

## • 모델 로드 및 설정

```

33 def load_model(
34     self, model_id, model_config, quantize_config: Optional[QuantizeArgs] = None
35 ):
36     device = self.get_device()
37     model = AutoModelForCausalLM.from_pretrained(
38         model_id,
39         torch_dtype=model_config.torch_dtype,
40         low_cpu_mem_usage=True,
41         use_cache=False,
42         device_map=0,
43         max_length=model_config.max_length,
44         quantization_config=quantize_config,
45     )
46     model.gradient_checkpointing_enable()
47     model.enable_input_require_grads()
48     model.is_parallelizable = False
49     model.model_parallel = False
50     if quantize_config is not None:
51         model = prepare_model_for_kbit_training(model)
52     return model
53
54 def get_lora_target(self, lora_target_config):
55     return lora_target_config.targets
56
57 def get_lora_config(self, lora_config, lora_targets: List):
58     lora_config = LoraConfig(
59         task_type=TaskType.CAUSAL_LM,
60         inference_mode=False,
61         r=lora_config.r,
62         lora_alpha=lora_config.alpha,
63         lora_dropout=lora_config.dropout,
64         target_modules=lora_targets,
65         bias="none",
66     )
67     return lora_config

```

모델을 로드하고 양자화

메모리 최적화

LoRA 설정 정의

## • Custom 모델 학습

```

90 def get_train_args(self, train_args):
91     args = TrainingArguments(
92         output_dir=train_args.output_dir,
93         run_name=train_args.run_name,
94         logging_steps=train_args.logging_steps,
95         num_train_epochs=train_args.num_train_epochs,
96         per_device_train_batch_size=train_args.per_device_train_batch_size,
97         per_device_eval_batch_size=train_args.per_device_eval_batch_size,
98         gradient_accumulation_steps=train_args.gradient_accumulation_steps,
99         dataloader_num_workers=train_args.dataloader_num_workers,
100         learning_rate=train_args.learning_rate,
101         weight_decay=train_args.weight_decay,
102         warmup_ratio=train_args.warmup_ratio,
103         lr_scheduler_type=train_args.lr_scheduler_type,
104         save_steps=train_args.save_steps,
105         eval_steps=train_args.eval_steps,
106         fp16=train_args.fp16,
107         eval_strategy=train_args.eval_strategy,
108         remove_unused_columns=train_args.remove_unused_columns,
109         report_to=train_args.report_to,
110     )
111     return args
112
113 def get_trainer(
114     self,
115     model,
116     tokenizer,
117     train_args,
118     train_dataset,
119     eval_dataset,
120     callbacks,
121     data_collator,
122 ):
123     trainer = Trainer(
124         model=model,
125         tokenizer=tokenizer,
126         args=train_args,
127         train_dataset=train_dataset,
128         eval_dataset=eval_dataset,
129         callbacks=callbacks,
130         data_collator=data_collator,
131     )
132     return trainer

```

훈련에 사용할 파라미터 정의

메서드를 호출해 학습

- 학습된 모델 저장 및 배포

```
134     def train(self, trainer):
135         trainer.train()  학습 완료된 모델 경로에 저장
136
137     def save_model(self, model, save_path):
138         model.save_pretrained(save_path, from_pt=True)
139
140     def model_to_huggingface(self, model, model_id):
141         model.push_to_hub(model_id)
142
143     def tokenizer_to_huggingface(self, tokenizer, model_id):
144         tokenizer.push_to_hub(model_id)
```

Huggingface Hub에 모델과 토큰라이저를 업로드 하고 경로에 배포  
다른 사용자도 Hub에서 Custom한 모델을 다운로드해서 사용 할 수 있음