Model Pembelajaran dalam Jaringan Saraf Konvolusional untuk Klasifikasi Teks (Abstrak Penelitian Ilmu Komputer)

Adi Putra Sugiarto[1)], Sriyanto[2)]
Institut Informatika dan Bisnis Darmajaya
Bandar Lampung, Lampung, (0721) 787214
e-mail: adi.tjong@outlook.com

Abstrak

Jumlah informasi teks yang tersedia dari dokumen baru yang tersedia secara online terus meningkat,, salah satu dokumen yang tersedia adalah dalam bidang penelitian computer science, yang merupakan parential category (domain) dan memiliki sebelas sub parential category (sub domain) seperti artificial intelligence, etc. dengan memiliki banyak sub domain menjadikannya sulit untuk secara efektif memberikan kategori yang benar untuk dokumen baru, dengan tersedianya banyak referensi dokumen yang telah memiliki label maka diperlukan metode yang dapat membangun sebuah model yang mampu melakukan proses klasifikasi pada dokumen baru dengan tingkat akurasi yang baik sehingga dapat digunakan sebagai acuan dalam pemberian label kategori sub domain penelitian bidang computer science.

Pada penelitian ini akan menggunakan pendekatan baru dalam memahami bahasa alami (natural language) di mana mempertimbangkan input teks sebagai gambar dan menerapkan 2D convolutional neural network untuk mempelajari semantik lokal dan global kalimat dari variasi pola visual kata-kata, embedding, dan 2D max pooling, tujuan dari penelitian ini adalah untuk melakukan klasifikasi teks berbasis alfabet non-latin dengan 11 dataset klasifikasi teks yang merupakan subdomain dari bidang penelitian computer science.

Model deep learning yang dihasilkan mampu mengklasifikasikan abstrak penelitian dalam bidang computer science, ke dalam beberapa subdomain penelitian menggunakan model deep learning convolutional neural network dengan tingkat akurasi 0.9522 dan loss 0.1493, validasi dilakukan dengan melakukan split pada dataset, perbandingan 80% untuk data training dan 20% untuk data testing.

Kata Kunci : *Deep Learning, Convolutional Neural Network, Classification, Text Mining, Natural Language Processing.*

1. PENDAHULUAN

Dalam beberapa tahun terakhir, *big data* telah berkembang pesat dan jumlah informasi teks yang tersedia dari dokumen baru yang tersedia secara *online* terus meningkat, mengelola untuk mengklasifikasikannya dengan benar menjadi lebih sulit karena kemampuan untuk secara efektif mengambil kategori yang benar untuk dokumen baru sangat bergantung pada jumlah dokumen yang sudah tersedia untuk dijadikan referensi, salah satu dokumen yang tersedia adalah dalam bidang penelitian *computer science,* banyak penelitian yang telah dilakukan dan hasil dari penelitian tersebut terus dilaporkan dan diterbitkan melalui jurnal-jurnal penelitian seperti yang tersedia pada situs web *ScienceDirect* yang merupakan penyedia akses berlangganan ke pangkalan data berisikan penelitian ilmiah.

Computer science merupakan *parential category* (domain) yang tersedia pada situs web *ScienceDirect* dan memiliki *sub parential category* (sub domain) seperti *artificial intelligence, computational theory and mathematic, computer graphics and computer-aided design, computer networks and communications, etc.* dengan memiliki banyak sub domain pada bidang penelitian *computer science*, menjadikannya sulit untuk secara efektif memberikan kategori yang benar untuk dokumen baru, dengan tersedianya banyak referensi dokumen yang telah memiliki label maka diperlukan metode yang dapat membangun sebuah model yang mampu melakukan proses klasifikasi pada dokumen baru dengan tingkat akurasi yang baik sehingga

dapat digunakan sebagai acuan dalam pemberian label kategori sub domain penelitian bidang *computer science.*

Berdasarkan latar belakang masalah di atas, banyak peneliti mengusulkan sejumlah metode dan dilaporkan dalam berbagai literatur, tetapi pada penelitian ini akan menggunakan pendekatan baru dalam memahami bahasa alami *(natural language)* di mana mempertimbangkan *input* teks sebagai gambar dan menerapkan 2D *convolutional neural network* untuk mempelajari semantik lokal dan global kalimat dari variasi pola visual kata-kata, *embedding,* dan 2D *max pooling,* tujuan dari penelitian ini adalah untuk melakukan klasifikasi teks berbasis alfabet non-latin dengan 11 *dataset* klasifikasi teks yang merupakan *subdomain* dari bidang penelitian *computer science.*

2. METODE PENELITIAN

Klasifikasi merupakan proses menetapkan kategori atau beberapa kategori di antara yang sudah ditentukan sebelumnya untuk setiap item data. Pada proses klasifikasi terdapat dua proses penting yang dilakukan, yaitu *learning (training)* dan *testing.* Pada proses *learning* model dilatih dengan melakukan proses pembelajaran menggunakan data yang telah memiliki kategori/label, dan proses yang kedua adalah proses *testing* untuk menguji/memvalidasi model dengan menggunakan data *testing.*

Sebagai tugas awal untuk klasifikasi, daftar kategori sudah ditentukan sebelumnya sebagai sistem klasifikasi dan item data dialokasikan ke setiap kategori sebagai data sampel, pada penelitian ini *dataset* yang digunakan adalah abstrak penelitian dari dokumen penelitian pada *website sciencedirect,* domain penelitian yang ditentukan adalah *computer science,* dengan masing-masing *subdomain* memiliki 100 item data. *Domain* dan *subdomain computer science* yang digunakan dalam penelitian adalah *Computer Science* sebagai *parrent category/domain*, dan *sub parrent category/sub domain* adalah *Artificial Intelligence, Computational Theory and Mathematics, Computer Graphics and Computer-Aided Design, Computer Networks and Communications, Computer Science, Computer Vision and Pattern Recognition, Hardware and Architectire, Human-Computer Interaction, Informations Systems, Signal Processing, Software*.

2.1. Tahapan Penelitian

Pada penelitian ini akan dibagi menjadi beberapa tahapan, tahapan penelitian dimulai dengan mengumpulkan data berupa abstrak penelitian, kemudian data akan melalui proses pengolahan *(data preprocessing)* dan menghasilkan *output* yang akan digunakan pada tahap selanjutnya yaitu tahapan *training process* yang merupakan tahap proses pembelajaran, dari proses tersebut akan menghasilkan model, kemudian model akan digunakan pada tahapan selanjutnya yaitu proses evaluasi dan validasi dari model yang dihasilkan dengan memberikan *input* berupa data abstrak yang tidak memiliki *subdomain,* dari hasil evaluasi atau prediksi akan menghasilkan *subdomain* atau label hasil prediksi dari *input* yang diberikan dengan tingkat probabilitas.

2.1.1 Tahap Data *Collection*

Pada tahapan *data collection*, data berupa teks abstrak penelitian dikumpulkan dan digunakan untuk proses selanjutnya, data yang dibutuhkan diambil dari *website sciencedirect,* pengambilan data dilakukan secara manual dengan menggunakan akun yang telah terdaftar karena dibutuhkan akses terhadap data, proses selanjutnya adalah melakukan pencarian berdasarkan *keywords* dengan memasukkan *subdomain* dan menggunakan beberapa filter yang disediakan untuk melakukan pencarian data yang lebih spesifik, beberapa filter yang diberikan adalah tahun penelitian *(years)* adalah 2019 untuk mendapatkan penelitian terbaru, dan tipe artikel adalah *research article*, dari pencarian tersebut diambil masing-masing *subdomain* adalah 100 data artikel penelitian berupa teks judul dan abstrak.

Proses pencarian dilakukan secara berulang hingga semua data dengan pencarian berdasarkan *subdomain* telah selesai, hasil pencarian data kemudian disimpan dalam bentuk *file* berisikan teks dengan tipe *file* yang berbeda untuk memudahkan dalam proses selanjutnya,

masing-masing tipe *file* menginisiasikan *subdomain* penelitian pada domain penelitian *computer science.* Nama *file* dan *subdomain* penelitian dari hasil pengumpulan data dijelaskan pada tabel 1 berikut :

Tabel 1. *Filename* dan *Subdomain* Hasil Pengumpulan Data

Filename	***Subdomain***
abstract.ai	*Artificial Intelligence*
abstract.cgcad	*Computer Graphics and Computer-Aided Design*
abstract.cnc	*Computer Networks and Communications*
abstract.cs	*Computer Science*
abstract.ctm	*Computational Theory and Mathematics*
abstract.cvpr	*Computer Vision and Pattern Recognition*
abstract.ha	*Hardware and Architectire*
abstract.hci	*Human-Computer Interaction*
abstract.is	*Informations Systems*
abstract.s	*Software*
abstract.sp	*Signal Processing*

2.1.2 Tahap *Data Preprocessing*

Pada tahapan *data preprocessing* dibagi menjadi dua tahapan, yaitu pengolahan data berupa *dataset* teks abstrak penelitian yang akan digunakan dalam proses *training*, dan pengolahan data berupa abstrak penelitian yang akan digunakan dalam proses evaluasi.

Tahap data *preprocessing* dimulai dengan mengolah data abstrak penelitian bidang *computer science*, `load_data_and_labels` merupakan fungsi untuk menghasilkan data dan label, data ditampilkan dalam bentuk *list* dengan membuka dan membaca *file* data dalam format *encoding latin-1*, variabel data abstrak dan label ditunjukkan pada tabel 2 berikut :

Tabel 2. *Varible data*

Subdomain	***Variable data***	***Label data***
Artificial Intelligence	ai_examples	ai_labels `[0 0 0 0 0 0 0 0 0 0 1]`
Computational Theory and Mathematics	ctm_examples	ctm_labels `[0 0 0 0 0 0 0 0 0 1 0]`
Computer Graphics and Computer-Aided Design	cgcad_examples	cgcad_labels `[0 0 0 0 0 0 0 0 1 0 0]`
Computer Networks and Communications	cnc_examples	cnc_labels `[0 0 0 0 0 0 0 1 0 0 0]`
Computer Science	cs_examples	cs_labels `[0 0 0 0 0 0 1 0 0 0 0]`
Computer Vision and Pattern Recognition	cvpr_examples	cvpr_labels `[0 0 0 0 0 1 0 0 0 0 0]`
Hardware and Architectire	ha_examples	ha_labels `[0 0 0 0 1 0 0 0 0 0 0]`
Human-Computer Interaction	hci_examples	hci_labels `[0 0 0 1 0 0 0 0 0 0 0]`
Informations Systems	is_examples	is_labels `[0 0 1 0 0 0 0 0 0 0 0]`
Signal Processing	sp_examples	sp_labels `[0 1 0 0 0 0 0 0 0 0 0]`
Software	s_examples	s_labels `[1 0 0 0 0 0 0 0 0 0 0]`

Setelah mendefinisikan variabel pada data penelitian, data yang berisikan teks kemudian diolah dengan menggunakan fungsi $clean\_str$ untuk mengubah teks menjadi huruf kecil, menghapus angka dan tanda baca, hasil dari proses tersebut di definisikan dengan variabel x_text dan label yang digunakan di definisikan dengan variabel y yang merupakan matriks $1x11$. Proses yang berlangsung dirumuskan seperti sebagai berikut :

$$x\_text = ai\_examples + ctm\_examples + cgcad\_examples + cnc\_examples + cs\_examples + cvpr\_examples + ha\_example + hci\_examples + is\_examples + sp\_examples + s\_examples$$
$$x\_text = [clean\_str(sent)\ for\ sent\ in\ x\_text]$$
$$x\_text = [sent.split(" ")\ for\ sent\ in\ x\_text]$$
$$y = np.concatenate([ai\_labels, ctm\_labels, cgcad\_labels, cnc\_labels, cs\_labels, cvpr\_labels, ha\_labels, hci\_labels, is\_labels, sp\_labels, s\_labels], 0)$$

Selanjutnya fungsi $clean\_str$ akan diterapkan pada data yang berisikan teks tersebut dan hasil dari proses tersebut diberikan variabel x_text .

Proses selanjutnya adalah $pad\_senteces$ yang merupakan fungsi untuk menentukan parameter dengan mengisi kalimat yang berisi teks sehingga panjang semua kalimat sama dengan panjang kalimat maksimum.

$$sequence\_length = max(len(x)\ for\ x\ in\ sentences)$$
$$padded\_sentences = [\,]$$
$$for\ i\ in\ range\ (len(sentences)):$$
$$sentence = sentences[i]$$
$$num\_padding = sequence\_length - len(sentence)$$
$$new\_sentence = sentence + num\_padding * [padding\_word]$$
$$padded\_sentences.append(new\_sentence)$$

Hasil dari fungsi $pad\_sentences$ di definisikan dengan variabel $padded\_sentences$, dengan menambahkan *token* <PAD> khusus ke semua kalimat lain untuk membuat kalimat mempunyai jumlah kata yang sama, sehingga memungkinkan pengelompokan data dapat dilakukan secara efisien karena setiap contoh dalam sebuah *batch* harus memiliki panjang yang sama. Variabel $pad\_sentences$ akan digunakan dalam fungsi $build\_vocab$, yang bertujuan untuk merekonstruksi objek yang berisi semua informasi yang diperlukan dengan cara *serialisasi* dan *de-serialisasi* struktur objek kemudian menuliskannya ke *file* dan disimpan di *disk.*

$vocab\_dir$ merupakan variabel yang mendefinisikan letak lokasi *file* dari fungsi $build\_vocab$, jika *file* dengan format *.pkl* tidak ditemukan pada lokasi *(path)* yang telah di definisikan, maka proses akan diulangi kembali hingga *file* pada *path* ditemukan, *file* disimpan dengan nama vocab.pkl dengan direktori data\abstract\vocab.pkl.

Setelah *file* ditemukan dan berhasil dibuka maka proses selanjutnya yang dilakukan adalah *glossary, map indeks to word, map word to index,* proses dilakukan menggunakan modul *pickle* dan *itertools* pada pemrograman *python.* Hasil dari fungsi $build\_vocab$ adalah variabel $vocabulary$ dan $vocabulary\_inv$.

Proses selanjutnya adalah $build\_input$ yang merupakan fungsi untuk menempatkan data ke dalam *array* agar terstruktur, proses ditunjukkan dengan rumus berikut :

$$x = np.array([[vocabulary[word]\ for\ word\ in\ sentence]\ for\ sentece\ in\ sentences])$$

$$y = np.array(labels)$$

Variabel x merupakan *array* dari data berupa teks yang telah melalui *data preprocessing* dengan fungsi $clean\_str$, $load\_data\_and\_labels$, $pad\_sentences$ dan $build\_vocab$, dan variabel y merupakan *array* dari label yang telah didefinisikan sebelumnya dengan variabel berisi matriks.

Proses terakhir adalah mendefinisikan variabel yang akan digunakan dalam proses *data training*, proses tersebut di definisikan dengan fungsi $load\_data$, proses mendefinisikan variabel dapat dilihat sebagai berikut :

$$sentences, labels = load\_data\_and\_labels()$$
$$sentences\_padded = pad\_sentences(sentences)$$
$$vocabulary, vocabulary\_inv = build\_vocab()$$
$$x, y = build\_input(sentences\_padded, labels, vocabulary)$$

Pengolahan data berupa abstrak penelitian yang akan digunakan pada tahap *evaluate* termasuk ke dalam tahapan *data preprocessing*, tetapi memiliki proses yang berbeda dari pengolahan data sebelumnya.

$input\_x$ merupakan variabel *input* yang diberikan pada saat proses evaluasi, *input* berupa data abstrak yang belum memiliki label atau tidak memiliki *subdomain* yang akan di klasifikasikan. Selanjutnya *input* akan di proses menggunakan fungsi $build\_input\_abstract$ dan menghasilkan nilai *input_x.reshape(1,sequence_length)* proses ditunjukkan seperti sebagai berikut :

$$input\_x = input\_x.split(' ')$$
$$num\_padding = sequence\_length - len(input\_x)$$
$$input\_x = input\_x + num\_padding * [padding\_word]$$
$$input\_x = np.array([vocabulary[word]\ if\ word\ in\ vocabulary$$
$$else\ vocabulary[padding\_word]\ for\ word\ in\ input\_x$$

2.1.3 Tahap *Training Process*

Pada tahapan *training process*, data yang telah diolah pada proses sebelumnya akan digunakan dalam membangun sebuah model menggunakan *deep learning model convolutional neural network*. Tahapan *data training* dimulai dengan proses mengambil variabel dengan menggunakan fungsi $load\_data$, variabel yang akan digunakan sebagai *input* dalam proses selanjutnya adalah :

$x = (11277, 135)$
y = (11277, 11)
$vocabulary = 14114$
$vocabulary\_inv = 14114$

Setelah variabel berhasil di *load* maka proses selanjutnya adalah *dividing data sets* atau membagi set data yang akan digunakan dalam proses *training* dan *testing*, proses pembagian *dataset* dirumuskan sebagai berikut :

$$X\_train, X\_test, y\_train, y\_test = train\_test\_split\ (x, y, text\_size = 0.2\ , random\_state = 42)$$

Pembagian *dataset* menggunakan *split* dengan nilai 0.2 untuk data yang digunakan pada proses *testing*, sehingga variabel menghasilkan nilai sebagai berikut :

$$X\_train = (9021,135)$$
$$y\_train = (9021,11)$$
$$X\_test = (2256,135)$$
$$y\_test = (2256,11)$$

Setelah pembagian *dataset* dilakukan untuk proses *training* dan *testing* , maka variabel yang dibutuhkan sudah memiliki nilai atau *shape,* proses selanjutnya adalah menentukan parameter yang akan digunakan dalam pembentukan model, yaitu :

$$sequence\_length = x\_train.shape[1]$$
$$vocabulary\_size = len(vocabulary\_inv)$$
$$embedding\_dim = 256$$
$$filter\_sizes = [3,4,5]$$
$$num\_filters = 512$$
$$drop = 0.5$$
$$epoch = 100$$
$$batch\_size = 30$$

Parameter $sequence\_length$ merupakan panjang kalimat, $vocabulary\_size$ merupakan ukuran kosakata yang diperlukan untuk menentukan ukuran layer pada *embedding layer*, dan $embedding\_dim$ merupakan dimensi yang akan digunakan pada proses *embedding* sebelum model dibentuk. $Filter\_sizes$ adalah jumlah kata yang ingin disaring dengan filter *convolutional*. num_sizes merupakan variabel yang digunakan untuk jumlah filter per ukuran filter. $drop$ merupakan nilai yang digunakan pada *dropout* untuk menghindari *overfitting* pada model. $epoch$ merupakan jumlah iterasi yang akan digunakan pada seberapa banyak model akan dilatih *(training),* sehingga berdasarkan variabel di atas maka jumlah iterasi yang digunakan adalah 100 untuk melihat bagaimana perubahan pada *training loss* dan akurasi saat model dilatih. $batch\_size$ merupakan jumlah sampel yang digunakan dalam satu kali iterasi, proses ini mempercepat proses komputasi tetapi membutuhkan lebih banyak *memory*.

Proses selanjutnya adalah membentuk model, lapisan pertama adalah proses *word embedding* ke dalam vektor *low dimensional,* lapisan berikutnya adalah melakukan *convolution* pada *word vector* menggunakan beberapa ukuran filter, yaitu variabel $filter\_size$ yang telah di definisikan, yaitu menggeser 3, 4, atau 5 kata sekaligus.

Embedding layer, merupakan layer pertama dalam pembentukan model, yang memetakan indeks kata pada kosa kata menjadi representasi vektor *low dimensional,* (tabel pencarian yang dipelajari dari data), proses *embedding* dirumuskan sebagai berikut :

$$Inputs = Input(shape = (sequence\_length,), dtype = 'int32')$$
$$embedding = Embedding\ (input\_dim = vocabulary\_size, output\_dim = embedding\_dim, input\_length = sequence\_length)(inputs)$$
$$reshape = Reshape\ ((sequence\_length, embedding\_dim, 1))(embedding)$$

Hasil dari operasi *embedding* adalah tensor berbentuk 3 dimensi yaitu [*None*, *sequence_length*, *embedding_dim*] , pada operasi *convolutional 2D TensorFlow* membutuhkan *tensor* 4 dimensi dengan dimensi yang sesuai dengan *batch, width, height* dan *channel.* Tetapi pada hasil *embedding* tidak memiliki dimensi *channel* , sehingga ditambahkan secara manual

dan membentuk *layer* dengan *shape* [*None, sequence_length, embedding_dim,* 1], seperti dirumuskan pada variabel $reshape$ di atas.

Tahap selanjutnya dalam membentuk model adalah *convolution* dan *max-pooling layers*, karena menggunakan filter dengan ukuran yang berbeda, sehingga setiap *convolution* akan menghasilkan *tensor* dengan bentuk yang berbeda, proses akan diulangi dengan membuat masing-masing layer dan kemudian menggabungkan hasil menjadi satu vektor. *Convolution* dan *max-pooling layer* dirumuskan sebagai berikut :

$$
\begin{aligned}
conv\_0 = Conv2D(&num\_filters, kernel\_size \\
&= (filter\_sizes[0], embedding\_dim), padding \\
&= 'valid', kernel\_initializer = 'normal', activation \\
&= 'relu'\ (reshape) \\
conv\_1 = Conv2D(&num\_filters, kernel\_size \\
&= (filter\_sizes[1], embedding\_dim), padding \\
&= 'valid', kernel\_initializer = 'normal', activation \\
&= 'relu'\ (reshape) \\
conv\_2 = Conv2D(&num\_filters, kernel\_size \\
&= (filter\_sizes[2], embedding\_dim), padding \\
&= 'valid', kernel\_initializer = 'normal', activation \\
&= 'relu'\ (reshape) \\
maxpool\_0 = MaxPool2d(&pool\_size \\
&= (sequence\_length - filter\_sizes[0] + 1,1), striders \\
&= (1,1), padding = 'valid')\ (conv\_0) \\
maxpool\_1 = MaxPool2d(&pool\_size \\
&= (sequence\_length - filter\_sizes[1] + 1,1), striders \\
&= (1,1), padding = 'valid')\ (conv\_1) \\
maxpool\_2 = MaxPool2d(&pool\_size \\
&= (sequence\_length - filter\_sizes[2] + 1,1), striders \\
&= (1,1), padding = 'valid')\ (conv\_2)
\end{aligned}
$$

$$
\begin{aligned}
&concatenated\_tensor = \\
&Concatenate(axis = 1)\ ([maxpool\_0, maxpool\_1, maxpool\_2]) \\
&flatten = Flatten()\ (concatenated\_sensor)
\end{aligned}
$$

Setiap *filter* menerapkan *embedding* , tetapi berapa banyak kata yang tercakup adalah bervariasi. $Padding = 'valid'$ berarti bahwa menggeser filter di atas kalimat tanpa *padding edges*, sehingga memberikan *output shape* [*sequence_length-filter_size* +1,1,1].

Kemudian *max-pooling* diterapkan pada *output* dengan spesifik filter tertentu dan menghasilkan *feature vector*, di mana semua dimensi terakhir sesuai dengan yang dibutuhkan, semua *output tensor* yang dikumpulkan dari setiap *filter size* kemudian digabungkan menjadi satu *feature vector* menggunakan *concatenated_tensor* dan meratakan dimensi menggunakan *flatten.*

Tahap selanjutnya adalah *dropout layer*, *dropout* adalah metode yang paling populer untuk meregulasi *convolutional neural network,* secara stokastik menonaktifkan sebagian kecil dari neuron dan mencegah neuron untuk beradaptasi secara bersamaan, dan memaksa untuk mempelajari fitur yang bermanfaat secara individual, fraksi neuron yang diaktifkan telah di definisikan sebelumnya pada parameter, yaitu variabel $drop$ dengan nilai 0.5.

Setelah *dropout* diterapkan, maka *feature vector* dari *max-pooling* dapat digunakan untuk menghasilkan prediksi dengan melakukan perkalian matriks dan memilih kelas dengan skor tertinggi, selanjutnya menggunakan fungsi aktivasi *softmax* untuk mengubah skor menjadi

probabilitas yang dinormalisasi pada *output* yang dihasilkan dan merupakan layer yaitu *dense layer*, lapisan yang paling umum digunakan dan terhubung dengan semua *weight* dan *bias.*

Proses selanjutnya setelah model dibentuk adalah *training data* dan menggunakan *chekcpoint* untuk menyimpan hasil dari *training.* Proses *training* dilakukan pada *TensorFlow* menggunakan CPU, dan menggunakan Adam *optimizer* untuk mengoptimalkan *loss function* pada jaringan yang digunakan. *Checkpoint* merupakan fitur yang disediakan oleh *TensorFlow* yang memungkinkan untuk menyimpan prameter model dan menggunakannya kembali, penggunaan *chekcpoint* bermanfaat untuk memilih pengaturan parameter terbaik dan menghentikan proses yang berlangsung, *checkpoints* dibuat dengan menyimpan objek. Fungsi *checkpoint* ditunjukkan sebagai berikut :

$$checkpoint = ModelCheckpoint('results/abstract.weights.\{epoch:03d\} - \{val\_acc:.4f\}.hdf5', monitor = 'val\_acc', verbose = 1, save\_best\_only = True, mode = 'auto')$$

Adam optimizer digunakan dengan paramater sebagai berikut :

$$adam = Adam(lr = 1e-4, beta\_1 = 0.9, beta\_2 = 0.999, epsilon = 1e-08, decay = 0.0)$$

Proses *training* berlangsung hingga iterasi ke-25, dan pada iterasi-21 (Epoch 21/100) tidak mengalami penambahan akurasi sehingga proses *training* dihentikan. Hasil dari proses *training data* disimpan menjadi *file* abstract.weights.021-0.9522.hdf5 dengan hasil akurasi adalah 0.9522 dan *loss* 0.1493.

2.1.4 Tahap *Evaluate*

Pada tahapan *evaluate*, diberikan *input* berupa data abstrak penelitian *computer science* yang belum di klasifikasi dan memiliki *subdomain.* Proses dimulai dengan mendefinisikan variabel $abstract\_model$ yang berisikan model yang disimpan pada tahapan *training data*, dan mendefinisikan fungsi $predict\_abstract$ ke dalam variabel $abstract\_model$, proses ditunjukkan sebagai berikut :

$$abstract\_model = load\_model('result/abstract.weights.021 - 0.9522.hdf5')$$
$$predict\_abstract = (abstract\_model)$$

Setelah *model* berhasil di *load* maka proses selanjutnya adalah mendefinisikan fungsi yang akan digunakan dalam proses *prediction* yaitu $predict\_abstract(model)$, dengan memberikan *input* berupa data abstrak yang belum di klasifikasikan dan di definisikan sebagai variabel $input\_x$. Jika *input* yang diberikan adalah *exit* maka akan melakukan eksekusi perintah keluar *(exit)* dan proses *prediction/evaluate* tidak dilanjutkan, proses tersebut dijelaskan sebagai berikut :

$$If\ input\_x = 'exit'\ :$$
$$exit()$$

Jika *input* yang diberikan selain *exit* maka data akan di proses menggunakan fungsi yang telah di definisikan pada tahapan *data preprocessing* yaitu fungsi *build_input_abstract*, fungsi diterapkan pada variabel $input\_x$, seperti ditunjukkan sebagai berikut :

$$input\_x = build\_input\_abstract(input\_x)$$

Kemudian di dapatkan *output* berupa $input\_x.shape$, proses selanjutnya adalah melakukan prediksi kelas/*subdomain* pada data *input* yang diberikan, *subdomain* hasil prediksi di definisikan dengan variabel y_pred, proses dirumuskan sebagai berikut :

$$y\_pred = model.predict(input\_x)$$
$$result = list(y\_pred[0])$$

Hasil yang di dapatkan di definisikan dengan variabel $result$, dan hasil disajikan dalam bentuk *list*, kemudian dimasukkan ke dalam *rule* yang telah di deskripsikan, yaitu :

$Condition\ 1, Condition\ 2, \ldots, Condition\ n$ terdapat 11 *rule* yang menghasilkan *subdomain* dengan tingkat probabilitas yang dihasilkan.

3. HASIL DAN PEMBAHASAN

Hasil penelitian yang dilakukan terhadap klasifikasi abstrak penelitian bidang *computer science* menggunakan *deep learning model convolutional neural network 2D* dengan *subdomain* sebanyak 11, masing-masing *subdomain* terdiri dari 100 data berupa teks abstrak penelitian yang kemudian di proses dan menghasilkan sebuah model yang memiliki beberapa layer dengan *output shape* dan parameter yang dihasilkan. Model yang dihasilkan dapat dilihat pada struktur model *convolutional neural network* berikut :

Tabel 3. Struktur Model CNN

Layer (type)	**Output Shape**	**Param #**
input_1 (InputLayer)	(None ,135)	0
embedding_1 (Embedding)	(None, 135, 256)	3613184
reshape_1 (Reshape)	(None, 135, 256, 1)	0
conv2d_1 (Conv2D)	(None, 133, 1, 512)	393728
conv2d_2 (Conv2D)	(None, 132, 1, 512)	524800
conv2d_3 (Conv2D)	(None, 131, 1, 512)	655872
max_pooling2d_1 (MaxPooling2D)	(None, 1, 1, 512)	0
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 512)	0
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 512)	
concatenate_1 (Concatenate)	(None, 3, 1, 512)	0
flatten_1 (Flatten)	(None, 1536)	0
dropout_1 (Dropout)	(None, 1536)	0
dense_1 (Dense)	(None, 11)	16907

Model yang dihasilkan akan digunakan pada sistem yang terdiri dari satu aktor yaitu *user*. *User* dapat memberikan *input* berupa teks abstrak dokumen penelitian yang belum diketahui kategorinya atau *subdomain* penelitiannya, dan selanjutnya sistem akan mengklasifikasikan secara otomatis dengan tingkat probabilitas yang dihasilkan. *User interface* pada klasifikasi teks menggunakan model *deep learning* hanya terdiri dari satu *form* utama yaitu klasifikasi *(classification)*.

3.1 Hasil Penelitian

Terdapat beberapa tahapan dalam mengimplementasikan *convolutional neural network* untuk melakukan pengolahan data berupa teks *natural language processing*, yaitu *training* yang merupakan tahapan utama untuk melatih jaringan dengan mempelajari *input* data yang diberikan, proses selanjutnya adalah melakukan uji coba dan validasi dengan memberikan *input*

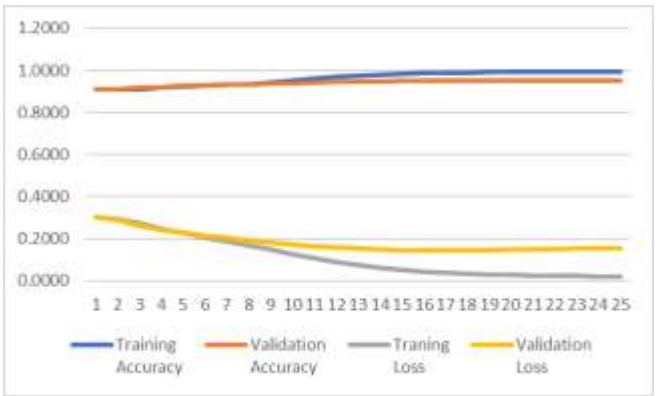
berupa teks abstrak penelitian yang belum memiliki *label/subdomain* penelitian, atau menggunakan *dataset* sebagai data *testing*.

Pada penelitian ini proses *training* dan *validation* dilakukan secara bersamaan dengan membagi *dataset* yang digunakan dengan persentase data yang digunakan dalam proses *training* adalah 80% dari total keseluruhan data berupa teks yaitu dengan total kalimat *(sentence)* adalah 9021 kalimat, dan 20% data digunakan untuk proses *validation* adalah 2256 kalimat. Komputasi dilakukan dengan menggunakan CPU. Hasil *training* dan *validation* serta waktu yang dibutuhkan dalam proses *training* pada model yang dibangun ditunjukkan pada tabel 5 berikut :

Tabel 4. Hasil *Training* dan *Validation*

Epoch	***Training Accuracy***	***Validation Accuracy***	***Traning Loss***	***Validation Loss***	***Time***
1/100	0.9091	0.9091	0.3039	0.3010	832s 92ms/step
2/100	0.9091	0.9091	0.2946	0.2906	835s 93ms/step
3/100	0.9114	0.9174	0.2723	0.2624	964s 107ms/step
⋮	⋮	⋮	⋮	⋮	⋮
20/100	0.9919	0.9517	0.0270	0.1474	339s 38ms/step
21/100	0.9923	0.9522	0.0249	0.1493	340s 38ms/step
22/100	0.9928	0.9518	0.0228	0.1501	339s 38ms/step
23/100	0.9927	0.9520	0.0218	0.1524	346s 38ms/step
24/100	0.9928	0.9516	0.0212	0.1532	338s 37ms/step
25/100	0.9930	0.9520	0.0200	0.1552	337s 37ms/step

Dalam proses *training* dan *validation* yang dilakukan terdapat parameter yang diberikan, di antaranya adalah jumlah *epoch* yaitu 100, sehingga proses *training* dilakukan hingga 100 iterasi, berdasarkan grafik pada gambar 4.1 di iterasi ke-8 *(epoch 8/100)* hasil dari *training accuracy* mulai melebihi *validation accuracy* dan terus meningkat hingga pada iterasi ke-24 model menunjukkan perbedaan terbesar yaitu 0.0412. berdasarkan hasil yang ditunjukkan pada tabel 4.2 akurasi terbesar dicapai pada iterasi ke-21 yaitu dengan tingkat akurasi *(validation accuracy)* 0.9522 dan selanjutnya tidak mengalami peningkatan sehingga proses *training* dihentikan pada iterasi ke-25. Grafik yang menunjukkan *training accuracy, training loss, validation accuracy* dan *validation loss* ditunjukkan pada gambar 1 berikut :



Gambar 1. Grafik *Training Accuracy*, *Validation Accuracy*, *Training Loss* dan *Validation Loss*

Dari model yang dihasilkan kemudian diimplementasi ke dalam hasil perancangan sistem di mana aplikasi siap untuk dioperasikan pada keadaan yang sebenarnya, sehingga dapat diketahui apakah model yang telah dibangun dapat menghasilkan *output* sesuai dengan tujuan yang diharapkan.

Berdasarkan model yang dibangun dan rancangan *user interface* yang telah dibuat, maka model yang digunakan untuk melakukan klasifikasi teks abstrak dokumen penelitian dengan domain *computer science* dan 11 *subdomain* penelitian yaitu *artificial intelligence, computational theory and mathematics, computer graphics and computer-aided design, computer networks and communications, computer science, computer vision and pattern recognition, hardware and architecture, human-computer interaction, information systems, signal processing, software* .

3.2 Pembahasan

Berdasarkan hasil pengujian model *deep learning convolutional neural network* yang dilakukan melalui sistem aplikasi klasifikasi teks abstrak penelitian bidang penelitian *computer science* menggunakan *convolutional neural network deep learning model* dapat disimpulkan memiliki beberapa kelebihan dan kelemahan sebagai berikut :

Kelebihan model *deep learning* yang dihasilkan :

1. Model *convolutional neural network* mampu melakukan ekstraksi fitur yang relevan secara otomatis terhadap tugas yang diberikan.
2. Model yang dihasilkan tidak mengalami *overfitting* karena menerapkan metode *dropout* untuk meregularisasi *convolutional neural network* sehingga di dapatkan hasil akhir model memiliki akurasi *training* 0.9923 dan akurasi *validation* 0.9522, sedangkan *training loss* 0.0249 dan *validation loss* 0.1493.
3. Dapat dilakukan pengaturan parameter pada *convolutional neural network* yang digunakan pada saat pelatihan data dan melakukan uji coba pada model untuk mendapatkan akurasi yang terbaik, karena model menggunakan fitur *checkpoint TensorFlow* untuk menyimpan parameter model.

Kelemahan model yang dihasilkan :

1. Memerlukan biaya komputasi yang tinggi, pada tahap *data training*, proses dilakukan dengan memanfaatkan *TensorFlow* berbasis CPU sehingga membutuhkan waktu lebih lama dibandingkan dengan menggunakan GPU.
2. *Convolutional nerual network (CNN)* memiliki ketergantungan pada parameter dan jumlah parameter yang diberikan untuk memberikan nilai awal pada saat deklarasi variabel sesuai dengan masalah yang dihadapi, dan membutuhkan beberapa pengetahuan ahli dalam domain tersebut.
3. Model yang dihasilkan hanya dapat mengolah *input* yang diberikan dengan jumlah maksimum 135 *words* karena model yang terbentuk dari proses *training data*.

4. SIMPULAN

Berdasarkan uraian pada bab-bab sebelumnya, maka kesimpulan yang diperoleh dari hasil penelitian ini yaitu :

1. Model *deep learning* yang dihasilkan mampu mengklasifikasikan abstrak penelitian dalam bidang *computer science,* ke dalam beberapa *subdomain* penelitian menggunakan model *deep learning convolutional neural network* dengan tingkat akurasi 0.9522 dan *loss* 0.1493, validasi dilakukan dengan melakukan *split* pada *dataset*, perbandingan 80% untuk *data training* dan 20% untuk *data testing*.
2. *Convolutional neural network* yang merupakan *model deep learning* pada umumnya diterapkan pada pengolahan gambar atau dalam bidang *computer vision* dapat diterapkan ke dalam *natural language processing*.
3. Abstrak penelitian tidak sepenuhnya merepresentasikan domain penelitian, dari hasil uji coba yang dilakukan dengan memberikan *input* berupa teks abstrak dokumen di dapatkan hasil memiliki lebih dari satu *subdomain* penelitian, walaupun *subdomain* lainnya memiliki tingkat probabilitas yang rendah, sehingga dapat disimpulkan bahwa suatu penelitian memungkinkan untuk memiliki lebih dari satu label (*subdomain*

penelitian), tetapi tetap memiliki satu *subdomain* penelitian yang menjadi pokok pembahasan dengan tingkat probabilitas yang tinggi.

4. Model yang dibangun dapat memberikan wawasan tentang penelitian pada bidang *computer science* beserta *sub* bidang penelitiannya dan membantu dalam proses identifikasi *file* dokumen penelitian *computer science.*

DAFTAR PUSTAKA

[1] Bai, Xiang et al. 2017. "Text/Non-Text Image Classification in the Wild with Convolutional Neural Networks." *Pattern Recognition*.

[2] Chaturvedi, Akshay, Onkar Pandit, and Utpal Garain. 2019. "CNN for Text-Based Multiple Choice Question Answering."

[3] Conneau, Alexis, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. "Very Deep Convolutional Networks for Natural Language Processing." *EACL*.

[4] Du, Jiachen, Lin Gui, Ruifeng Xu, and Yulan He. 2018. "A Convolutional Attention Model for Text Classification." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*,.

[5] Gambäck, Björn, and Utpal Kumar Sikdar. 2017. "Using Convolutional Neural Networks to Classify Hate-Speech." In *Proceedings of the First Workshop on Abusive Language Online*,.

[6] Johnson, Rie, and Tong Zhang. 2015. "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks." In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,.

[7] Lai, S., L. Xu, K. Liu, and J. Zhao. 2015. "Recurrent Convolutional Neural Networks for Text Classification." In *Proceedings of the National Conference on Artificial Intelligence*,.

[8] Maggiori, Emmanuel, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. 2016. "Fully Convolutional Neural Networks for Remote Sensing Image Classification." In *International Geoscience and Remote Sensing Symposium (IGARSS)*,.

[9] Wang, Peng et al. 2015. "Semantic Clustering and Convolutional Neural Network for Short Text Categorization."

[10] Xiao, Liqiang et al. 2018. "Transformable Convolutional Neural Network for Text Classification." In *IJCAI International Joint Conference on Artificial Intelligence*,.