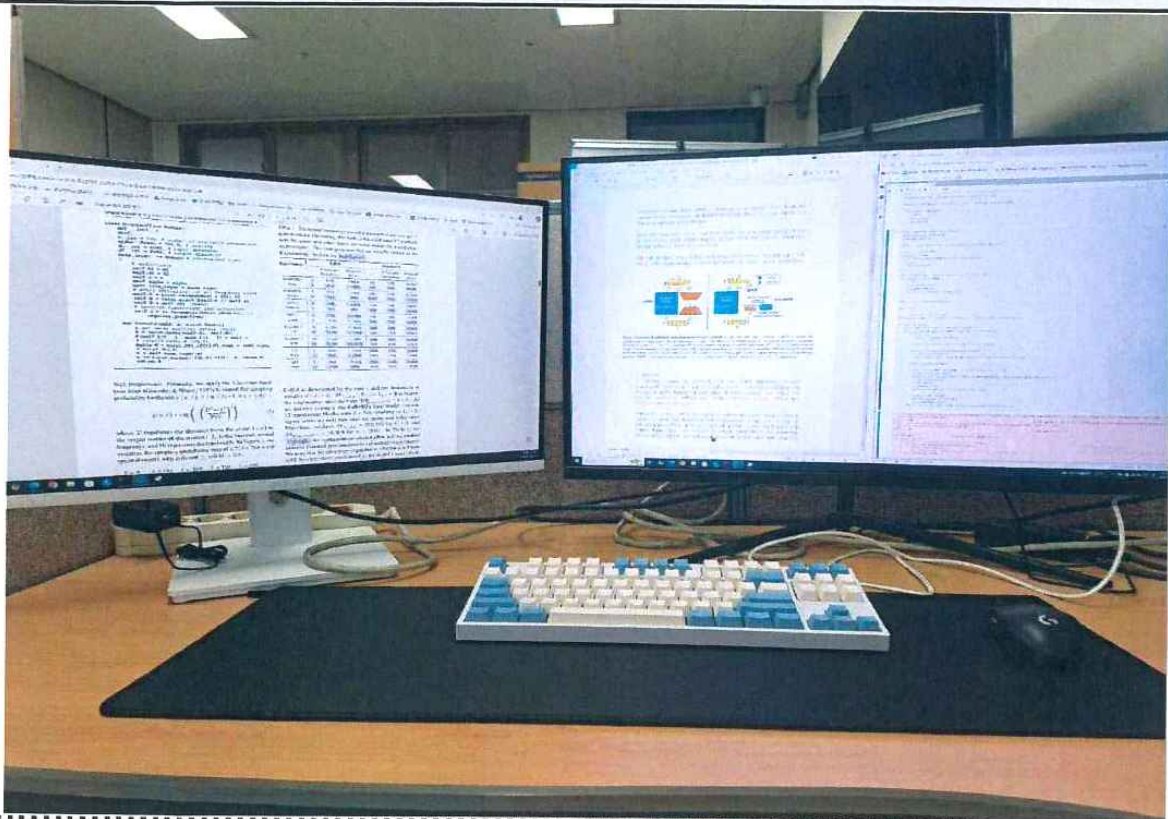


학부생 연구실 인턴 활동보고서

인적사항	소속학과	수학과	성명	김연범		
	학번	202000826	학년	4학년		
전체 활동 기간	2025. 3. 1. ~ 2025. 8. 31 (6개월)					
해당월 활동 기간	2025. 6. 1. ~ 2025. 6. 30 (1개월)					
활동 내용	<ul style="list-style-type: none"> - FourierFT 논문 탐독 및 정리 (Parameter-Efficient Fine-Tuning with Discrete Fourier Transform) - FourierFT 기반 llama3 LLM Fine-tuning 진행 : 논문 기반의 FourierFT 적용 방법 이해 및 정리 : python을 활용한 FourierFT 함수 구현 및 적용					
활동 성과 및 소감	a) 활동 성과 <ul style="list-style-type: none"> - FourierFT 방법에 대한 이해 달성 <ol style="list-style-type: none"> 1) 전 layer에 공유할 spectral entry matrix $E(2 \times n)$를 랜덤하게 초기화, frozen $\rightarrow E$는 각 layer에서 어떤 위치(=주파수)를 사용할지 나타내는 mask or index table임 \rightarrow 각 layer 별 학습 대상인 spectral coefficients는 Gaussian 분포를 따라 초기화 진행 2) 각 layer 별 E의 인덱스에 해당하는 부분의 주파수 값을 가져오고, 나머지는 0으로 채운 Dense Spectral Matrix F 설정 $\rightarrow F$에 역이산푸리에 변환(IDFT)을 적용해 ΔW 구함 3) ΔW를 기존 pre-trained weight에 덧셈 \rightarrow 구해진 LOSS 기반으로 Gradient Descent 진행 $\rightarrow n \times (L)$ 파라미터 업데이트 - Python기반 FourierFT 함수 구현 및 Fine-tuning 진행 방법 이해달성 <ol style="list-style-type: none"> 1) 기존 Full Fine-tuning과 달리 모델의 모든 파라미터를 업데이트하지 않고, 사전 지정된 주파수 좌표에서 희소한 복소수 계수만을 학습 2) Fourier 스펙트럼의 학습 가능한 위치는 임의로 선택된 n개의 2D 좌표로 구성됨. 나머지 스펙트럼은 0으로 유지 3) IDFT를 통해 복원된 ΔW는 기존 pre-trained 모델의 특정 Linear Layer의 weight에 직접 더해짐 b) 소감 pre-trained LLM에 FourierFT를 활용한 파인튜닝을 적용해 LLM을 내가 직접 원하는 모델로 down-stream task를 수행한 것이 인상적이었다.					
평가	성실성	업무능력	활용효과			
	상 . 중 . 하	상 . 중 . 하	상 . 중 . 하			
* 활동사진 첨부 <div style="text-align: center;">2025 년 6 월 18일</div> <div style="text-align: right;"> 보고자 : 김연범 김연범 담당교수(지도교수) : 김정삼(인) </div>						
충남대학교 소프트웨어중심대학사업단장 귀하						

활동사진(2장)



```

m_lm.py - JupyterLab
127.0.0.1:8888/lab/tree/manne-lm/m_lm.py
YouTube NAVER 네이버대학교 포털 네이버대학교 컴퓨터... 네이버대학교 수학과 Google Drive 한국과학기술인 Gmail Baekjoon Online Ju...
File Edit View Run Kernel Help
m_lm.py Markdown Python 3 (ipykernel)

def print_trainable_parameters(model):
    total = sum(p.numel() for p in model.parameters())
    trainable = sum(p.numel() for p in model.parameters() if p.requires_grad)
    print(f"총 파라미터 수: {total}")
    print(f"학습 가능한 파라미터 수: trainable")
    print(f"파라미터 효율률 비율: 100 * trainable / total, %.1f%%" % (trainable / total))

def apply_fourier_to_model(
    model,
    target_modules=["q_proj", "v_proj"],
    n_heads,
    alpha=0.5
):
    print_trainable_parameters(model)
    applied_fourier(layer for target_modules in q_proj, v_proj)
    결과 파라미터 수: 6,620,277,632
    학습 가능한 파라미터 수: 10,184
    파라미터 효율률 비율: 0.000154

from torch.utils.data import Dataset
import json

class LstmDataset(Dataset):
    def __init__(self, file_path, tokenizer, max_length=2048):
        self.samples = []
        with open(file_path, 'r', encoding='utf-8') as f:
            for line in f:
                obj = json.loads(line)
                prompt = obj.get("prompt", "")
                completion = obj.get("completion", "")
                text = prompt + completion
                tokenized = tokenizer(text, truncation=True, max_length=max_length, padding='max_length', return_tensors='pt')
                self.samples.append(tokenized)

    def __len__(self):
        return len(self.samples)

    def __getitem__(self, idx):
        item = self.samples[idx]
        labels = item["input_ids"].clone()
        return item

from torch.utils.data import DataLoader

tokenizer.pad_token = tokenizer.eos_token # padding 토큰 설정
dataset = LstmDataset("data/train_data.jsonl", tokenizer, max_length=2048)

train_loader = DataLoader(
    dataset,
    batch_size=1,
    shuffle=True
)

from transformers import get_scheduler
from torch.optim import AdamW
from tqdm import tqdm

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# fourier의 파라미터 설정
optimizer = AdamW(
    [p for p in model.parameters() if p.requires_grad],
    lr=2e-5,
    weight_decay=0.1
)

num_training_steps = len(train_loader) * 2 # 2 epochs
lr_scheduler = get_scheduler(
    "linear",
    optimizer=optimizer,
    num_warmup_steps=100,

```