4, ANN memiliki neuron berupa unit komputasi. Setiap input pada neuron diberi pembobotan, yang memengaruhi fungsi yang digunakan untuk menghitung pada unit tersebut. ANN menghitung sebuah fungsi dari input tersebut dengan meneruskan nilai komputasi tersebut dari neuron input ke neuron output dan menggunakan bobot sebagai parameter penghubung. Gambar 2.5 menunjukkan susunan umum ANN yang secara garis besar cukup mirip dengan jaringan neuron otak manusia.



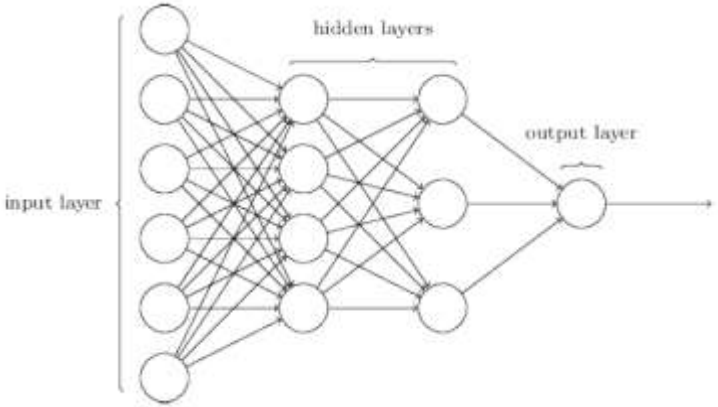

**Gambar 2.5.** Susunan ANN secara sederhana (sumber: Stanford Course)

Mesin dapat belajar dengan memodifikasi nilai bobot yang menghubungkan neuron-neuron. Seperti halnya rangsangan luar diperlukan untuk belajar pada otak manusia, ANN memperoleh rangsangan luar dari data

latihan (*training data*) yang memuat contoh-contoh dari masukan-luaran dari suatu data yang akan dipelajari. Sebagai contoh, data latihan memuat gambar-gambar sebagai input dan label yang diberikan pada gambar tersebut sebagai output, contohnya nama buah-buahan. Data latihan ini digunakan untuk melatih neuron untuk mengambil keputusan. Saat nilai galat atau ketidakpastian dalam prediksi masih sangat tinggi, dilakukan modifikasi nilai bobot agar pada iterasi berikutnya nilai galat dalam komputasinya semakin kecil. (Aggarwal, 2018)

### 2.3.2. Fungsi Aktivasi, *Loss Function*, dan Pembobotan

Meskipun skema ANN sudah dijelaskan di Gambar 2.4, pada kenyataannya terdapat beberapa lapisan (*layer*) yang diperlukan oleh mesin agar proses belajar menjadi lebih akurat. Gambar 2.5 menunjukkan konfigurasi ANN yang memiliki beberapa lapisan. Gambar 2.4 disebut *single layer perceptron* karena hanya memiliki satu lapisan. Sedangkan umumnya ANN berupa *multilayer perceptron* (MLP).




**Gambar 2.6.** Susunan MLP dengan dua lapisan tersembunyi (Nielsen, n.d.)

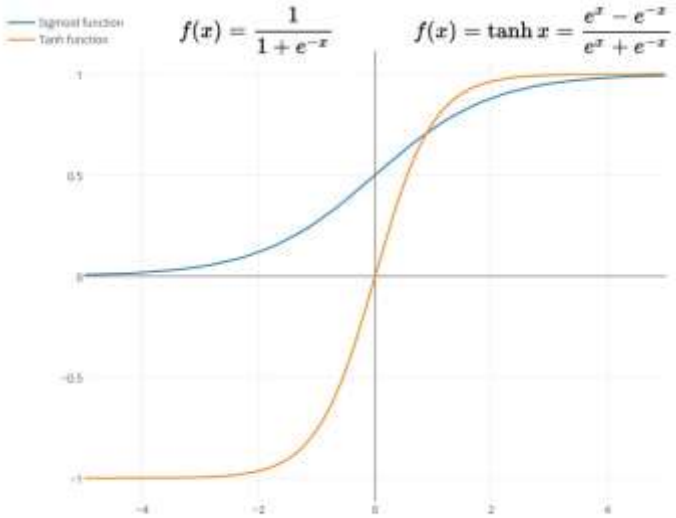
Pada setiap lapisan, terdapat pembobotan di tiap masukannya dan juga ada bias yang membantu mesin dalam menggeneralisir proses belajar. Pada setiap lapisan, tiap neuron menerima input ($x_i$) dan melakukan operasi dot dengan tiap bobotnya ($w_i$), menjumlahkannya (weighted sum) dan menambahkan bias ($b$).

$$\sum w_i x_i + b \tag{2.3}$$

Hasil dari operasi ini akan dijadikan parameter dari fungsi aktivasi yang akan dijadikan output dari neuron tersebut. Fungsi aktivasi ini befungsi untuk menentukan apakah neuron tersebut harus "aktif" atau tidak berdasarkan dari weighted sum dari input. Di fungsi aktivasi akan dibandingkan hasil

penjumlahan dengan *threshold* (nilai ambang) tertentu. Jika nilai melebihi *threshold*, maka aktivasi neuron akan dibatalkan, sebaliknya, jika masih dibawah nilai *threshold*, neuron akan diaktifkan. Fungsi aktivasi dapat berguna untuk memberi kompleksitas pada model. Dengan adanya fungsi-fungsi kompleks, dapat dilakukan proses belajar yang akurasinya lebih baik. Tanpa adanya fungsi aktivasi, ANN hanya menjadi regresi linear biasa.

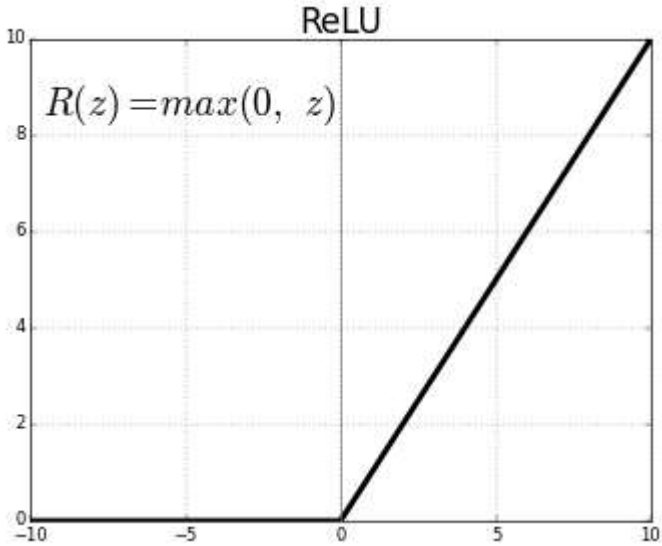
Terdapat beberapa jenis fungsi aktivasi yang umumnya digunakan. Tanpa modifikasi apa pun, fungsi aktivasi ANN adalah fungsi linear. Jenis fungsi aktivasi kedua adalah sigmoid dan tanh. Rentang nilai dari fungsi sigmoid, berdasarkan Persamaan 2.4, adalah 0 hingga 1, sedangkan rentang nilai fungsi tanh berdasarkan Persamaan 2.5 adalah -1 hingga 1. Gambar 2.6 mengilustrasikan fungsi aktivasi sigmoid dan tanh. Akan tetapi, masalah pada fungsi aktivasi sigmoid dan tanh adalah terdapat fenomena gradient yang sangat cepat mendekati nol (*vanishing gradient).* Hal ini terjadi karena fungsi sigmoid dan tanh dapat menekan nilai fungsi menjadi nol sehingga proses belajar ANN menjadi kurang efektif (Aggarwal, 2018). Untuk mengatasi hal ini, fungsi aktivasi yang sering digunakan adalah ReLU (*Rectified Linear Unit*) yang ditunjukkan oleh Gambar 2.7 dan Persamaan 2.6.




**Gambar 2.7.** Grafik fungsi aktivasi sigmoid dan tanh

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.5}$$




**Gambar 2.8.** Grafik fungsi aktivasi ReLU

$$f(x) = \max(0, x) \tag{2.6}$$

Lalu, hasil akan diteruskan ke output dan dihitung nilai galatnya. Nilai galat di sini berupa perbedaan antara nilai sesungguhnya dan nilai yang diprediksi oleh model ANN. Dalam dunia *deep learning*, fungsi ini seringkali disebut sebagai *loss function*. *Loss function* dihitung dengan persamaan berikut:

$$Loss = (Target - Prediction)^2 \tag{2.7}$$

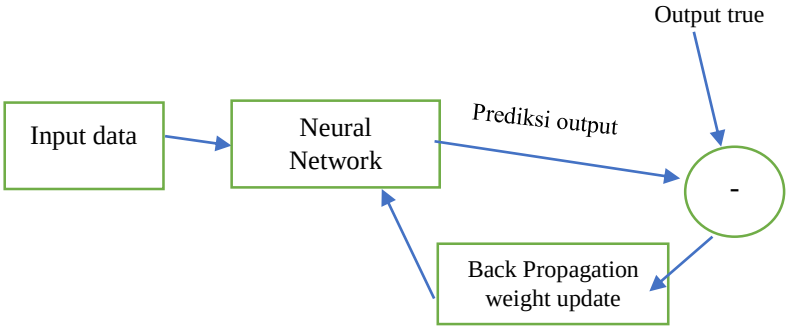
Persamaan inilah yang nilainya akan diminimalkan seiring *epoch* (iterasi pada seluruh data) diperbesar. Jika semua titik pelatihan (dari 1 hingga n) digunakan dalam perhitungan *loss function*, semua *loss function* dijumlahkan seperti pada Persamaan 2.8:

$$L = \sum_{i=1}^{n} L_i \tag{2.8}$$

### 2.3.3. *Forward Propagation* dan *Back Propagation*

*Forward pass* atau biasa juga disebut *forward propagation* adalah proses data pada input dibawa untuk melewati tiap neuron pada *hidden layer* sampai kepada *output layer* yang nanti akan dihitung galatnya. Error yang kita dapat pada *forward propagation* akan digunakan untuk meng-*update* setiap

weight dan bias dengan *learning rate* tertentu. Kedua proses berikut akan dilakukan berulang-ulang sampai didapatkan nilai *weight* dan bias yang dapat memberikan nilai galat sekecil mungkin pada *output layer* (pada *saat forward pass*). Hal ini dijelaskan lagi pada Gambar 2.8 yang mengilustrasikan tahap dalam melatih jaringan neuron.




**Gambar 2.9.** Skema kerja *training neural network*

Pada *back propagation*, dilakukan proses yang bernama *gradient descent.* Metode ini digunakan untuk meng-*update* nilai parameter bobot dan bias. Pada *gradient descent*, nilai *loss function* diminimalkan dengan menggeser parameter ke arah gradien negatif. Contohnya, jika terdapat *d* parameter bobot, maka parameter yang digunakan pada *gradient descent* adalah $\overline{W} = (w_1, w_2, w_3, \dots, w_n)$. Dengan demikian, secara standar, semua titik data digunakan untuk perhitungan *gradient descent* dengan persamaan sebagai berikut (Persamaan 2.9 dan disingkat menjadi Persamaan 2.10). Pada persamaan berikut, α adalah laju pembelajar (*learning rate*) yang menentukan seberapa cepat *gradient descent* berjalan.

$$\overline{W_{i+1}} = \overline{W_i} - \alpha\left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \dots, \frac{\partial L}{\partial w_d}\right) \quad (2.9)$$

$$\overline{W_{i+1}} = \overline{W_i} - \alpha \frac{\partial L}{\partial W} \quad (2.10)$$

Jika data sangat besar, cara untuk melakukan *gradient descent* adalah *mini-batch gradient descent*. Dengan cara ini, seluruh data dibagi menjadi beberapa kelompok yang berukuran sama. Hal ini membuat komputasi lebih ringan secara memori dan tidak terjebak pada minimum lokal. *Gradient descent* stokastik (SGD) juga digunakan dengan memilih sampel data secara acak dan meng-*update* parameter berdasarkan fungsi biaya.

Alternatif lain dari metode *gradient descent* adalah Adaptive Moment Estimation (ADAM). Optimisasi ADAM yang digagas oleh (Kingma & Ba,

2015) menggunakan momen pertama dan kedua gradien. Karena komputasinya didesain efisien, ADAM memerlukan lebih sedikit memori dan memiliki performa lebih baik pada sampel data yang besar. Untuk memulai algoritma optimisasi ini, diperlukan $p_0$ (vektor momen pertama seperti rata-rata), $q_0$ (vektor momen kedua seperti variansi), dan t (pencacahan waktu) diinisialisasi nilai 0. Jika dianggap f(w) adalah fungsi objektif stokastik dengan parameter w, nilai parameter ADAM yang disarankan adalah α = 0.001, $m_1 = 0.9$, $m_2 = 0.999$, $\in = 10^{-8}$. Keunggulan lain dari algoritma ADAM adalah parameter tidak bergantung pada pengubahan skala gradient yang digunakan. Hanya saja, karena diperlukan turunan kedua pada perhitungan, diperlukan fungsi biaya yang lebih besar. Algoritma ADAM dijelaskan sebagai berikut:

Langkah 1: hitung gradien $g_t = \frac{\partial f(x,w)}{\partial w}$ (2.10)

Langkah 2: hitung $p_t = m_1 . p_{t-1} + (1 - m_1) . g_t$ (2.11)

Langkah 3: hitung $q_t = m_2 . q_{t-1} + (1 - m_2) . g_t^2$ (2.11)

Langkah 4: hitung $\widehat{p_t} = p_t/(1 - m_1^t)$ (2.12)

Langkah 5: hitung $\widehat{q_t} = q_t/(1 - m_2^t)$ (2.13)

Langkah 6: update parameter $w_t = w_{t-1} - \alpha . \widehat{p_t}/(\sqrt{\widehat{q_t}} + \in)$ (2.14)

Jika $w_t$ konvergen

stop

Nilai laju pembelajar merupakan salah satu parameter yang dinamakan *hyperparameter*. Hiperparameter merupakan parameter yang ditentukan sebelum proses pelatihan ANN dimulai. Untuk memperoleh nilai hiperparameter yang optimal, dilakukan proses *tuning*. Salah satu cara adalah dengan *grid search* yang mencoba semua kemungkinan nilai hiperparameter pada suatu *grid*. Cara lainnya adalah dengan menggunakan optimisasi Bayesian (Snoek, et al., 2015).

## 2.4. *Convolutional Neural Network*

*Convolutional Neural Network* (CNN) atau seringkali disebut sebagai ConvNet adalah salah satu jenis *neural network* yang biasa digunakan pada data gambar. CNN bisa digunakan untuk mendeteksi dan mengenali fitur objek pada sebuah gambar. Alasan utama CNN lebih diminati dibandingkan model tradisional lainnya adalah dapat dilakukan pembagian pembobotan secara silang pada daerah yang tidak dilalui akson. Dengan demikian, parameter yang digunakan dalam proses pelatihan dapat dikurangi secara signifikan. Hal ini membantu dalam meningkatkan generalisasi model dalam mengenali data. Alasan kedua adalah CNN tidak mudah mengalami *overfitting*. Alasan terakhir

adalah CNN lebih mudah digunakan dibandingkan ANN untuk kumpulan data yang sangat besar. (Indolia, Goswami, Mishra, & Asopa, 2018)

Secara garis besar CNN tidak jauh berbeda dengan *neural network* biasanya. Yang membedakan CNN dengan ANN biasa adalah tambahan *convolutional layer* dan *pooling layer*. Terdapat medan reseptif yang terbentuk oleh adanya korelasi lokal antara satu neuron dengan neuron di *layer* sebelumnya. Dari medan reseptif ini, terbentuklah vektor bobot. Karena neuron pada suatu bidang memiliki bobot yang sama, fitur yang terletak pada lokasi berbeda di data masukan dapat dideteksi. Vektor bobot, atau filter/kernel, digeser pada vektor masukan untuk menghasilkan pemetaan fitur tersebut. Penggeseran ini dinamakan konvolusi. Luaran $a_{ij}$ pada *layer* berikutnya di lokasi *(i, j)* dihitung setelah melakukan perhitungan konvolusi pada Persamaan 2.11:

$$a_{ij} = \sigma((W * X)_{ij} + b) \qquad (2.11)$$

dengan X adalah masukan yang diberikan ke *layer* berupa tensor dengan dimensi (jumlah gambar × lebar gambar × tinggi gambar × kedalaman gambar), W adalah filter/kernel yang digunakan untuk konvolusi yang lebar dan tingginya adalah hiperparameter serta kedalamannya sama dengan kedalaman masukan, b adalah bias, dan $\sigma$ adalah faktor pemberi non-linearitas pada model (Maggiori, Tarabalka, Charpiat, & Alliez, 2017).

Selanjutnya adalah *pooling layer*. Setelah fitur pada suatu gambar terdeteksi, tempat suatu fitur tersebut terdeteksi menjadi tidak terlalu signifikan. Oleh karena itu, digunakan *pooling layer* untuk mengurangi jumlah parameter yang digunakan untuk pelatihan dan membuat pergeseran lokasi fitur pada gambar nantinya menjadi tidak signifikan pada prediksi fitur. Untuk melakukan operasi *pooling*, sebuah jendela dipilih dan elemen pada jendela tersebut diteruskan ke fungsi *pooling*. Pada umumnya, digunakan *max-pooling* untuk mengurangi penggunaan memori secara signifikan. Dari *layer* ini, dapat diperoleh vektor luaran yang kemudian diteruskan ke *layer* konvensional berikutnya. (Indolia, Goswami, Mishra, & Asopa, 2018)

*Halaman ini sengaja dikosongkan*

# BAB III
# METODOLOGI

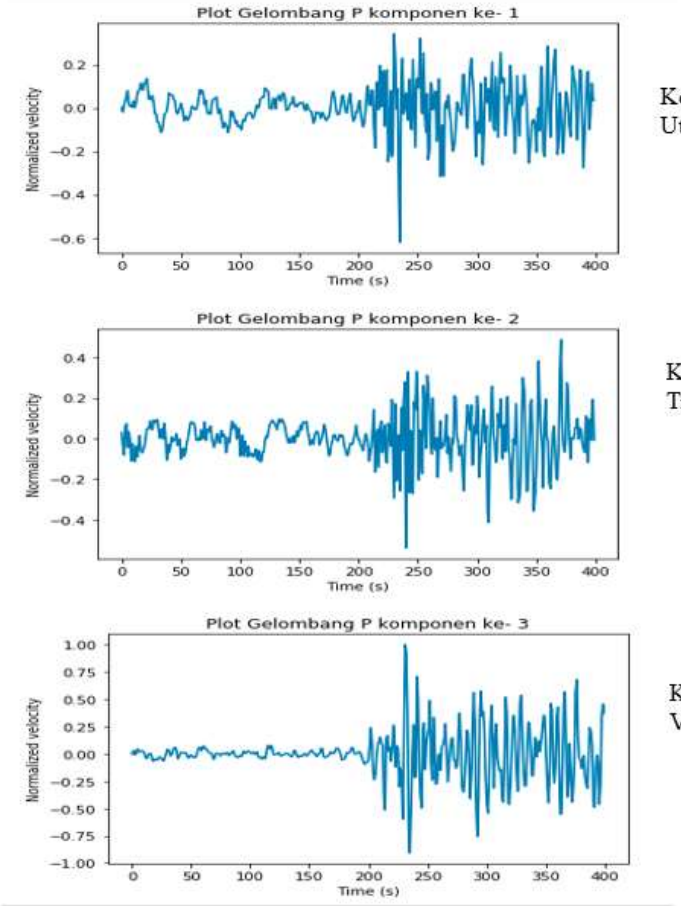
## 3.1. Data

Data seismogram yang digunakan pada penelitian ini adalah data riil di Lapangan "X" dengan format (*.msd). Seismogram ini terekam dalam 3 komponen (utara-selatan, timur-barat, dan vertikal). Data direkam dengan *sampling frequency* 500 Hz dan tiap *file* direkam dalam rentang waktu satu jam. Data yang digunakan akan disortir berdasarkan kenampakan event dalam masa perekaman data. Setelah dilakukan sortir data, selanjutnya adalah converting data dari (*.msd) menjadi (*.mseed).

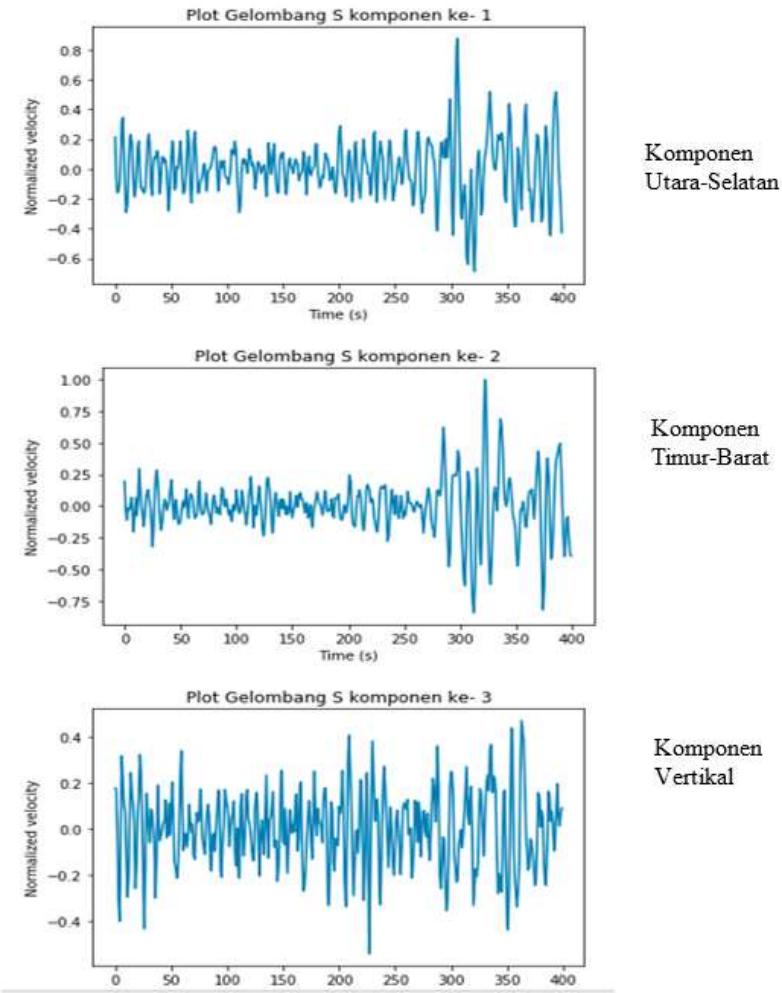
Untuk membuat model CNN sendiri, diperlukan data *training*. Data *training* yang digunakan berasal dari Southern California Seismic Network Catalog dengan jumlah 4.6 juta seismogram *velocity*. Data ini dibagi secara rata antara gelombang P, gelombang S, dan *noise*. Sampel data masing-masing memiliki durasi 4 sekon dan frekuensi *sampling* 100 Hz, sehingga terdapat 400 sampel. Data *training* ini menunjukkan bagaimana gelombang P, gelombang S, dan *noise* diklasifikasikan berdasarkan amplitudo suatu komponen seismogram relatif terhadap komponen yang lain. Pada Gambar 3.1 adalah contoh gelombang P yang memiliki amplitudo pada komponen vertikal yang lebih besar dibandingkan dengan komponen lainnya. Gambar 3.2 menunjukkan contoh gelombang S yang memiliki amplitudo pada komponen horizontal yang lebih besar dibandingkan dengan komponen lainnya. Sedangkan Gambar 3.3 menunjukkan contoh *noise* yang tidak memiliki pola tertentu.

Perangkat lunak untuk mengolah data yang digunakan pada penelitian tugas akhir ini adalah Python 3.7.3 (64 bit). Modul Python yang digunakan pada penelitian tugas akhir ini adalah Numpy, Pandas, Scipy, Matplotlib, Tensorflow, Keras, dan Obspy. Penelitian akan dijalankan pada perangkat keras laptop dengan spesifikasi:

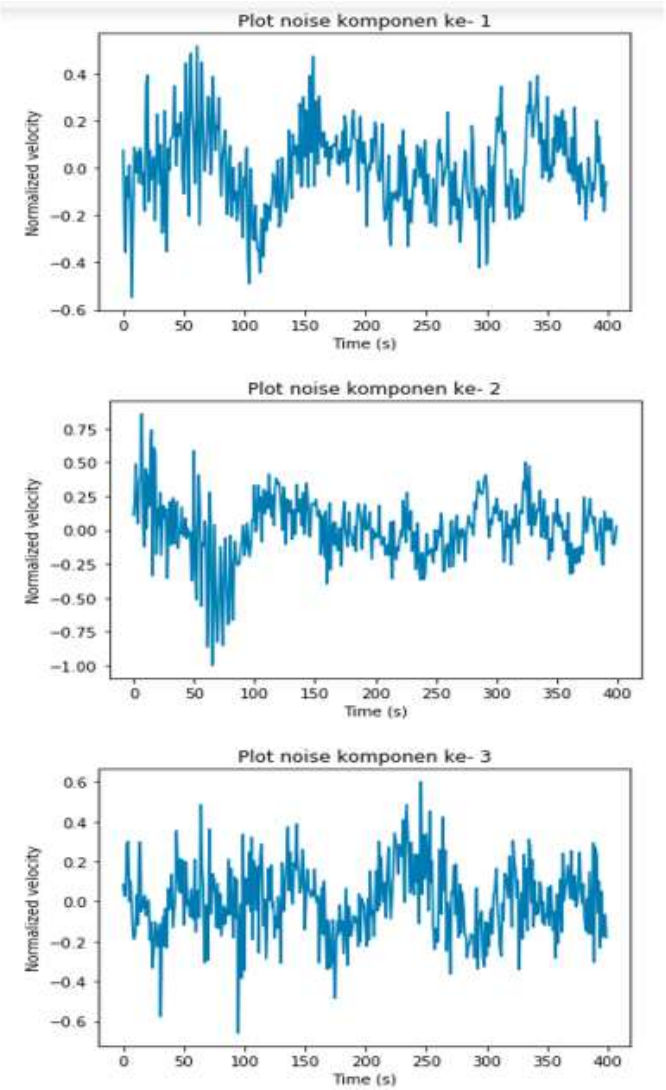
| | | |
|---|---|---|
| Operating System | : | Linux (Ubuntu 18.04) |
| Processor | : | Intel(R) Core(TM) i5-5200U @ 2.20 GHz |
| GPU | : | Nvidia 940M 2 GB |
| RAM | : | 8 Gb |




**Gambar 3.1.** Contoh gelombang P yang ditunjukkan pada 3 komponen seismogram




**Gambar 3.2.** Contoh gelombang S yang ditunjukkan pada 3 komponen seismogram



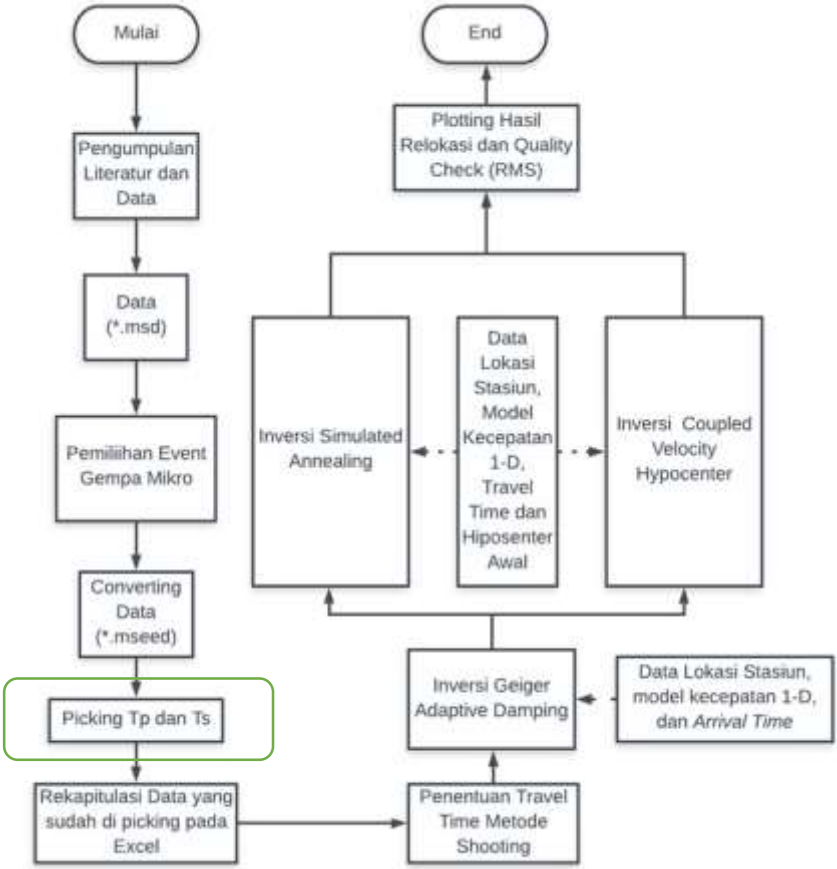

**Gambar 3.3.** Contoh *noise* yang ditunjukkan pada 3 komponen seismogram

## 3.2. Skema Kerja

### 3.2.1. Diagram Alir Penelitian secara Keseluruhan

Gambar 3.4 menunjukkan proses penelitian keseluruhan untuk melakukan *monitoring* panas bumi di Lapangan "X". Pada penelitian ini,

sebagaimana ditandai oleh kotak hijau, hanya dilakukan bagian *picking* untuk memperoleh nilai tp dan ts yang kemudian dapat digunakan untuk penelitian selanjutnya.

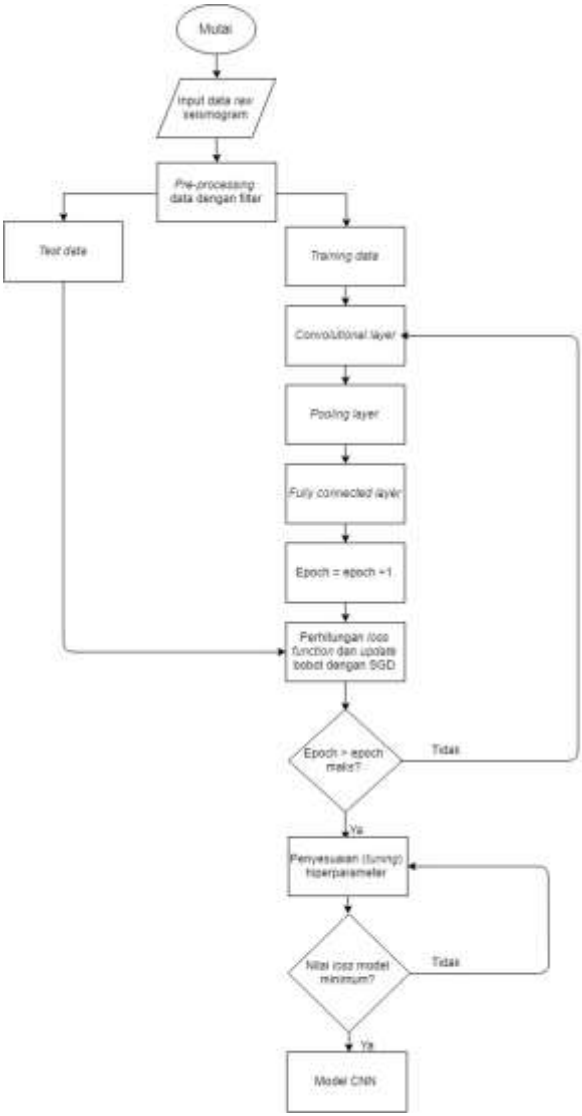



**Gambar 3. 4.** Diagram alir penelitian secara keseluruhan

### 3.2.2. Diagram Alir Pembuatan Model CNN

Pada proses penelitian kali ini, perlu dilakukan pembuatan model CNN terlebih dahulu, seperti dijelaskan pada Gambar 3.5. Sebelum membuat model, dilakukan *training-testing splitting* sejumlah 80%-20%. Kemudian, data *training* diteruskan ke *layer* konvolusi untuk mengekstrak fitur seismogram yang akan digunakan untuk membedakan fase seismogram, yaitu kontras amplitudo tiap komponen. Setelah itu diteruskan ke *layer neural network* biasa untuk

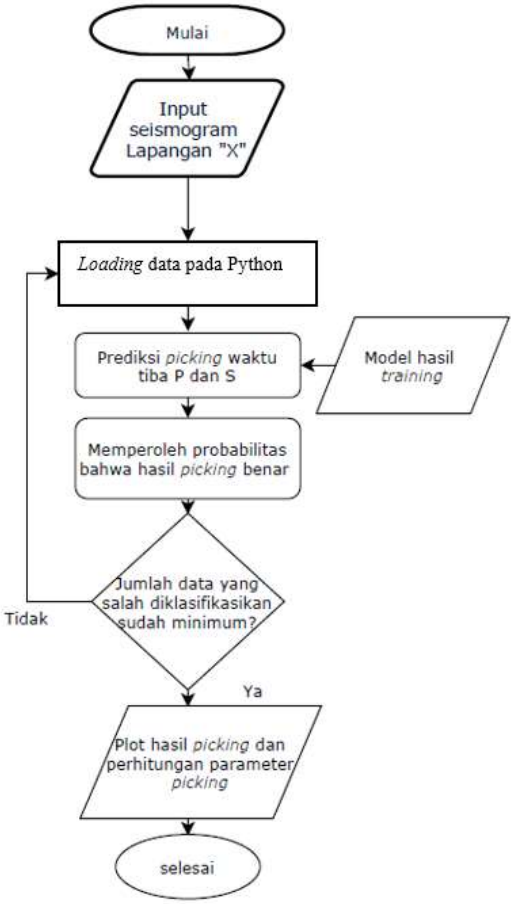
melakukan proses pembelajaran parameter. Proses ini kemudian diulangi hingga diperoleh konvergensi nilai akurasi dan *loss*. Data *testing* kemudian digunakan untuk menguji akurasi dan *loss* dan mengetahui apakah data sudah dapat digunakan untuk data di luar sampel data *training*.




**Gambar 3.5.** Diagram alir proses *training* CNN

### 3.2.3. Diagram Alir Deteksi Fase Seismogram dengan CNN

Pada proses *picking*, ada pula proses analisis tambahan. Jika sebuah *event* terdeteksi oleh 3 stasiun oleh lebih, *event* tersebut akan direkap untuk menjadi bahan analisis selanjutnya setelah dilakukan prosedur seperti pada Gambar 3.4 dan 3.5.




**Gambar 3. 6.** Diagram alir proses *picking* otomatis dengan model CNN

## 3.2. Alur Kerja

### 3.3.1. Proses *Training* CNN

Pada proses *training* CNN, pada awalnya dilakukan pengumpulan data *training*. Mengingat keterbatasan data mikroseismik yang tersebar secara gratis dan dapat dipublikasikan, maka digunakan data gempa secara umum. Pada data *training* tersebut, tidak hanya ada gempa mikroseismik, namun juga gempa teleseismik, gempa tektonik biasa, dan juga aktivitas manusia lainnya.

Di tahap selanjutnya, dilakukan pembuatan *script* Python yang bertujuan untuk melakukan *training* CNN menggunakan data tersebut. Tujuannya adalah untuk diperoleh model yang kemudian dapat digunakan untuk mendeteksi fase gelombang P dan S pada seismogram Lapangan "X".

### 3.3.2. Proses *Picking* Otomatis dengan Model CNN

Tahap selanjutnya adalah melakukan prediksi *picking* pada data seismogram Lapangan "X". Hal pertama yang perlu diperhatikan adalah pembuatan *sliding window*, karena data Lapangan "X" memiliki durasi perekaman 3600 sekon yang jauh lebih besar dari durasi data *training*. *Sliding window* ini akan membantu sehingga prediksi dilakukan secara bertahap pada data, tidak langsung secara keseluruhan. Kemudian, dari *script* Python yang sudah ada, dilanjutkan sehingga dapat membaca baik model yang sudah tersimpan di perangkat komputer maupun data seismogram Lapangan "X". Setelah *sliding window* diimplementasikan pada data, barulah dapat dilakukan prediksi tp dan ts pada data.

Secara spesifik, tahap *input* yang dilakukan adalah memasukkan nama *file* tiap data dengan format:

Nama file seismogram_komponen utara-selatan.mseed Nama file seismogram_komponen timur-barat.mseed Nama file seismogram_komponen vertikal.mseed

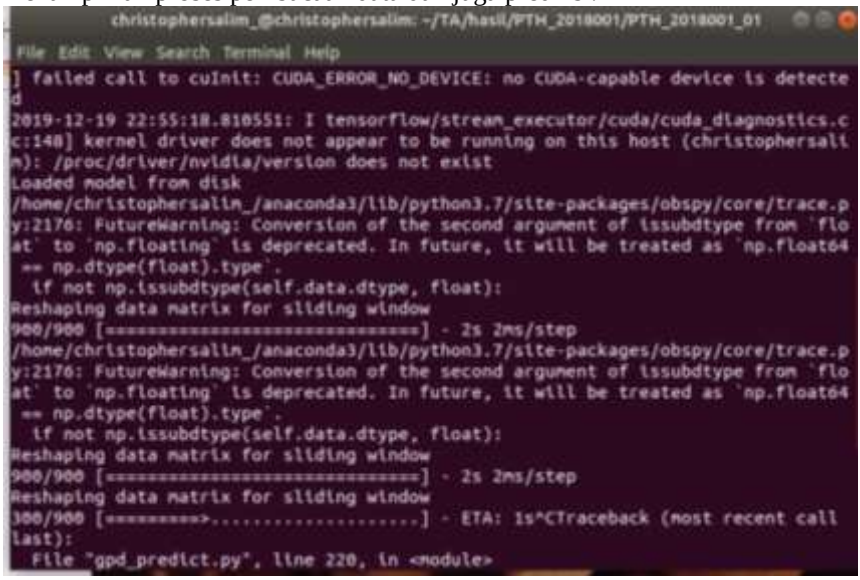
Jika ingin menambahkan data di waktu yang sama dengan stasiun berbeda, cukup mengikuti format yang sama dan dimasukkan ke baris yang baru. Contoh ada pada Gambar 3.7.

**Gambar 3.7.** Contoh *input* file seismogram untuk dibaca pada prediksi tp-ts.

Kemudian, untuk melakukan *running* program, dapat dilakukan di *command window* (jika OS Windows) atau terminal (jika OS Linux atau Mac). Format untuk *running* adalah sebagai berikut:

./path/to/directory/python **namascript**.py -I testing.in -O results.out

Jika sudah berhasil, akan ditemui hasil seperti pada Gambar 3.8 yang menampilkan proses pembacaan data dan juga prediksi.




**Gambar 3.8.** Contoh *running* pada terminal Linux yang berhasil membaca *script* dan data.

*Halaman ini sengaja dikosongkan*

# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1. Hasil

### 4.1.1. Hasil Pembuatan Model CNN

Pembuatan *script* Python telah berhasil dilakukan untuk membuat model CNN (Lampiran 1). Pada *script* yang telah dibuat, dilakukan tahap-tahap berupa *import library*, memasukkan data *training* yang awalnya berformat hdf5 ke Python, analisis awal beberapa data. Data *training* kemudian dipisah menjadi data *training* dan data *testing*. Sebelum proses *training* dan *testing*, *script* melakukan pembuatan model yang susunannya ditunjukkan di Tabel 4.1:

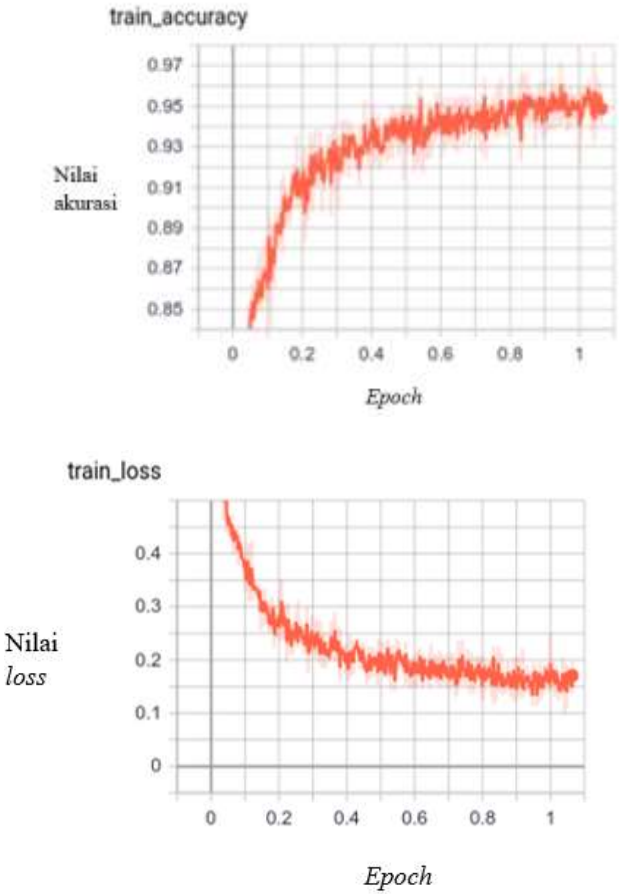
**Tabel 4.1.** Susunan model CNN yang dibuat

**Model: "sequential_22"**

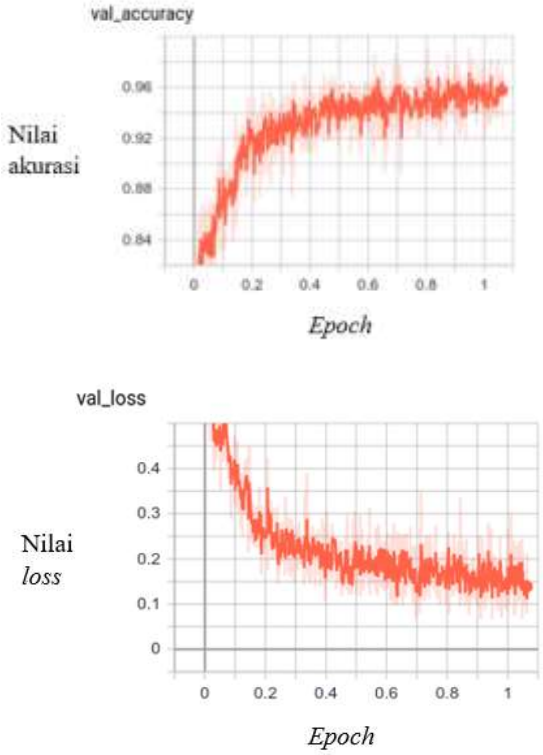
| **Layer (type)** | **Output Shape** | **Param #** |
|---|---|---|
| **conv1d_137 (Conv1D)** | **(None, 391, 64)** | **1984** |
| **conv1d_138 (Conv1D)** | **(None, 382, 64)** | **41024** |
| **max_pooling1d_49 (MaxPooling** | **(None, 127, 64)** | **0** |
| **conv1d_139 (Conv1D)** | **(None, 118, 32)** | **20512** |
| **conv1d_140 (Conv1D)** | **(None, 109, 32)** | **10272** |
| **global_average_pooling1d_31** | **(None, 32)** | **0** |
| **dropout_31 (Dropout)** | **(None, 32)** | **0** |
| **dense_39 (Dense)** | **(None, 3)** | **99** |

**Total params: 73,891**
**Trainable params: 73,891**
**Non-trainable params: 0**

Kemudian, dilakukan proses *training* dan *testing* menggunakan data dan model yang ada. Hasil dari *training* dan *testing* ini menghasilkan nilai *accuracy* dan *loss* yang ditunjukkan pada Gambar 4.1 dan Gambar 4.2. Yang terakhir, *script* menyimpan model, baik arsitektur maupun bobot pada masing-masing *layer*, agar dapat digunakan pada tahap selanjutnya.




**Gambar 4. 1.** Hasil evaluasi *running* CNN (*accuracy* dan *loss*) pada data *training.* Pada akhir iterasi, nilai akurasinya adalah 0.95 dan *loss*-nya dalah 0.17.




**Gambar 4. 2.** Hasil evaluasi *running* CNN (*accuracy* dan *loss*) pada data *testing.* Pada akhir iterasi, nilai akurasinya adalah 0.94 dan *loss*-nya dalah 0.14.

### 4.1.2. Hasil Prediksi *Arrival Time*

Model yang telah dibuat kemudian digunakan untuk memprediksi *arrival time* berupa tp dan ts. Seperti yang terlihat pada Gambar 4.3, digunakan batasan probabilitas minimum untuk dapat mendeteksi fase gelombang adalah 0.95, jumlah titik data pada tiap pergeseran adalah 400 titik data. *Script* lengkap dari prediksi *arrival time* dapat dilihat pada Lampiran 2.

```
#####################
# Hyperparameters
min_proba = 0.95 # Minimum softmax probability for phase detection
freq_min = 6.0
freq_max = 12.0
filter_data = False
decimate_data = True # If false, assumes data is already 100 Hz samprate
n_shift = 400 # Number of samples to shift the sliding window at a time
n_gpu = 0 # Number of GPUs to use (if any)
#####################
batch_size = 300

half_dur = 2.0
only_dt = 0.01
n_win = int(half_dur/only_dt)
n_feat = 2*n_win

#----------------------------------------------------------------
```

**Gambar 4. 3.** Cuplikan *script* untuk mendeteksi tp dan ts.

Setelah *script* di-*run* sesuai dengan cara yang telah dideskripsikan pada Bab 3, diperoleh hasil dalam format csv yang memuat stasiun, waktu tiba gelombang P yang dibaca, dan waktu tiba gelombang S (Gambar 4.4.)

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Hari | Jam | Station | tp (s) | ts (s) | | | |
| 2 | 1 | 0 | B13C | [1324. 1328. 3504. 3532. 3544.] | [1104. 2780.] | | | |
| 3 | | | B14E | [] | [] | | | |
| 4 | | | B157 | [2288. 2916.] | [] | | | |
| 5 | | 1 | B13C | [] | [] | | | |
| 6 | | | B14E | [864. 868.] | [] | | | |
| 7 | | | B157 | [] | [] | | | |
| 8 | | 2 | B13C | [] | [ 240. 2040.] | | | |
| 9 | | | B14E | [] | [] | | | |
| 10 | | | B157 | [] | [] | | | |
| 11 | | 3 | B13C | [] | [ 88. 252. 528. 748. 940. 1072. 1808.] | | | |
| 12 | | | B14E | [] | [] | | | |
| 13 | | | B157 | [] | [] | | | |
| 14 | | 4 | B13C | [] | [] | | | |

**Gambar 4. 4.** Cuplikan hasil prediksi *picking* yang telah ditabulasi.

## 4.2. Pembahasan

### 4.2.1. Analisis Kualitas Data *Training*

Berdasarkan pengamatan visual terhadap data *training* yang telah ada, data telah mengikuti label yang telah diberikan. Pada proses *picking*, perlu diperhatikan kaidah *picking* yang tepat. Jika dilihat dari Gambar 3.1 hingga 3.3, terlihat bahwa label sudah sesuai. Pada Gambar 3.1, amplitudo maksimal pada komponen vertikal mencapai hampir 1, sementara pada komponen lainnya amplitudo maksimumnya tidak terlalu signifikan. Waktu ketika terjadi amplitudo maksimum antara ketiga komponen juga konsisten, sehingga dapat dikatakan bahwa fase tersebut adalah gelombang P. Hal yang sama juga dapat dikatakan untuk Gambar 3.2 yang memiliki amplitudo di komponen horizontal yang lebih signifikan daripada komponen vertikalnya. Hal ini dapat dijelaskan oleh sifat gelombang P dan gelombang S.

Kemudian, sebagaimana yang telah diteliti oleh Ross et al (2018), data yang diberi label sebagai *noise* merupakan data *noise* pra-*event* gempa. Data *training* yang disediakan oleh Southern California Earthquake Data Center tidak terlalu banyak mengandung data *noise* yang berasal dari aktivitas manusia (pergerakan manusia, kendaraan bergerak, angin, dan lain sebagainya). Terlepas dari hal itu, data *noise* yang ada sudah cukup merepresentasikan bagaimana karakter *noise* yang direkam oleh seismogram. Salah satu contoh karakteristiknya adalah nonstasioneritas pada data, seperti pada Gambar 3.3.

Secara garis besar, data *training* ini memberi gambaran kepada peneliti apa saja parameter yang akan dipelajari oleh model CNN dan apa saja yang dapat dikembangkan pada penelitian selanjutnya. Pada data ini, parameter utama yang dipelajari oleh model adalah perbandingan amplitudo maksimum pada suatu komponen dibandingkan pada komponen yang lain. Dari parameter tersebut, dapat diperoleh apakah gelombang tersebut termasuk gelombang P, S, atau *noise*.

### 4.2.2. Analisis Model CNN

*Script* dan model CNN ini dapat dianalisis dalam berbagai sudut pandang. Sebelum membahas performa model, hal pertama yang perlu dibahas adalah *layer* apa saja yang digunakan dalam model dan jumlah parameter yang akhirnya diperoleh. Pada *layer* pertama adalah *layer input* yang berupa *layer convolutional* untuk melakukan proses konvolusi. Kemudian *layer* kedua juga berfungsi sama. Lalu ada juga *maxpooling* untuk membuat agar model lebih baik dalam menangkap anomali fitur, tidak bergantung pada amplitudonya. Dua *layer* selanjutnya juga merupakan *layer* konvolusi, untuk memastikan semakin banyak parameter yang dipelajari. Lalu ada juga *global pooling layer* untuk menggabungkan hasil semuanya. Selanjutnya adalah *dropout layer* untuk menghindari *overfitting*, dan yang terakhir adalah *fully connected neural network* yang berfungsi sebagai *neural network* biasa dan menghasilkan luaran berupa probabilitas jenis fase gelombang apakah gelombang P, S, atau *noise*.

Sudut pandang pertama yang digunakan untuk menganalisis model CNN ini adalah dalam segi jumlah parameternya. Secara teori, semakin banyak parameter, berarti semakin kompleks *layer* yang digunakan dan semakin banyak fitur yang dipelajari dari suatu seismogram. Jumlah parameter ini dapat disesuaikan terutama dengan mengubah ukuran kernel yang digunakan dalam proses konvolusi dan jumlah filter yang digunakan. Idealnya, semakin banyak jumlah parameter yang digunakan, semakin baik performa model ini. Namun, setelah melalui beberapa kali percobaan, ditemukan bahwa jumlah parameter sebanyak 73.891 adalah jumlah maksimum yang dapat ditangani oleh kemampuan komputasi saat ini.

Jumlah filter memengaruhi ukuran fitur yang akan dipelajari dalam suatu data. Secara teori, tidak ada ukuran yang pasti untuk menentukan pastinya nilai filter, karena model dibuat bergantung pada data yang karakteristiknya berbeda-beda. Jika filter dibuat lebih kecil, diasumsikan data yang ingin dipelajari memiliki fitur atau parameter yang lebih detail, sehingga filter tersebut baik dalam mendeteksi perubahan kecil pada data. Akan tetapi, jika ingin diperoleh fitur yang lebih luas, digunakan filter yang besar. Hal ini yang menyebabkan jumlah filter yang digunakan pada model ini besar. Fitur anomali amplitudo merupakan fitur yang lebih regional dalam suatu data, karena kita tidak ingin *noise* yang merupakan fitur lebih detail pada seismogram juga terdeteksi. Pada akhirnya, jumlah filter ini juga secara langsung memengaruhi kedalaman dari jaringan otak yang digunakan pada model ini. Karena data tereduksi dengan cepat oleh filter yang besar, model juga kurang "*deep*" dibandingkan dengan model dengan filter kecil. Hal ini juga memperhatikan kemampuan komputasi dari komputer yang dipunyai maupun komputer lain yang dapat diakses.

Hal kedua yang dapat dipelajari dari model ini adalah perlunya menggunakan *dropout layer*. *Dropout layer* adalah salah satu jenis *layer* pada CNN yang fungsinya adalah untuk menghindari *overfitting*. *Overfitting* dapat terjadi ketika model terlalu detail dalam mempelajari data sehingga *noise* pada data juga terlalu dipelajari dalam model. Secara mudahnya, model menjadi terlalu kompleks sehingga akurasi pada proses *training* baik namun akurasi pada proses *test* jauh lebih buruk. Untuk menghindari hal tersebut, digunakan *dropout layer* dengan *dropout rate* 0.1. Artinya, untuk setiap iterasi, sebanyak 10% dari total neuron input dinonaktifkan secara acak dan proses berjalan seperti biasa. Hal ini mengurangi sensitivitas neuron terhadap *noise* yang dijumpai pada data *training*. Nilai *dropout rate* sendiri tidak memiliki patokan yang pasti, bergantung pada tingkat *noise* pada data dan kebutuhan model.

Kemudian, pada *script* diputuskan untuk melakukan pemisahan *training* dan *testing*. Pemisahan ini dilakukan agar model memiliki validasi apakah sudah dapat memprediksi fase gelombang meski diberi data yang berbeda dengan data *training*. Rasio pemisahan *training-testing* dari model ke model berbeda, namun pada kali ini digunakan pemisahan 80% untuk data *training* dan 20% untuk data

*testing*. Hal ini dilakukan untuk memastikan bahwa data *training* cukup banyak untuk model dapat mempelajarinya, dan di saat yang sama terdapat cukup banyak data untuk memvalidasi model tersebut. Idealnya, digunakan jumlah data *training* yang lebih besar daripada data *testing*. Hal ini karena semakin banyak data *training*, semakin banyak fitur yang dapat dipelajari sehingga model dapat lebih mudah menggeneralisasi. Akan tetapi, juga diperlukan sebagian *test* data untuk menilai, sehingga digunakan rasio tersebut. Rasio ini berdasarkan prinsip Pareto (Chopra et al, 2019).

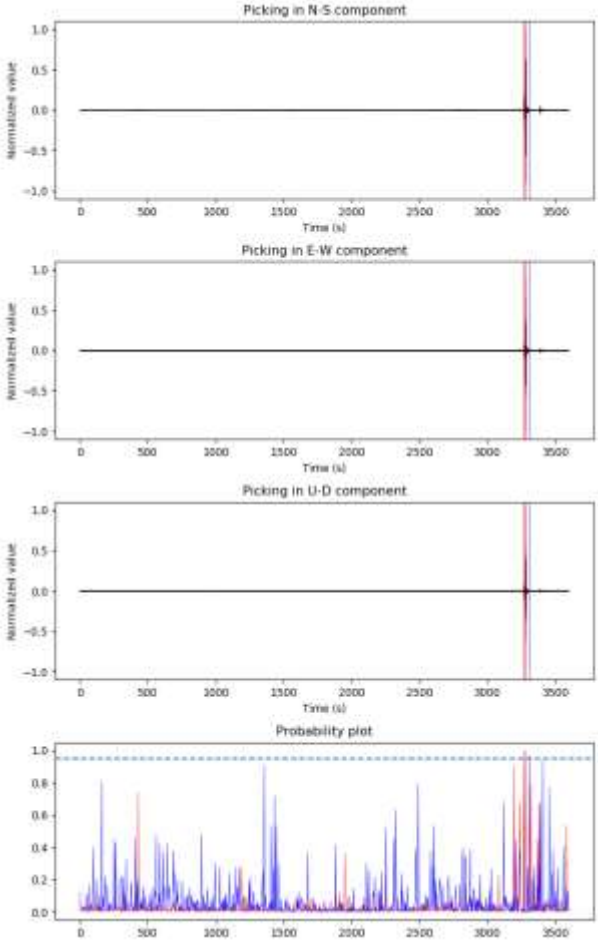
Setelah dilakukan pemisahan *training-testing*, proses *training* diimplementasikan dengan metode *train on batch*. Hal ini berarti data *training* dibagi-bagi menjadi beberapa kelompok agar proses komputasi menjadi lebih cepat dan memakan lebih sedikit memori. Semakin kecil ukuran sampel per *batch*, semakin mudah proses komputasinya. Pemilihan berapa sampel per *batch* sendiri tidak memiliki patokan baku. Jika kita ingin komputasi yang lebih cepat, tapi hasil yang kurang detail dibandingkan dengan *batch* yang memiliki banyak sampel, digunakan jumlah sampel per *batch* yang kecil. Hal ini disebabkan oleh semakin banyak sampel pada setiap *batch*, semakin banyak parameter yang dipelajari pada setiap iterasi untuk satu *epoch*.

Selanjutnya, pada *script*, dapat dilihat pula bagaimana progres kenaikan akurasi dan penurunan *loss* pada model. Idealnya, akurasi selalu naik di setiap *batch* dan *loss* semakin kecil. Jika hal tersebut tidak terjadi, perlu dilakukan perbaikan model maupun proses *input* data. Pada proses *training* dan *testing* yang telah dilakukan, diperoleh akurasi dan *loss* yang baik. Secara bentuk kurva, keduanya menunjukkan bentuk eksponensial, yang berarti baik nilai akurasi maupun *loss*-nya mencapai suatu titik jenuh pada iterasi tertentu. Dari hasil ini, dapat ditemukan bahwa baik *training* maupun *testing* sudah mengalami konvergensi. Nilai akurasi *training* dan *testing* tidak berbeda jauh. Dengan demikian, model sudah mengalami proses *fitting* yang sesuai pada proses pembelajarannya. Jika hasil *training* maupun *testing* kurang baik, kemungkinan besar terjadi *underfitting* atau model menjadi terlalu sederhana untuk menjelaskan atau memprediksi data selanjutnya. Jika hasil *training* jauh lebih baik secara signifikan daripada hasil *testing*, kemungkinan besar terjadi *overfitting*. Pada hasil ini, model sudah memiliki hasil yang sangat baik pada data *training* maupun *testing*.

### 4.2.3. Analisis Kualitatif *Picking* Otomatis pada Data Lapangan "X"

*Script* yang digunakan untuk memprediksi *arrival time* merupakan modifikasi dari *script* yang telah dibuat oleh Ross et al (2018). Pada data Lapangan "X", durasi perekaman adalah 3600 detik dan frekuensi akhirnya adalah 100 Hz, sehingga terdapat 360000 titik data. Seperti yang telah dijelaskan oleh subbab 3.3.2, *sliding window* digunakan untuk melakukan prediksi fase seismogram secara bertahap.

Secara sekilas, jika dilihat dari hasil *picking* otomatis, terdapat beberapa kemiripan. Sebagai contoh, hasil *picking* divisualisasikan oleh Gambar 4.4. Pada hasil visualisasi, terdapat seismogram tiga komponen, *picking* P yang ditandai oleh warna merah dan *picking* S yang ditandai oleh warna biru, sedangkan pada baris keempat terdapat probabilitas atau tingkat kebolehjadian apakah fase suatu gelombang pada waktu tertentu adalah P atau S. Digunakan batasan probabilitas 0.95 yang berarti jika tingkat probabilitas suatu gelombang lebih dari atau sama dengan 0.95, titik pada waktu tersebut akan ditandai sebagai gelombang P atau S.




Keterangan:

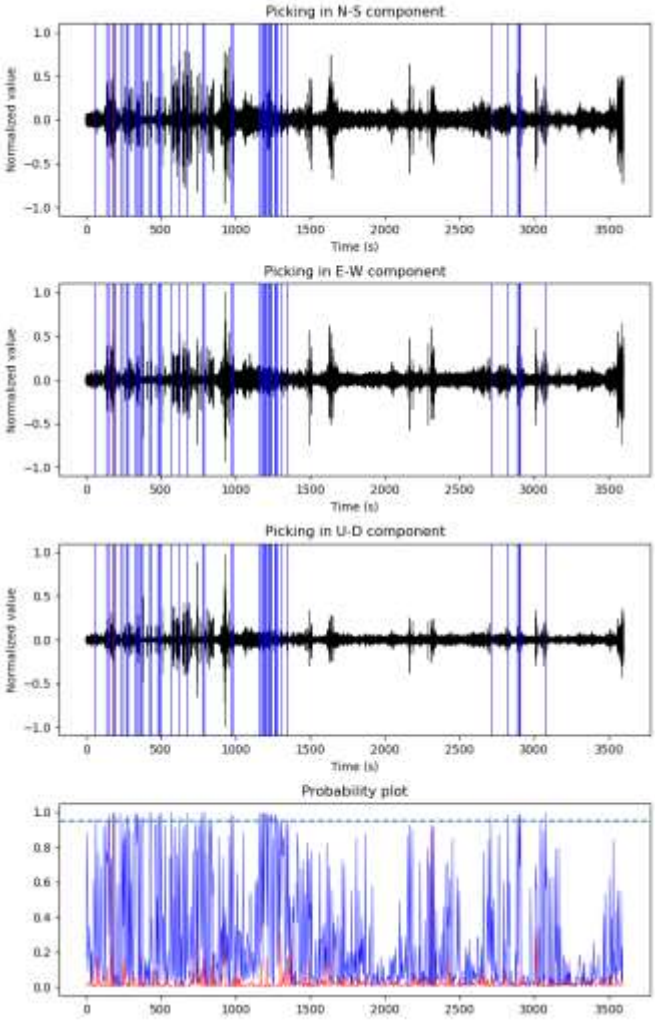
: *Picking* P

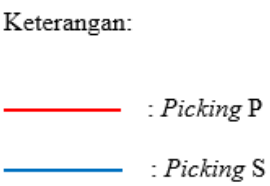
: *Picking* S

**Gambar 4.5.** Contoh hasil *picking* otomatis dengan menggunakan model CNN pada hari 1 jam 22.00 hingga 23.00.

Pada Gambar 4.4 di atas, jika dilihat secara waktu tiba gelombang P dan gelombang S, diperoleh nilai tp dan ts-nya masing-masing 3276 s dan 3304 s sedangkan hasil *picking* manual menandakan nilai tp dan ts-nya masing-masing 3266.138 s dan 3266.61 s. Jika dibandingkan, hasilnya cukup mirip. Akan tetapi, hal yang menjadi perhatian lebih adalah ketika perbedaan antara tp dan ts yang lebih besar dari 3 sekon. Padahal, menurut penelitian-penelitian sebelumnya, ciri khas *event* mikroseismik salah satunya adalah memiliki selisih tp dan ts kurang dari 3 sekon. Hal ini dapat disebabkan oleh *window* yang digunakan untuk memprediksi tp dan ts menggunakan model CNN. Jika *sliding window* diperkecil, selisih antara tp dan ts bernilai kurang dari 3 sekon tersebut dapat diperoleh karena model menjadi lebih detail dalam menganalisis tiap bagian jendela waktu pada data, namun *noise* yang memiliki lonjakan amplitudo kecil juga dapat terdeteksi sebagai *event*, memperburuk kualitas *picking* secara keseluruhan. Akan tetapi, jika *sliding window* terlalu besar, *event* yang lebih detail tidak dapat terbaca oleh model. Oleh karena itu, pada penelitian kali ini dilakukan pengambilan panjang window sejumlah 400 sampel per *window* agar sesuai dengan panjang sampel pada data *training* maupun *testing*. Diharapkan hal ini dapat memaksimalkan akurasi *picking*. Akan tetapi, berdasarkan hasil di atas, meski selisih waktu tidak sesuai dengan kaidah mikroseismik, rentang terjadinya *event* mikroseismik sudah tepat diprediksi oleh model CNN.

Hal selanjutnya yang dapat dianalisis oleh *picking* otomatis ini adalah hasil *picking* otomatis yang banyak menjumpai *event* gelombang P dan atau S pada suatu data, meski pada *picking* manual tidak dijumpai *event*. Contoh yang dapat dilihat ada pada Gambar 4.5. Pada gambar tersebut, terlihat banyak bagian yang kemungkinan besar diduga sebagai *noise*, namun dideteksi sebagai gelombang S oleh model karena tingkat probabilitas gelombang S-nya yang tinggi. Penyebab pertama menurut analisis penelitian ini adalah *surface wave* yang menyerupai gelombang S pada data ini. Dapat dilihat pada 1500 sekon pertama bahwa banyak sekali lonjakan amplitudo pada komponen utara-selatan relatif terhadap komponen yang lain. Penyebab kedua adalah faktor instrumen pengukuran yang lebih sensitif terhadap *noise*. Penyebab ketiga adalah data terlalu panjang dibandingkan dengan data *training* yang hanya memiliki panjang 400 sekon. Hal ini menyebabkan proses prediksi menjadi sedikit lebih detail daripada yang seharusnya. Oleh karena itu, juga digunakan parameter pembatas tambahan yaitu *event* harus terdeteksi di tiga stasiun atau lebih. Batasan ini juga berdasarkan fakta bahwa metode tradisional untuk mendeteksi *event* adalah membuat lingkaran di sekitar setiap stasiun yang mengenai titik *event*. Jika titik *event* disinggung oleh ketiga lingkaran, tingkat keyakinannya adalah yang paling tinggi.

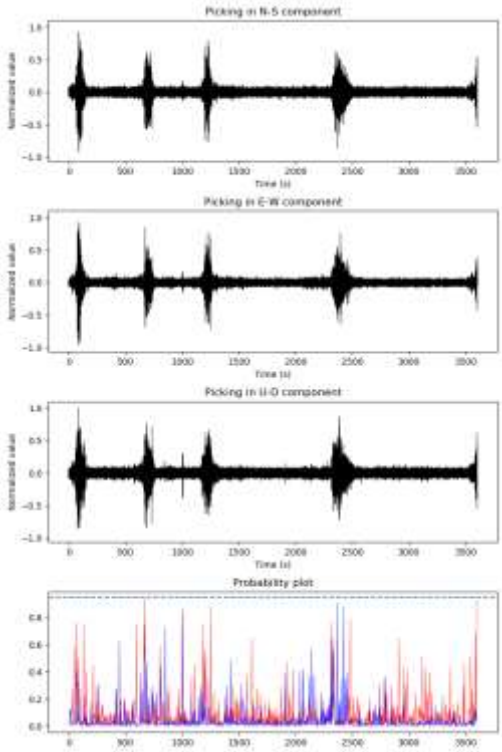

Picking in N-S component
Normalized value
Time (s)
Picking in E-W component
Normalized value
Time (s)
Picking in U-D component
Normalized value
Time (s)
Probability plot




**Gambar 4.6.** Contoh hasil *picking* otomatis dengan menggunakan model CNN yang menduga *noise* ataupun gelombang permukaan sebagai *event* pada hari 3 jam 00.00 hingga 01.00.

Fenomena ketiga yang dijumpai pada hasil *picking* otomatis ini adalah *picking* otomatis yang tidak menemukan adanya *event*, meskipun di *picking* manual mendeteksi *event*. Hal ini mayoritas disebabkan oleh amplitudo puncak ketiga komponen saling identik sehingga model tidak mendeteksi sebagai gelombang P atau S. Perlu diingat bahwa model ini menentukan gelombang P ataupun S berdasarkan perbandingan amplitudo maksimum relatif terhadap komponen lainnya. Oleh karena itu, ketika dijumpai seismogram seperti pada Gambar 4.6 yang amplitudonya relatif mirip, model tidak dapat mendefinisikan bahwa itu adalah *event*. Hal ini dapat diperbaiki dengan memperkecil *window* pada proses prediksi, meskipun kurang praktis jika diterapkan pada banyak data.

Secara kualitatif, terdapat banyak perbedaan antara *picking* manual dan otomatis. Akan tetapi, jika dilihat dari sudut pandang yang lain, *picking* otomatis ini memiliki lebih banyak kegunaan dibandingkan dengan *picking* manual yang hanya fokus pada *event* mikroseismik. *Picking* otomatis ini dapat berguna juga untuk perhitungan hiposenter yang hanya memerlukan tp maupun ts. Selain itu, program ini dapat digunakan untuk tujuan analisis kebencanaan, ketika hanya lonjakan amplitudo yang diperlukan untuk dianalisis.

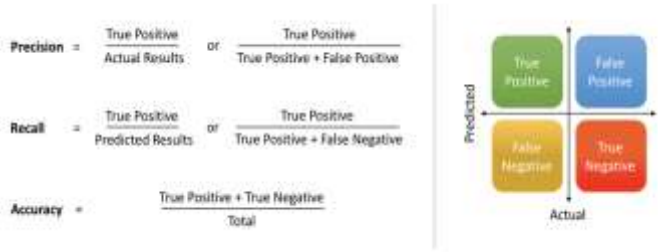



**Gambar 4. 7.** Contoh hasil *picking* otomatis dengan menggunakan model CNN yang tidak membaca anomali yang terlihat *event* sebagai *event* pada hari kelima jam 12.00-13.00.

### 4.2.4. Analisis Kuantitatif *Picking* Otomatis pada Data Lapangan "X"

Dalam analisis kuantitatif, diperlukan beberapa parameter yang seringkali digunakan dalam dunia *artificial intelligence* (AI). Parameter pertama adalah presisi, atau *precision*. *Precision* dihitung berdasarkan persentasi hasil *picking* otomatis yang relevan dengan hasil sebenarnya. Dengan asumsi *picking* manual benar, parameter ini menjelaskan berapa persen hasil *picking* otomatis

yang menyatakan *event* yang memang sebenarnya *event*. Parameter kedua, *recall*, didefinisikan sebagai persentase hasil relevan (*event* sebenarnya) yang diklasifikasikan secara benar oleh model ini. *Recall* menjawab pertanyaan yang berupa dari semua *event*, berapa persen yang benar diklasifikasikan sebagai *event* oleh model CNN ini. Parameter ketiga adalah akurasi. Akurasi dihitung dengan cara membagi semua hasil yang benar, baik terbaca *event* ketika benar terjadi *event* maupun tidak terbaca *event* ketika tidak ada *event*, dengan seluruh ruang sampel atau jumlah data yang dibaca. Ketiga parameter ini sangat signifikan dalam memvalidasi analisis kualitatif yang telah dilakukan di subbab sebelumnya. Selain itu, parameter-parameter tersebut juga dapat menunjukkan dengan pasti akar permasalahan dari model ini dan menemukan solusi perbaikan di masa mendatang.




**Gambar 4.8.** Ilustrasi cara perhitungan parameter AI pada model CNN. (Goutte & Gaussier, 2005)

Tahap pertama sebelum menghitung parameter AI di atas adalah membuat *confusion matrix*. *Confusion matrix* adalah matriks yang menunjukkan keempat komponen seperti pada Gambar 4.7, yaitu *true positive* (*event* yang memang dibaca sebagai *event* oleh CNN), *false positive* (bukan *event* yang dibaca sebagai *event* oleh CNN), *false negative* (*event* yang tidak dibaca sebagai *event* oleh CNN), dan *true negative* (bukan *event* yang juga tidak dibaca sebagai *event* oleh CNN). Dari *confusion* matriks ini, dapat dihitung nilai *precision*, *recall*, maupun *accuracy*-nya. Gambar 4.8 menunjukkan hasil *confusion matrix* yang didapat dari pembacaan data 15 hari oleh model CNN.

| | *Event* pada *picking* manual | *Bukan event* pada *picking* manual |
|---|---|---|
| *Event* pada *picking* CNN | 8 | 2 |
| *Bukan event* pada *picking* CNN | 23 | 248 |

**Gambar 4.9.** *Confusion matrix* model CNN.

*Confusion matrix* menunjukkan bahwa model dapat secara tepat mengklasifikasikan *false negative*, ditunjukkan oleh banyaknya frekuensi. Juga terdapat 8 *true positive*, yang menandakan model sudah cukup baik dalam mendeteksi *event*. Akan tetapi, terdapat cukup banyak kasus ketika model tidak membaca *event* pada data, padahal pada kenyataannya terdapat *event*, yaitu sebanyak 23 kali. Jika dilakukan perhitungan lebih lanjut:

$$Precision = 8/(8+2) = 0.8$$

$$Recall = 8/(23+8) = 0.26$$

$$Accuracy = (248+8)/(248+8+23+2) = 0.91$$

$$F1 = 2 \text{ x precision x recall } /(\text{precision + recall})$$

$$= 2(0.8 * 0.26)/1.06 = 0.39$$

Dapat dilihat bahwa model ini memiliki nilai presisi yang tinggi yaitu 0.8 dan akurasi yang baik sebesar 0.91, namun nilai *recall* cukup rendah, yaitu 0.26. Dilakukan juga satu perhitungan tambahan yaitu skor F1. Skor F1 menentukan akurasi tidak hanya berdasarkan *precision* atau *recall*, namun mempertimbangkan keduanya dengan mengambil rata-rata harmonik antara keduanya. Nilai F1 juga cukup rendah sebesar 0.39.

Dari hasil parameter berikut, dapat dijawab apa yang terjadi pada proses *training* dan prediksi. Berdasarkan interpretasi, nilai *recall* yang rendah menandakan banyak *event* yang dibaca sebagai *event* oleh *picking* manual, namun oleh CNN tidak dibaca sebagai *event.* Hal ini mengacu kembali kepada Gambar 4.6, bahwa banyak data yang memiliki karakteristik seperti gambar tersebut, yaitu sekilas terlihat seperti *event*, padahal secara ketentuan perbedaan amplitudo, data tersebut belum memenuhi. Model CNN ini secara objektif menentukan bahwa hal tersebut bukan *event*, karena yang disebut sebagai gelombang P di *picking* manual ternyata tidak memiliki amplitudo yang lebih besar di komponen vertikal dibandingkan komponen lainnya. Begitu juga dengan yang disebut sebagai gelombang S di *picking* manual ternyata tidak memiliki amplitudo yang lebih besar di komponen horizontal dibandingkan komponen vertikal. Seluruh perhitungan di atas menggunakan asumsi bahwa *picking* manual adalah hasil yang benar. Akan tetapi, tidak menutup kemungkinan bahwa hasil *picking* manual memiliki tingkat kesalahan tertentu. *Picking* manual tetap dilakukan oleh manusia, sehingga terdapat subjektivitas yang memengaruhi proses *picking*. Akan tetapi, dengan menggunakan model CNN, subjektivitas dapat dikurangi dengan memberlakukan kaidah *picking* yang benar pada data seismogram, khususnya data mikroseismik di Lapangan "X". Dengan hasil yang ada, tidak dapat dikatakan bahwa model CNN ini dapat sepenuhnya menggantikan manusia. Tetapi, model AI ini dapat melengkapi peran geofisikawan sebagai kontrol kualitas *picking* apakah hasil *picking* yang dilakukan sudah cukup objektif, tidak hanya mengambil *picking* tertentu karena ada lonjakan amplitudo sembarang.

Secara keseluruhan, model ini dapat memberlakukan objektivitas pada proses *picking* gelombang P dan S pada data seismogram.

# BAB V
# SIMPULAN DAN SARAN

## 5.1. Simpulan

Berdasarkan penelitian yang telah dilakukan, diperoleh simpulan sebagai berikut:

1. Program *picking* otomatis gelombang seismik pasif telah berhasil dibuat dan diperoleh tp dan ts dengan lebih objektif memperhatikan kaidah picking gelombang P dan S.
2. Nilai *precision* dan *recall* untuk evaluasi model secara kuantitatif adalah masing-masing 0.8 dan 0.26.
3. Hal yang memengaruhi nilai parameter kuantitatif evaluasi CNN: nilai ambang probabilitas, kualitas data *training* dan data *testing*.

## 5.2. Saran

Berdasarkan penelitian yang telah dilakukan, penulis menyarankan sebagai berikut:

1. Digunakan data MEQ sebagai data training sehingga tingkat kesamaan antara data training dan data evaluasi lebih baik
2. Digunakan lebih banyak parameter untuk menentukan fase gelombang P dan S.
3. Digunakan data yang memiliki kualitas instrumen maupun filter yang lebih baik sehingga memicu kolaborasi dengan ahli instrumen geofisika untuk kemajuan penelitian mikroseismik di Indonesia.

*Halaman ini sengaja dikosongkan*

# BAB VI
# RENCANA KEBERLANJUTAN PENELITIAN

Program ini akan dirilis ke *repository* yang dapat diakses oleh siapa pun, yaitu Github. Program ini dirilis dengan nama PyMEQCNN dan dapat diakses di link berikut: https://github.com/christophersalim/PyMEQCNN. Harapannya, program ini dapat dimodifikasi dan diperbaiki sehingga dapat diperoleh *picker* otomatis yang memiliki akurasi lebih tinggi. Untuk memfasilitasi kemampuan dasar mahasiswa Departemen Teknik Geofisika dalam memodifikasi *script* ini, akan ada pelatihan dasar Python sebagai bahasa utama pemrograman.

*Halaman ini sengaja dikosongkan*

# DAFTAR PUSTAKA


Aggarwal, C. C. (2018). *Neural Networks and Deep Learning.* Springer.

Allen, R. (1982). Automatic Phase Pickers: Their Present Use and Future Prospects. *Bulletin of the Seismological Society of America, 72*, pp. S225-S242.

Baillard, C., Crawford, W. C., Ballu, V., Hibert, C., & Mangeney, A. (2014, February). An Automatic Kurtosis-Based P- and S-Phase Picker Designed for Local Seismic Networks. *Bulletin of the Seismological Society of America, 104*.

Chopra, R., England, A., & Alaudeen, M. N. (2019). *Data Science with Python: Combine Python with Machine Learning Principles to Discover Hidden Patterns in Raw Data.* Packt Publishing.

Colak, Ö. H., Destici, T. C., Özen, Ş., Arman, H., & Çerezci, O. (2009). DETECTION OF P- AND S-WAVE ARRIVAL TIMES USING THE DISCRETE WAVELET TRANSFORM IN REAL SEISMOGRAMS. *The Arabian Journal for Science and Engineering*.

Diehl, T., & Kissling, E. (2007). *Users Guide for Consistent Phase Picking at Local to Regional Scales.* Zurich: Swiss Federal Institute of Technology Zurich (ETH).

Fauzi. (2010). *Analisis Data Seismogram untuk Menentukan Parameter Magnitude Gempa Bumi (Studi Kasus Gempabumi Padang 30 September 2009).* Jakarta: Universitas Islam Negeri Syarif Hidayatullah.

Goutte, C., & Gaussier, E. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. *Advances in Information Retrieval.* ECIR 2005.

Hamzah, M., Makhrani, & Hasni, N. (n.d.). *PENENTUAN HIPOSENTER GEMPA MIKRO MENGGUNAKAN METODE SINGLE EVENT DETERMINATION DAN JOINT HYPOCENTER DETERMINATION PADA LAPANGAN PANAS BUMI X.* Retrieved from Repository Universitas Hassanudin.

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network - A Deep Learning Approach. *International Conference on Computational Intelligence and Data Science (ICCIDS 2018)*, (pp. 679–688).

Isroi, A. R., Singarimbun, A., & Herdiansyah, T. (2015). Relokasi Hiposenter Gempa Mikro Menggunakan Metode SED (Single Event Determination) di Area Geothermal Kamojang. *Simposium Nasional Inovasi dan Pembelajaran Sains.* Bandung.

Kamah, Y. (2016). *Laporan Periodik Monitoring Gempa Mikro (MEQ) PT. PERTAMINA (PERSERO) Area Geothermal Kamojang.* Bandung: Unpublished.

Kingma, D., & Ba, L. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 645-657.

Mehrotra, K., Mohan, C. K., & Ranka, S. (1997). *Elements of Artificial Neural Networks.* MIT Press.

Meier, M.-A., Ross, Z. E., Ramachandran, A., Balakrishna, A., Nair, S., Kundzicz, P., . . . Yue, Y. (2018). Reliable Real-Time Seismic Signal/Noise Discrimination With Machine Learning. *Journal of Geophysical Research: Solid Earth*.

Mousavi, S. M., Zhu, W., Sheng, Y., & Beroza, G. C. (2019). *CRED: A Deep Residual Network of Convolutional and Recurrent Units for Earthquake Signal Detection.* Nature.

Nainggolan, P. R., & Santosa, B. J. (2013). Relokasi Hiposenter untuk Data Gempa Bumi di Wilayah Sumatera Barat dan Sekitarnya dengan Menggunakan Hypo71 (2009-10-01 – 2010-12-21). *JURNAL SAINS DAN SENI POMITS*.

Nielsen, M. (n.d.). *Neural Networks and Deep Learning*. Retrieved from http://neuralnetworksanddeeplearning.com

Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances*.

Ross, Z. E., Meier, M.-A., Hauksson, E., & Heaton, T. H. (2018). Generalized Seismic Phase Detection with Deep Learning. *Bulletin of the Seismological Society of America*, pp. 2894-2901.

Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C., & Johnson, P. A. (2017). Machine Learning Predicts Laboratory Earthquakes. *arXiv:1702.05774v1 [physics.geo-ph]*.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., . . . Adams, R. P. (2015). Scalable Bayesian optimization using deep neural networks. *Proceeding ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning*, (pp. 2171-2180 ).

Stimac, J., Goff, F., & Goff, C. J. (2015). Chapter 46 - Intrusion-Related Geothermal Systems. In H. S. (Ed.), *The Encyclopedia of Volcanoes (Second Edition)* (pp. 799-822). Elsevier.

Toth, A., & Bobok, E. (2015). *Flow and Heat Transfer in Geothermal Systems.* Elsevier.

Trnkoczy, A. (2012). Understanding and parameter setting of STA/LTA trigger algorithm. In P. (. Bormann, *New Manual of Seismological*

*Observatory Practice 2 (NMSOP-2)* (pp. 1-20). Potsdam : Deutsches GeoForschungsZentrum GFZ.

Zhang, X., Zhang, J., Yuan, C., Liu, S., Chen, Z., & Li, W. (2018). Locating earthquakes with a network of seismic stations via a deep learning method. *arXiv:1808.09603v1 [physics.geo-ph]*.

Zhu, W., & Beroza, G. C. (2019). PhaseNet: a deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International*, 261–273.

# BIOGRAFI PENULIS

Christopher Salim lahir di Surabaya, 20 Juni 1998. Penulis menempuh pendidikan formal dimulai di SD YPPI II Surabaya (2004-2010), SMP YPPI I Surabaya (2010-2013), SMA Katolik Santa Agnes Surabaya (2013-2016), dan pendidikan sarjana di Departemen Teknik Geofisika, Institut Teknologi Sepuluh Nopember Surabaya. Selama menjadi mahasiswa, penulis menjalani berbagai kegiatan, baik akademik maupun non-akademik. Penulis pernah menjadi asisten mata kuliah Geofisika Matematika, Komputasi Geofisika, dan Fisika Batuan. Penulis memiliki beberapa kesempatan untuk belajar di luar kampus, seperti mengikuti Sakura Science Program yang didanai oleh Japan Science and Technology Agency (JST) di Earthquake Research Institute, The University of Tokyo dan kerja praktik di PT. Pertamina EP Asset 5. Di bidang non-akademik, penulis pernah menjadi staf Department of Academia, Society of Petroleum Engineers ITS Student Chapter (SPE ITS SC) pada periode 2017/2018 dan menjadi Koordinator Himpunan Mahasiswa Geofisika Wilayah IV pada periode 2018/2019. Penulis sangat berkenan apabila ada saran, kritik, maupun ajakan diskusi tentang komputasi, pemrograman, atau yang lain. Berikut kontak email penulis christopher.salim@yahoo.com jika ada pertanyaan maupun diskusi yang ingin disampaikan.

## LAMPIRAN 1

### *SCRIPT* UNTUK MELAKUKAN *TRAINING* MODEL CNN

```
import keras
import tensorflow as tf
import numpy as np
import h5py
from matplotlib import pyplot as plt
import pickle
datapath = 'training_data_raw.hdf5'


with h5py.File(datapath, 'r') as f:

print(list(f.keys()))

waveform_data = f['X']

label = f['Y']

print(waveform_data.shape)

print(label[0:10])

waveform_P = waveform_data[0,:,:]

waveform_S = waveform_data[1,:,:]

waveform_noise = waveform_data[2,:,:]

print(waveform_P.shape)

for k in range(0,3):
    plt.figure(k)
    plt.plot(waveform_P[:,k])
    plt.title('Plot Gelombang P komponen ke- ' + s
tr(k+1), color='black')
    plt.xlabel('Time (s)', color='black')
    plt.ylabel('Normalized velocity', color='black
')
for k in range(0,3):
    plt.figure(k+3)
    plt.plot(waveform_S[:,k])
```

```
    plt.title('Plot Gelombang S komponen ke- ' + s
tr(k+1), color='black')
    plt.xlabel('Time (s)', color='black')
    plt.ylabel('Normalized velocity', color='black
')
for k in range(0,3):
    plt.figure(k+6)
    plt.plot(waveform_noise[:,k])
    plt.title('Plot noise komponen ke- ' + str(k+1
), color='black')
    plt.xlabel('Time (s)', color='black')
    plt.ylabel('Normalized velocity', color='black
')
from keras.utils.io_utils import HDF5Matrix
train_start = 0
n_training_examples = int(0.8 * 4773750)
test_start = n_training_examples + 1
n_test_examples = int(0.2 * 4773750)
x_train = HDF5Matrix(datapath, 'X', train_start, t
rain_start+n_training_examples)
y_train = HDF5Matrix(datapath, 'Y', train_start, t
rain_start+n_training_examples)
x_test = HDF5Matrix(datapath, 'X', test_start, tes
t_start+n_test_examples)
y_test = HDF5Matrix(datapath, 'Y', test_start, tes
t_start+n_test_examples)
train_shape = y_train.shape[0]
test_shape = y_test.shape[0]
from keras.utils import to_categorical


from keras.models import Sequential
from keras.layers import Dense, Reshape, Conv1D, M
axPooling1D, GlobalAveragePooling1D, GaussianNoise
, Dropout, Dense
num_classes = 3
model_m = Sequential()
model_m.add(Conv1D(filters=256, kernel_size=100, a
ctivation='relu', input_shape=(400, 3)))
model_m.add(Conv1D(filters=256, kernel_size=10, ac
tivation='relu'))
```

```
model_m.add(MaxPooling1D(3))
model_m.add(Conv1D(filters=64, kernel_size=10, act
ivation='relu'))
model_m.add(Conv1D(filters=64, kernel_size=10, act
ivation='relu'))
model_m.add(MaxPooling1D(3))
model_m.add(GlobalAveragePooling1D())
model_m.add(Dropout(0.1))
model_m.add(Dense(64))
model_m.add(Dense(3, activation='softmax'))
# Specify the training configuration (optimizer, l
oss, metrics)
model_m.compile(optimizer='adam',  # Optimizer
              # Loss function to minimize
              loss='categorical_crossentropy',
              # List of metrics to monitor
              metrics=['categorical_accuracy'])
print(model_m.summary())
def write_log(callback, names, logs, batch_no):
    for name, value in zip(names, logs):
        summary = tf.Summary()
        summary_value = summary.value.add()
        summary_value.simple_value = value
        summary_value.tag = name
        callback.writer.add_summary(summary, batch
_no)
        callback.writer.flush()
```

```
from keras.callbacks import TensorBoard
from keras.models import Model
epoch = 3
batch_num=4000
batchsize_train = train_shape // batch_num
batchsize_test = test_shape // batch_num
log_path = './logs'
callback = TensorBoard(log_path)
callback.set_model(model_m)
train_names = ['train_loss', 'train_accuracy']
val_names = ['val_loss', 'val_accuracy']
```

```
from keras.utils import to_categorical
average_train_accuracy = np.zeros((epoch, 1))
average_train_loss = np.zeros((epoch, 1))
average_test_accuracy = np.zeros((epoch, 1))
average_test_loss = np.zeros((epoch, 1))
train_loss = 0
train_accuracy = 0
test_loss = 0
test_accuracy = 0
for i in range(epoch):
    print('epoch = ' + str(i) + '---------------')
    for j in range(batch_num):
        batch_x_train = x_train[j * batchsize_trai
n: (j + 1) * batchsize_train]
        batch_y_train = to_categorical(y_train[j *
batchsize_train:(j + 1) * batchsize_train])
        batch_x_test = x_test[j * batchsize_test:
(j + 1) * batchsize_test]
        batch_y_test = to_categorical(y_test[j * b
atchsize_test:(j + 1) * batchsize_test])
        print(batch_y_train.shape)
        print(batch_y_test.shape)
        c1 = model_m.train_on_batch(batch_x_train,
batch_y_train)
        train_loss += c1[0]
        train_accuracy += c1[1]
        c2 = model_m.test_on_batch(batch_x_test, b
atch_y_test)
        test_loss += c2[0]
        test_accuracy += c2[1]
        print('train', c1)
        print('val', c2)
        write_log(callback, train_names, c1, i)
        write_log(callback, val_names, c2, i)
    average_train_accuracy[i] = train_accuracy/bat
chsize_train
    average_train_loss[i] = train_loss/batchsize_t
rain
    average_test_accuracy[i] = test_accuracy/batch
size_test
```

```
    average_test_loss[i] = test_loss/batchsize_tes
t
    print('average train accuracy = ' + str(averag
e_train_accuracy))
    print('average train loss = ' + str(average_tr
ain_loss))
    print('average test accuracy = ' + str(average
_test_accuracy))
    print('average test loss = ' + str(average_tes
t_loss))
print("end of training")
model_m.save_weights('model_weights.h5')
model_m.save('model.h5')
with open('model_architecture.json', 'w') as f:
    f.write(model_m.to_json())
```

# LAMPIRAN 2

## *SCRIPT* UNTUK MELAKUKAN PREDIKSI WAKTU TIBA DI DATA LAPANGAN "X"

```
#
/bin/en
v
python
          # Modified from Ross et al (2018)
          # PyMEQCNN (Christopher Salim, 2019)
          # Additions from previous version: disabled the -V
          input to simplify the usage, added the csv save
          # feature, also enabled the user to save the plot
          obtained from the running result immediately.


          import string
          import time
          import argparse as ap
          import sys
          import os


          import numpy as np
          import obspy.core as oc
          import keras
          from keras.models import Sequential
          from keras.layers import Dense, Dropout,
          Activation, Flatten
          from keras.layers import Conv1D, MaxPooling1D
          from keras import losses
```

```
from keras.models import model_from_json
import tensorflow as tf
import matplotlib as mpl
import pylab as plt
import pandas as pd
mpl.rcParams['pdf.fonttype'] = 42


######################
# Hyperparameters
min_proba = 0.95 # Minimum softmax probability for
phase detection
freq_min = 6.0
freq_max = 12.0
filter_data = False
decimate_data = True # If false, assumes data is
already 100 Hz samprate
n_shift = 400 # Number of samples to shift the
sliding window at a time
n_gpu = 0 # Number of GPUs to use (if any)
######################
batch_size = 300


half_dur = 2.0
only_dt = 0.01
n_win = int(half_dur/only_dt)
n_feat = 2*n_win


#-------------------------------------------------
-----------
```

```
def sliding_window(data, size, stepsize=1,
padded=False, axis=-1, copy=True):
    """
    Calculate a sliding window over a signal
    Parameters
    ----------
    data : numpy array
        The array to be slided over.
    size : int
        The sliding window size
    stepsize : int
        The sliding window stepsize. Defaults to 1.
    axis : int
        The axis to slide over. Defaults to the
last axis.
    copy : bool
        Return strided array as copy to avoid
sideffects when manipulating the
        output array.
    Returns
    -------
    data : numpy array
        A matrix where row in last dimension
consists of one instance
        of the sliding window.
    Notes
    -----
    - Be wary of setting `copy` to `False` as
undesired sideffects with the
      output values may occurr.
```

```
    Examples
    --------
    >>> a = numpy.array([1, 2, 3, 4, 5])
    >>> sliding_window(a, size=3)
    array([[1, 2, 3],
           [2, 3, 4],
           [3, 4, 5]])
    >>> sliding_window(a, size=3, stepsize=2)
    array([[1, 2, 3],
           [3, 4, 5]])
    See Also
    --------
    pieces : Calculate number of pieces available
by sliding
    """
    if axis >= data.ndim:
        raise ValueError(
            "Axis value out of range"
        )


    if stepsize < 1:
        raise ValueError(
            "Stepsize may not be zero or negative"
        )


    if size > data.shape[axis]:
        raise ValueError(
            "Sliding window size may not exceed
size of selected axis"
        )
```

```
    shape = list(data.shape)
    shape[axis] = np.floor(data.shape[axis] /
stepsize - size / stepsize + 1).astype(int)
    shape.append(size)


    strides = list(data.strides)
    strides[axis] *= stepsize
    strides.append(data.strides[axis])


    strided = np.lib.stride_tricks.as_strided(
        data, shape=shape, strides=strides
    )


    if copy:
        return strided.copy()
    else:
        return strided


if __name__ == "__main__":
    parser = ap.ArgumentParser(
        prog='gpd_predict.py',
        description='Automatic picking of seismic
waves using'
                    'Generalized Phase Detection')
    parser.add_argument(
        '-I',
```

```
    type=str,
    default=None,
    help='Input file')
parser.add_argument(
    '-O',
    type=str,
    default=None,
    help='Output file')
parser.add_argument(
    '-P',
    default=True,
    action='store_false',
    help='Suppress plotting output')
parser.add_argument(
    '-V',
    default=False,
    action='store_true',
    help='verbose')
args = parser.parse_args()


plot = args.P


# Reading in input file
fdir = []
evid = []
staid = []
with open(args.I) as f:
    for line in f:
        tmp = line.split()
        fdir.append([tmp[0], tmp[1], tmp[2]])
```

```
    nsta = len(fdir)
    print(nsta)
    # load json and create model
    json_file = open('model_architecture.json',
'r')
    loaded_model_json = json_file.read()
    json_file.close()
    model = model_from_json(loaded_model_json,
custom_objects={'tf':tf})


    # load weights into new model
    model.load_weights("model_weights.h5")
    print("Loaded model from disk")


    if n_gpu > 1:
        from keras.utils import multi_gpu_model
        model = multi_gpu_model(model, gpus=n_gpu)


    ofile = open(args.O, 'w')
    station=[]
    pick_p_fix = []
    pick_s_fix = []
    for i in range(nsta):
        fname = fdir[i][0].split("/")
        if not os.path.isfile(fdir[i][0]):
            print("%s doesn't exist, skipping" %
fdir[i][0])
            continue
        if not os.path.isfile(fdir[i][1]):
```

```
            print("%s doesn't exist, skipping" %
fdir[i][1])
            continue
        if not os.path.isfile(fdir[i][2]):
            print("%s doesn't exist, skipping" %
fdir[i][2])
            continue
        st = oc.Stream()
        st += oc.read(fdir[i][0])
        st += oc.read(fdir[i][1])
        st += oc.read(fdir[i][2])
        st = st.normalize()
        latest_start = np.max([x.stats.starttime
for x in st])
        earliest_stop = np.min([x.stats.endtime for
x in st])
        st.trim(latest_start, earliest_stop)


        st.detrend(type='linear')
        if filter_data:
            st.filter(type='bandpass',
freqmin=freq_min, freqmax=freq_max, corners=2,
zerophase=True)
        if decimate_data:
            st.interpolate(100.0)
        chan = st[0].stats.channel
        sr = st[0].stats.sampling_rate


        dt = st[0].stats.delta
        net = st[0].stats.network
```

```
        sta = st[0].stats.station
        if sta == 'TES':
            sta = str(fname[0][16:20])


        print("Reshaping data matrix for sliding
window")
        tt = (np.arange(0, st[0].data.size,
n_shift) + n_win) * dt
        tt_i = np.arange(0, st[0].data.size,
n_shift) + n_feat
        #tr_win = np.zeros((tt.size, n_feat, 3))
        sliding_N = sliding_window(st[0].data,
n_feat, stepsize=n_shift)
        sliding_E = sliding_window(st[1].data,
n_feat, stepsize=n_shift)
        sliding_Z = sliding_window(st[2].data,
n_feat, stepsize=n_shift)
        tr_win = np.zeros((sliding_N.shape[0],
n_feat, 3))
        tr_win[:,:,0] = sliding_N
        tr_win[:,:,1] = sliding_E
        tr_win[:,:,2] = sliding_Z
        tr_win = tr_win / np.max(np.abs(tr_win),
axis=(1,2))[:,None,None]
        tt = tt[:tr_win.shape[0]]
        tt_i = tt_i[:tr_win.shape[0]]


        ts = model.predict(tr_win, verbose=True,
batch_size=batch_size)
```

```
            prob_S = ts[:,1]
            prob_P = ts[:,0]
            prob_N = ts[:,2]

            station.append(sta)
            pick_p = np.where(prob_P>=0.95)
            pick_p_new = np.array(pick_p)
            np.savetxt('pick_P ' + str(sta),
pick_p_new*n_shift/100)
            pick_p_new = pick_p_new[0]
            pick_p_fix.append(pick_p_new*n_shift/100)

            pick_s = np.where(prob_S>=0.95)
            pick_s_new = np.array(pick_s)
            np.savetxt('pick_S ' + str(sta),
pick_s_new*n_shift/100)
            pick_s_new = pick_s_new[0]
            pick_s_fix.append(pick_s_new*n_shift/100)


            p_picks = []
            s_picks = []
            for picks in pick_p_new:
                stamp_pick = st[0].stats.starttime +
tt[picks]
                p_picks.append(stamp_pick)
                ofile.write("%s %s P %s\n" % (net, sta,
stamp_pick.isoformat()))
                np.savetxt('p_picks' + str(i), p_picks)
            for picks in pick_s_new:
```

```
            stamp_pick = st[0].stats.starttime +
tt[picks]
            s_picks.append(stamp_pick)
            ofile.write("%s %s S %s\n" % (net, sta,
stamp_pick.isoformat()))
        fig = plt.figure(figsize=(8, 12))
        ax = []
        ax.append(fig.add_subplot(4,1,1))

ax.append(fig.add_subplot(4,1,2,sharex=ax[0],sharey
=ax[0]))

ax.append(fig.add_subplot(4,1,3,sharex=ax[0],sharey
=ax[0]))

ax.append(fig.add_subplot(4,1,4,sharex=ax[0]))
        for j in range(3):

ax[j].plot(np.arange(st[j].data.size)*dt,
st[j].data, c='k', \
                        lw=0.5)
            ax[j].set_xlabel('Time (s)')
            ax[j].set_ylabel('Normalized value')
        ax[0].set_title('Picking in N-S component')
        ax[1].set_title('Picking in E-W component')
        ax[2].set_title('Picking in U-D component')
        ax[3].plot(tt, ts[:,0], c='r', lw=0.5)
        ax[3].plot(tt, ts[:,1], c='b', lw=0.5)
        ax[3].axhline(min_proba,linestyle='--')
        ax[3].set_title('Probability plot')
        for p_pick in p_picks:
            for k in range(3):
```

```
                for xc in pick_p_new:
                    ax[k].axvline(x=xc*4, c='r',
lw=0.5)
        for s_pick in s_picks:
            for k in range(3):
                for xc in pick_s_new:
                    ax[k].axvline(x=xc*4, c='b',
lw=0.5)
        plt.tight_layout()
        fig.savefig('Probability_plot_station' +
str(i+1))
    df = pd.DataFrame({'Station': station, 'tp':
pick_p_fix, 'ts': pick_s_fix})
    df.to_csv(r'./result_dataframe.csv', index =
None, header=True)
    ofile.close()
```