Implementasi Sistem Konfigurasi *Router* Berbasis *Natural Language Processing* dengan Pendekatan *Low Rank Adaptation Finetuning* dan *8-Bit Quantization*

Hanung Addi Chandra Utomo[1], Yuris Mulya Saputra[1,*], Agi Prasetiadi[2]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
hanungutomo45@mail.ugm.ac.id
[2]Program Studi Teknik Informatika, Institut Teknologi Telkom Purwokerto;
agi@ittelkom-pwt.ac.id
*Korespondensi: ym.saputra@ugm.ac.id;

Abstract* - Router configuration is one of the important things in a computer network. This process requires an understanding of the language and special syntax which can take a long time for someone who is not used to it. The application of Natural language processing can help overcome this problem. To achieve the goal of this implementation, Finetuning needs to be done on existing models such as the GPT-J-6B model which has been trained using 6 billion parameters. By using a dataset consisting of router configurations, it is hoped that the finetuning process can improve the performance of the model to detect the intent of the input text in natural language which can then provide commands that match the commands given. Apart from that, the use of other techniques such as Low Rank Adaptation (LoRA) can be used to optimize the Finetuning process to make it more efficient without reducing model performance, and the use of 8-bit quantization techniques to minimize resource usage when running models. With these techniques, the finetuning process can be carried out stably within Google Collaboratory. Therefore, with the implementation of NLP on this router configuration and the techniques above, you can increase the effectiveness of network management by using time and resources efficiently. Through this research, an NLP-based router configuration model was successfully obtained with an accuracy of 98%.

Keywords: Natural language processing, Router Configuration, Finetuning, LoRA, 8Bit Quantization

Intisari - Konfigurasi *Router* merupakan salah satu hal penting dalam jaringan komputer. Proses ini memerlukan pemahaman tentang bahasa dan sintaks khusus yang dapat memakan waktu lama bagi seseorang yang tidak terbiasa. Penerapan *Natural language processing* bisa membantu mengatasi masalah ini. Untuk mencapai tujuan dari penerapan ini, *Finetuning* perlu dilakukan pada *model* yang ada seperti *model* GPT-J-6B yang telah dilatih menggunakan 6 milyar parameter. Dengan menggunakan dataset yang terdiri dari konfigurasi *router*, diharapkan proses *finetuning* bisa meningkatkan performa *model* untuk mendeteksi maksud dari *input text* dalam Bahasa natural yang kemudian bisa memberikan *command-command* yang sesuai dengan perintah yang diberikan. Selain itu, penggunaan teknik lain seperti *Low Rank Adaptation* (LoRA) yang dapat digunakan untuk mengoptimalkan proses *Finetuning* agar lebih efisien tanpa mengurangi performa *model*, dan penggunaan teknik *8-bit quantization* untuk memperkecil penggunaan *resource* saat menjalankan *model*. Dengan beberapa teknik ini, proses *finetuning* dapat dilakukan dengan stabil dalam *Google Colaboratory*. Oleh karena itu, dengan implementasi NLP pada konfigurasi *router* ini dan teknik-teknik di atas, dapat meningkatkan efektivitas pengelolaan jaringan dengan menggunakan waktu dan sumber daya yang efisien. Melalui penelitian ini berhasil didapatkan model konfigurasi router berbasis NLP dengan akurasi sebesar 98%.

Kata Kunci: *Natural language processing*, Konfigurasi *Router*, *Finetuning*, LoRA, *8Bit Quantization*

I. PENDAHULUAN

Dalam konfigurasi jaringan, keefektifan konfigurasi secara otomatis dibandingkan dengan konfigurasi secara manual akan jauh lebih baik dikarenakan konfigurasi secara manual memiliki banyak kekurangan [1]. Dengan tujuan untuk mengurangi intervensi manusia, beban kerja berulang dan *error*, mereka mendapati bahwa penelitian ini mengurangi waktu yang diperlukan sebesar tiga kali lipat dibandingkan dengan pendekatan manual. Diteliti otomatisasi jaringan jauh lebih baik jika dibandingkan dengan pendekatan manual [2]. Otomatisasi konfigurasi jaringan sudah banyak digunakan, seperti contohnya pembuatan aplikasi berbasis *Dashboard* untuk manajemen jaringan berbasis *Software-Defined Network* (SDN) [3] sudah cukup membantu teknisi beralih dari konfigurasi jaringan secara manual langsung dari *router* fisik. Akan tetapi, sering kali dijumpai kesalahan konfigurasi karena kelalaian teknisi atau ketidaktahuan teknisi dalam menggunakan *Dashboard* tersebut, sehingga diperlukan *resource* lagi untuk melakukan *training* atau pelatihan. Sehingga, bisa dikatakan kurang efektif jika sering terjadi pergantian teknisi. Maka dari itu, penggunaan *Artificial Intelligence* ini diharapkan bisa lebih mengefisiensikan tahap tersebut.

Kemudian melihat dari perkembangan *Artificial Intelligence* atau Kecerdasan Buatan pada tahun 2023, seperti ChatGPT [4] dan *Generative Model* lainnya [5], telah mendukung beberapa inovasi baru terutama dalam bidang *Networking*. Salah satu inovasi tersebut adalah Sistem *Intent-Based Networking* (IBN) yang dibangun dari sistem *Software-Defined Networking* (SDN) [6] dengan memanfaatkan kecerdasan buatan untuk manajemen jaringan. Dengan adanya kecerdasan buatan, konfigurasi jaringan dapat dilakukan secara otomatis berdasarkan kebutuhan pengguna, sehingga memudahkan pengelolaan jaringan dan meningkatkan efisiensi.

Di sisi lain biaya pengembangan *Artificial intelegence* bisa dibilang cukup mahal. Beberapa *large language model* terkini seperti GPT 3 dan 4 yang dipakai *ChatGPT* memerlukan komputasi yang cukup besar. Hal ini tidak akan bisa dilakukan pada personal komputer maupun *Google Colaboratory*. Oleh karena itu, diperlukan cara yang efektif dan murah selain menggunakan *model* ini.

Sudah banyak penelitian yang dilakukan menggunakan beberapa algoritma *machine learning* [7, 8], seperti *DRL*, *Deep Learning*, *Supervised Learning*, dan lain lain. Akan tetapi, dijelaskan bahwa algoritma ini masih sebatas *black box* yang sulit dimengerti, sehingga tidak diketahui kemungkinan sistem ini akan terus bekerja [9].

Algoritma *Machine learning* terus berkembang sampai di titik di mana mesin dapat mengolah bahasa natural manusia, perkembangan ini juga turut berdampak pada bidang jaringan internet. Telah dikumpulkan beberapa data dari jurnal terkait *Natural Language Processing* dilakukan dikonfigurasi *router*. Dari data yang didapatkan, pelatihan NLP untuk konfigurasi *router* bisa dibilang tidak praktikal untuk dilakukan. Selama alat untuk NLP dikonfigurasi jaringan belum sempurna, penggunaan NLP akan terlihat tidak menarik karena masih diperlukan pengecekan *command* hasil generasi [10].

Oleh karena itu, perlu dilakukan penelitian lebih lanjut dengan menggunakan metode dan *model* yang tersedia untuk menyempurnakan NLP untuk konfigurasi jaringan. didukung dengan metode seperti *Low Rank Adaptation* [11] yang dapat digunakan untuk meningkatkan keefektivitasan waktu dan *resource* pelatihan *model*.

Penelitian ini bertujuan untuk membangun sebuah sistem konfigurasi *router* berbasis AI, dengan menggunakan *model Large language model* GPT-J [12] yang telah dilatih untuk bisa memberikan perintah-perintah yang biasa digunakan di dalam sebuah *router* saat melakukan konfigurasi. Dengan demikian, alur konfigurasi sebuah *router* akan lebih efektif dan mudah dipahami.

II. DASAR TEORI

A. Konfigurasi Router

Router adalah perangkat jaringan yang menghubungkan dan memfasilitasi komunikasi antara dua jaringan yang berbeda. Fungsi utamanya adalah untuk menentukan jalur terbaik untuk mengirimkan paket data melalui jaringan internet yang terhubung [13].

Sedangkan, konfigurasi *router* adalah proses mengatur perangkat jaringan yang disebut *router* untuk mengarahkan lalu lintas data di dalam jaringan komputer.

B. Natural Language Processing

Natural Language Processing merupakan salah satu fokus *machine learning* yang digunakan untuk memahami bahasa manusia, sehingga mesin bisa menganalisa dan menyelesaikan permasalahan yang ada sesuai dengan konteks yang diberikan [14].

C. Large Language Model

Large language model merupakan model yang dilatih menggunakan dataset text berbahasa natural dalam jumlah yang besar, Biasanya model ini digunakan dalam penerapan *Natural Language Processing* yang memerlukan dataset yang besar [15]. Karena dilatih menggunakan dataset yang besar, tentunya parameter model harus menyesuaikan sehingga biasanya LLM menggunakan ratusan maupun jutaan parameter yang tentu membuatnya berukuran besar dan diperlukan komputasi yang besar untuk menjalankan modelnya.

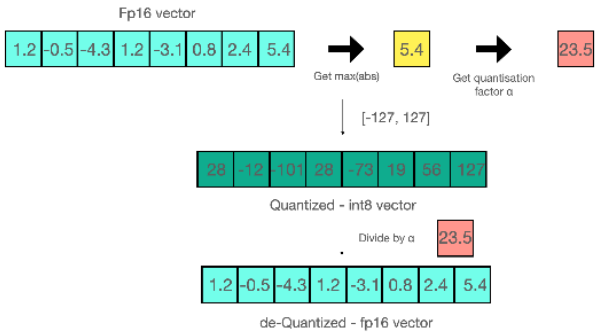
Generative Pre-Trained Transformers adalah model *natural language* yang telah dikembangkan oleh OpenAI [16]. GPT menggunakan arsitektur transformers, arsitektur ini memperkenalkan mekanisme perhatian (*attention mechanism*) yang memungkinkan model untuk memperhitungkan hubungan antara kata-kata dalam kalimat. Contoh model GPT oleh OpenAI yang sudah dikembangkan yaitu GPT-2, GPT-3 dan GPT-4. Selain itu ada juga model GPT yang telah dikembangkan secara *open source* oleh EleutherAI yaitu GPT-J-6B. Model ini memiliki parameter sebesar 6.053.381.344 parameter. Menurut *repository* GPT-J-6B [12] berikut ini merupakan dokumentasi performa dari model GPT-J dibandingkan dengan model GPT-3 varian 2.7B dan 6.7B.

D. Low Rank Adaptation

Teknik *Low Rank Adaptation* [11] bertujuan untuk mengurangi kompleksitas dan ukuran *LLM* tanpa mengorbankan kinerjanya secara signifikan. Pendekatan ini memanfaatkan sifat struktural dari *LLM* untuk mengidentifikasi dan mengeliminasi komponen yang kurang signifikan dalam representasi model. Berikut hasil training yang telah dilakukan di model GPT-3.

E. 8-Bit Quantization

8-bit quantization adalah teknik yang membatasi representasi parameter model menggunakan 8 bit, *Model* diubah dari bentuk *FP32* atau *floating-point 32* menjadi 8bit menggunakan *snippet code* dari gambar 4 dengan metode *absmax quantization* yang menggunakan *scaling factor* dari range 8bit yaitu 127 dibagi dengan nilai absolut tertinggi dari vektor *weight model* [17] seperti yang dijelaskan pada gambar 1 di bawah ini.



Gambar 1. *Absmax Quantization*

III. METODOLOGI

A. Alat

Di bawah ini merupakan alat yang digunakan pada penelitian ini:

- Laptop

Tabel 1. Spesifikasi Laptop

Tipe	Asus Nitro 5 AN515
Sistem Operasi	Windows 10
Processor	Intel i5 gen 10
RAM	16 GB

- *Google Colaboratory*

Tabel 2. Spesifikasi Google Colaboratory

Tipe	Free Colab
VGA	12 GB
RAM	Nvidia T400 16GB

- Koneksi Internet > 1Mbps

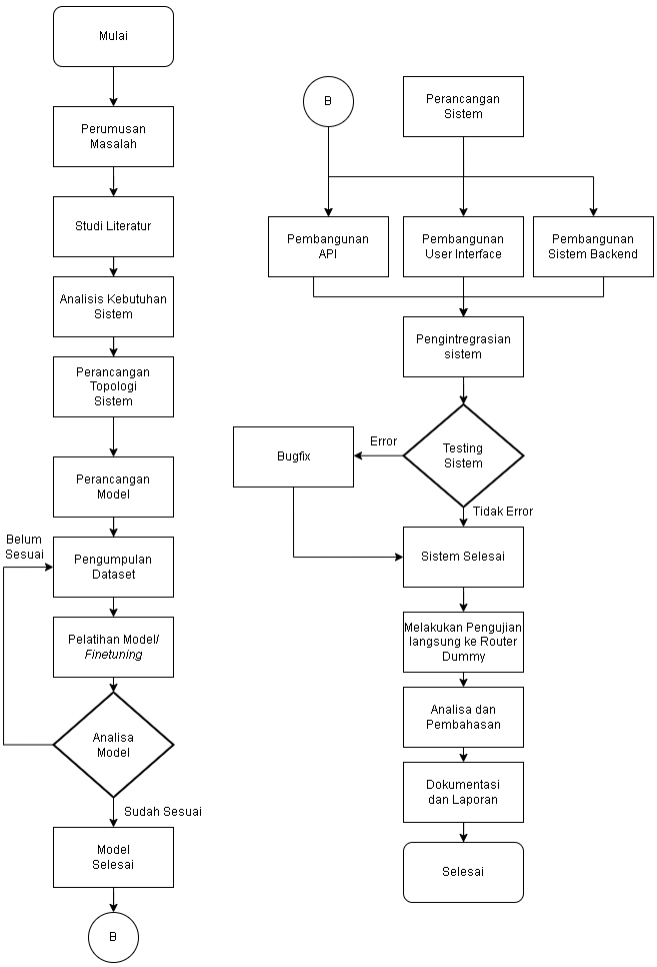
B. Bahan

Di bawah ini merupakan Bahan yang digunakan pada penelitian ini:

- *Google Colaboratory Notebook*
- *Visual Studio Code*
- *Huggingface*
- *Python*
- *PyTorch*
- *Gradio*
- *Flask*

C. Diagram Alir Penelitian

Secara ringkas alur penelitian secara singkat dapat dilihat pada gambar 2, alur penelitian dibagi menjadi 4 bagian seperti awalan, perancangan, testing, dan analisa.

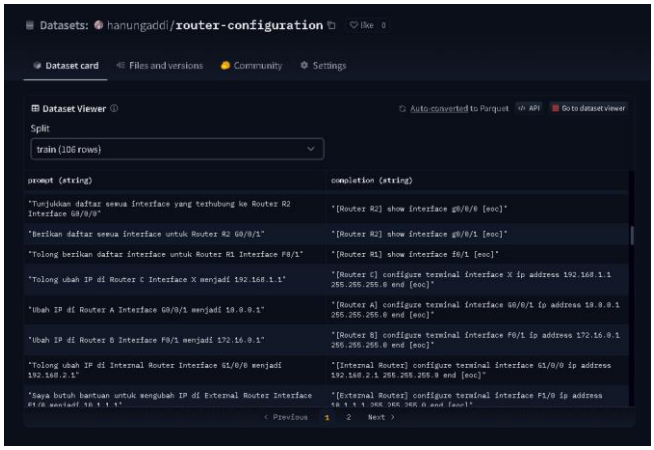


Gambar 2. Alur Penelitian

D. Pelatihan *Model* Konfigurasi *Model*

Pelatihan sebuah *model* supaya dapat digunakan untuk membantu melakukan konfigurasi menggunakan Bahasa natural, digunakan *model* GPT-J-6B yang akan dilakukan finetuning menggunakan dataset konfigurasi *router* yang telah dilabeli Bahasa natural dari fungsi *command* konfigurasi *router* tersebut. Dataset yang digunakan dapat dilihat pada gambar 3 berupa dataset yang disimpan pada *repository* huggingface, dataset ini dibuat secara manual dengan cara menambahkan label berupa bahasa natural pada *command* yang akan dipakai.

Selain itu, untuk mengurangi *resource* yang digunakan seperti umumnya *model* LLM yang memerlukan *resource* yang besar, akan digunakan teknik *8bit quantization* dan *Low-Rank Adaptation* saat melakukan finetuning. Gambar 4 merupakan *code* yang digunakan untuk konversi model menjadi model 8bit.



Gambar 3. Dataset Konfigurasi *Router*

```
def convert_to_int8(model):
    """Convert linear and embedding modules to 8-bit with optional adapters"""
    for module in list(model.modules()):
        for name, child in module.named_children():
            if isinstance(child, nn.Linear):
                print(name, child)
                setattr(
                    module,
                    name,
                    FrozenBNBLinear(
                        weight=torch.zeros(child.out_features, child.in_features, dtype=torch.uint8),
                        absmax=torch.zeros((child.weight.numel() - 1) // 4096 + 1),
                        code=torch.zeros(256),
                        bias=child.bias,
                    ),
                )
            elif isinstance(child, nn.Embedding):
                setattr(
                    module,
                    name,
                    FrozenBNBEmbedding(
                        weight=torch.zeros(child.num_embeddings, child.embedding_dim, dtype=torch.uint8),
                        absmax=torch.zeros((child.weight.numel() - 1) // 4096 + 1),
                        code=torch.zeros(256),
                    )
                )
```

Gambar 4. *Snippet Code 8bit Converter* [18]

Saat dilakukan *finetuning*, *Low Rank Adapter* akan diterapkan menggunakan *snippet code* pada gambar 5 di bawah ini. Fungsi di bawah ini akan menambahkan adapter pada *embedding layer*.

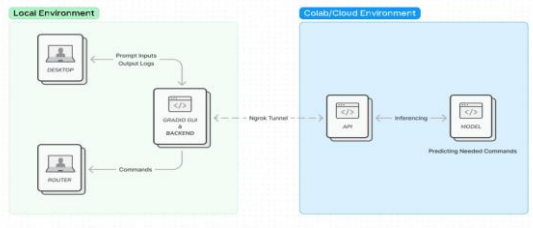
```
def add_adapters(model, adapter_dim=16):
    assert adapter_dim > 0

    for module in model.modules():
        if isinstance(module, FrozenBNBLinear):
            module.adapter = nn.Sequential(
                nn.Linear(module.in_features, adapter_dim, bias=False),
                nn.Linear(adapter_dim, module.out_features, bias=False),
            )
            nn.init.zeros_(module.adapter[1].weight)
        elif isinstance(module, FrozenBNBEmbedding):
            module.adapter = nn.Sequential(
                nn.Embedding(module.num_embeddings, adapter_dim),
                nn.Linear(adapter_dim, module.embedding_dim, bias=False),
            )
            nn.init.zeros_(module.adapter[1].weight)

add_adapters(gpt)
gpt.to(device)
```

Gambar 5. *Snippet Code LoRA Adapters* [18]

E. Perancangan Sistem



Gambar 6. Topologi sistem

Penelitian ini akan menggunakan *Google Colaboratory* sebagai server *Model Machine learning*, sehingga perlu dihubungkan dengan *router tester*. Oleh karena itu, akan digunakan gradio sebagai user *interface* dan flask sebagai *framework* API *Machine learning*, untuk topologi sistem dapat dilihat pada gambar 6 di atas.

API berisi *endpoint* yang dilengkapi dengan kode yang berfungsi untuk melakukan proses *inferencing* atau prediksi seperti yang telah disusun pada gambar 7 di bawah ini.

Run the API

```
#@title Run the API
#@markdown Wait until ngrok link appear
from flask import Flask, request, jsonify
from datetime import datetime as dt
import re
from flask_ngrok import run_with_ngrok

app = Flask(__name__)

run_with_ngrok(app)

def generate_text(input):
  print(f"[{dt.now()}] Processing Input: `{input}`")
  gpt.eval()
  prompt = f"prompt:{input},commands:["
  with torch.no_grad():
    preprocessed_prompt = tokenizer(prompt, truncation=True, padding=True, max_length=256,
    preprocessed_prompt = {key: value.to(device) for key, value in preprocessed_prompt.ite
    out = gpt.generate(**preprocessed_prompt, max_length=128, top_k=50, top_p=0.9, tempera
    outputs = tokenizer.decode(out[0]).split(",")
    full_commands = outputs[1].replace('\n',',')

  return full_commands

@app.route('/predict', methods=['POST'])
def predict():
  input = request.json['input']
  full_commands = generate_text(input)

  print(f"[{dt.now()}] Processing Output: `{full_commands}`")
  regex = re.search(r"\[(.+)\] (.+) \[eoc]",full_commands)
  router = regex[1]
  commands = regex[2]
  logs = []

  commands = {"commands": commands.split(",")}

  return jsonify(commands)

if __name__ == '__main__':
    app.run()
```

Gambar 7. *Snippet Code* API

User *interface* dibangun menggunakan gradio dan sistem *backend* yang berfungsi untuk mengirimkan input bahasa natural menggunakan kode *HTTP requests* pada gambar 8 ke API *Machine learning* dan menerima output berupa *commands router*.

```
payload = json.dumps(
    {
    "input": f"{input_text}"
    }
)
headers = {
    'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

result = response.json()
```

Gambar 8. *Snippet Code HTTP Request*

Kemudian mengirimkan *commands* menggunakan *library netmiko* dengan kode yang terdapat pada gambar 9.

F. Pengujian *Router* Dummy

Setelah sistem dibuat, akan dilakukan pengujian dengan 3 tipe *prompt* yang berbeda

- *Prompt* Pergantian *IP*

- *Prompt* Pembuatan *Static Routing*
- *Prompt* Pembuatan *VLAN*

```
with ConnectHandler(**device) as net_connect:
# Send the commands
    command = "\n".join(result['commands'])
    logs.append(f"[{dt.now()}] Sending \n{command} \nto {device['ip']}")
    net_connect.send_config_set(result['commands'])

router_log = parse_log()
```

Gambar 9. Snippet Kode Netmiko

G. Cara Analisa Hasil Pengujian

Pengujian dilakukan menggunakan 60 sampel *inferencing* dan dihitung menggunakan rumus *Similarity* pada rumus nomor (1) [19].

$$S = \left[\frac{Nmatch}{Nmax}\right] x\ 100\% \quad (1)$$

Dan kemudian diambil rata-rata menggunakan rumus (2).

$$AvgS(n) = \frac{sum(S(n))}{n} \quad (2)$$

IV. HASIL DAN PEMBAHASAN

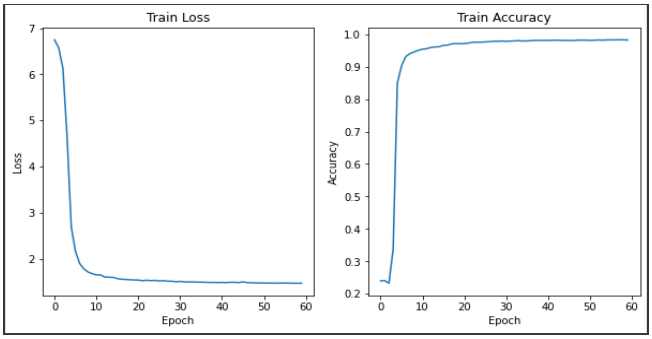
A. Hasil Pelatihan *Model*

Melalui pelatihan yang dilakukan selama 1 jam 6 menit 19 detik dengan total *step* sebanyak 1080 *step*, Didapatkan beberapa nilai matriks seperti tabel di bawah ini.

Tabel 3. Metriks Pelatihan *Model*

Metriks	Nilai
Waktu *Training*	1:06:19
Total *Step*	1080
Waktu/*Step*	3.14 Detik
Epoch Terakhir	60
Loss Terakhir	1.467
Accuracy Terakhir	0.982

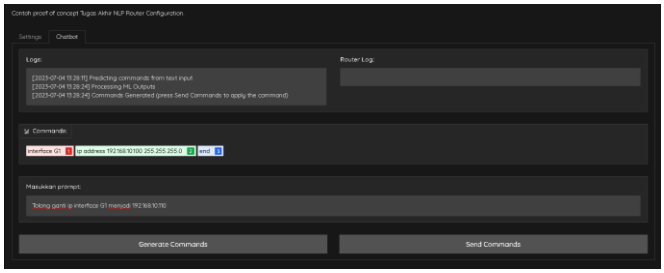
Melalui grafik pada gambar 10, dapat dianalisa bahwa nilai *loss* dan *accuracy* mulai menyentuh perbandingan 90% dari nilai awal pada *step* 30, dan perubahan *accuracy* dan *loss* sudah mulai stagnan pada *step* 30 ke atas. Dari grafik pada gambar 10 bisa ditarik analisa bahwa *training* yang efektif adalah di dalam *range step* 1-30.



Gambar 10. Grafik Pelatihan *Model*

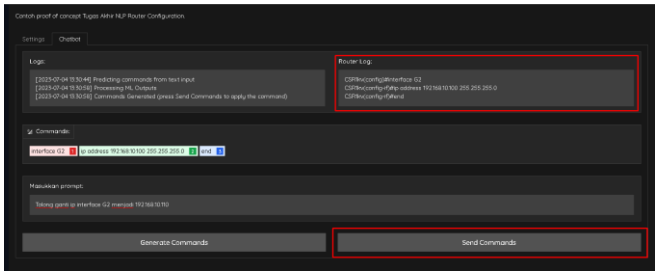
B. Hasil Perancangan Sistem

Berikut adalah hasil dari perancangan sistem, *User Interface* dapat digunakan di dalam mesin *local*. Gambar 11 menampilkan hasil *generate command*, *command* berhasil didapatkan sesuai dengan input *prompt* yang diberikan.



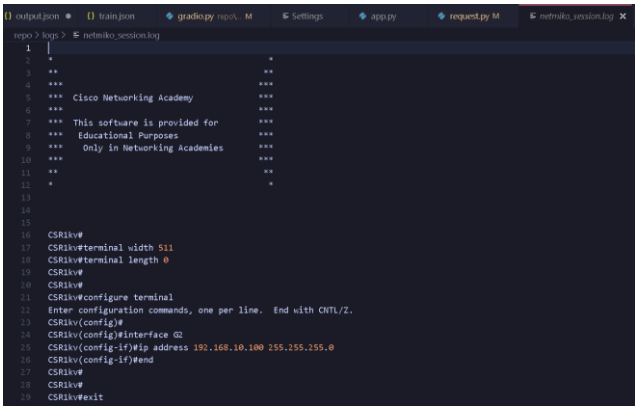
Gambar 11. *Generate Command* pada *User Interface*

Selanjutnya, pada gambar 12 fungsi *send commands* juga berhasil digunakan. *Command* yang degenerate diaplikasikan ke *router* dengan menggunakan *script* Netmiko. Dengan SSH netmiko mengirimkan *command* dan menjalankannya di *router* yang dituju.



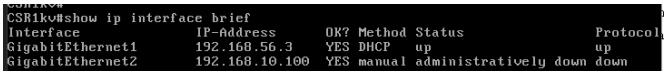
Gambar 12. *Send Command* pada *User Interface*

Pada gambar 13 dapat dilihat *log* dari fungsi *send commands* di atas. File *log* yang berada di folder user *interface* akan menyimpan *log* dari netmiko.



Gambar 13. *Logs* dari Netmiko

Hasil juga dapat dilihat pada gambar 14 melalui *router* dengan menggunakan *show ip interface brief*. Tertampil ip address G2 berhasil diubah dari *None* menjadi 192.168.10.100 dengan subnet mask 255.255.255.0



Gambar 14. Hasil Pergantian *IP* pada *Router*

C. Hasil dan Analisa Pengujian

Dari 20 pengujian per sub (total 60), diambil 2 contoh hasil pengujian.

Tabel 4. *Prompt* Pergantian *IP*

Prompt
tolong ganti ip interface F0/3 Router G menjadi 192.168.10.1 255.255.255.0
Generated command
[Router G] configure terminal, interface F0/3, ip address 192.168.10.1 255.255.255.0, end [eoc]<|endoftext|>
Correct command
[Router G] configure terminal, interface F0/3, ip address 192.168.10.1 255.255.255.0, end [eoc]<|endoftext|>
Nilai Similarity
100.00%
Prompt
tolong ganti ip *interface* F0/0/1 *Router* X menjadi 172.16.254.1 255.255.168.0
Generated command
[RouterX] configure terminal, interface F0/0/1, ip address 172.16.252.1
Correct command
[Router X] configure terminal, interface F0/0/1, ip address 172.16.254.1
Nilai Similarity
95.89%

Tabel 5. *Prompt* Pembuatan *Static Routing*

Prompt
Konfigurasikan routing statis di Router A untuk mengarahkan lalu lintas ke jaringan 172.16.254.0 dengan subnet mask 255.255.255.0 melalui interface G1
Generated command
[Router A] configure terminal, ip route 172.16.24.0 255.255.255.0 g1, end [eoc]<|endoftext|>
Correct command
[Router A] configure terminal, ip route 172.16.254.0 255.255.255.0 G1, end [eoc]<|endoftext|>
Nilai Similarity
98.40%
Prompt
Konfigurasikan routing statis di Router A untuk mengarahkan lalu lintas ke jaringan 192.168.100.0 dengan subnet mask 255.255.255.0 melalui interface G1
Generated command
[Router A] configure terminal, ip route 192.168.100.0 255.255.255.0 G1, end [eoc]<|endoftext|>
Correct command
[Router A] configure terminal, ip route 192.168.100.0 255.255.255.0 G1, end [eoc]<|endoftext|>
Nilai Similarity
100.00%

Tabel 6. *Prompt* Pembuatan *VLAN*

Prompt
Aktifkan VLAN 17 pada interface G0/2 di Router G dan berikan IP address 192.168.10.0 dengan subnet 255.255.255.0
Generated command
[Router G] configure terminal, int g0/2.17, encapsulation dot1q 19, ip address 192.168.10.0 255.255.255.0, end [eoc]<|endoftext|>
Correct command
[Router G] configure terminal, int G0/2.17, encapsulation dot1q 17, ip address 192.168.10.0 255.255.255.0, end [eoc]<|endoftext|>
Nilai Similarity
98.46%
Prompt
Aktifkan VLAN 23 pada interface G1 di Router A dan berikan IP address 192.168.10.0 dengan subnet 255.255.255.0
Generated command
[Router A] configure terminal, int G1.23, encapsulation dot1q 23, ip address 192.168.10.0 255.255.255.0, end [eoc]<|endoftext|>
Correct command
[Router A] configure terminal, int G1.23, encapsulation dot1q 23, ip address 192.168.10.0 255.255.255.0, end [eoc]<|endoftext|>
Nilai Similarity
100%

Tabel 7. Rata-Rata Hasil Pengujian

Tujuan	Rata Rata Similarity Sampel
Mengganti IP satu *Interface Router*	92.02%
Menambahkan Routing Static pada *Router*	97%
Menambahkan VLAN pada *router*	95%

Dari 60 sampel testing didapatkan 7 sampel memiliki nilai akurasi di bawah 90%, jika dilihat dari pola *generated command* pada tabel 4, 5, dan 6 terdapat *command* yang hilang atau tidak sesuai. Hal ini bisa terjadi dikarenakan terjadi *overfitting* yang disebabkan oleh dataset yang kurang merata. Selain itu, NLP memerlukan banyak memori untuk menyimpan konteks yang terjadi, *model* tidak bisa menangkap semua konteks yang diberikan juga bisa menjadi salah satu penyebabnya.

Hal ini bisa diatasi dengan cara melakukan *generate command* ulang sampai 2-3x. Jika melihat dari hasil yang diberikan, kesalahan generate fatal terjadi sekitar 10% dari keseluruhan sampel. Akan tetapi, hal ini tidak praktikal, sehingga untuk skenario sebenarnya tidak direkomendasikan.

V. SIMPULAN

Melalui penelitian ini, ditarik beberapa kesimpulan seperti di bawah ini:

- Sebuah *model Machine learning* bisa digunakan untuk membantu konfigurasi *router*, dengan proses Finetuning menggunakan dataset yang berisi konfigurasi *router* dan melabelinya dengan bahasa natural akan membuat *model* mengerti pola dari bahasa natural tersebut dan memberikan output berupa *command* yang diminta.
- Awalnya *model* GPT-J 6B tidak bisa di-load menggunakan Google Colaboratory dengan cara standar karena ketidak cukupan *resource*, akan tetapi dengan menggunakan teknik 8bit Quantization *model* akan bisa diload. Dan dengan teknik Low Rank Adaptation, proses Finetuning bisa dilakukan secara efektif.
- User bisa mengakses *model* dengan mudah menggunakan User *Interface* yang telah terhubung dengan API *Machine learning* yang berada diserver colab notebook, hal ini memungkinkan user tidak perlu menggunakan *resource* yang banyak untuk mengakses *model* karena tidak diperlukan loading *model*.
- Dengan percobaan training sebanyak 141 dataset, yang ditraining sebanyak 60 epoch (1080 *step*s) akan memakan waktu 1 jam 6 menit 19 detik. percobaan training ini mendapatkan akurasi yang sangat akurat yaitu sebesar 98%. Dengan menambahkan jumlah dataset, variasi data dan jumlah waktu training, training akan jauh lebih efektif.

REFERENSI

[1] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford, "The cutting EDGE of IP router configuration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 21–26, Jan. 2004, doi: 10.1145/972374.972379.

[2] A. -F. Sicoe, R. Botez, I. -A. Ivanciu and V. Dobrota, "Fully Automated Testbed of Cisco Virtual *Router*s in Cloud Based Environments," 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). Sofia, Bulgaria, pp. 49-53, 2022.

[3] C. Banse and S. Rangarajan, "A Secure Northbound *Interface* for SDN Applications," 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015, pp. 834-839

[4] OpenAI, ChatGPT [Large language *model*], 2023 https://chat.openai.com

[5] HuggingFace, Open-*source large language model*s leaderboard, 2023 https://huggingface.co/spaces/ HuggingFaceH4/open llm leaderboard

[6] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," in IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 625-655, 2023.

[7] Dugaev, Dmitrii & Matveev, Ivan & Siemens, Eduard & Shuvalov, Viatcheslav, Adaptive Reinforcement Learning-Based Routing Protocol for Wireless Multihop Networks. 209-218, 2018, 10.1109/APEIE.2018.8545412.f.

[8] A. Kattepur, S. David and S. K. Mohalik, "*Model*-based reinforcement learning for *router* port queue configurations," in Intelligent and Converged Networks, vol. 2, no. 3, pp. 177-197, 2021, doi: 10.23919/ICN.2021.0016.

[9] Bin Dai, Yuanyuan Cao, Zhongli Wu, Zhewei Dai, Ruyi Yao, Yang Xu, *Routing optimization meets Machine Intelligence: A perspective for the future network*. *Neurocomputing. Volume* 459, 2021.

[10] Houidi, Z. B., & Rossi, D, Neural language *model*s for network configuration: Opportunities and reality check. Computer Communications, 193, 118-125, 2022.

[11] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W, *Lora: Low-rank adaptation of large language models*. arXiv preprint arXiv:2106.09685, 2021.

[12] Wang, B., & Komatsuzaki, A, *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*, *2021*, Diakses melalui https://github.com/kingoflolz/mesh-transformer-jax

[13] C. Wijaya, "Simulasi Pemanfaatan Dynamic Routing Protocol OSPF Pada Router Di Jaringan Komputer Unpar," Research Report - Engineering Science;Vol.1 2011, 2011

[14] Khurana, Diksha & Koli, Aditya & Khatter, Kiran & Singh, Sukhdev, *Natural Language Processing: State of The Art, Current Trends and Challenges. Multimedia Tools and Applications*, 2022, 82. 10.1007/s11042-022-13428-4.

[15] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R, A survey of large language models. arXiv, 2023. preprint arXiv:2303.18223.

[16] Radford, A., & Narasimhan, K, Improving Language Understanding by Generative Pre-Training, 2018.

[17] Belkada Y. Dettmers T, *A Gentle Introduction to 8-bit Matrix Multiplication for transformers at scale using Hugging Face Transformers, Accelerate and bitsandbytes*, 2022, https://huggingface.co/b*log*/hf-bitsandbytes-integration.

[18] Coral G, *Fine-tuning 6-Billion GPT-J (& other models) in colab with LoRA and 8-bit compression*, 2022, https://github.com/gustavecortal/gpt-j-fine-tuning-example.

[19] John W. Ratcliff and David E. Metzener, Pattern Matching: The Gestalt Approach, Dr. Dobb's Journal, 1988.