assignment_2

October 24, 2025

[ ]:
```python
# === IMPORTS AND SETUP ===
import torch
from transformers import (
    AutoTokenizer, AutoModelForCausalLM,
    DataCollatorForLanguageModeling,
    TrainingArguments, Trainer, pipeline,
    EarlyStoppingCallback
)
from ray import tune
from datasets import load_dataset
import pandas as pd
import matplotlib.pyplot as plt

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

[ ]:
```python
# === DATA AND MODEL LOADING ===
model_id = "HuggingFaceTB/SmolLM2-135M"
# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
# Load dataset
raw_datasets = load_dataset("ag_news")
train_dataset = raw_datasets["train"]
eval_dataset  = raw_datasets["test"]
# Tokenize datasets
def tokenize_function(batch):
    return tokenizer(batch["text"], truncation=True)

tokenized_train = train_dataset.map(tokenize_function, batched=True,␣
 ↪remove_columns=["text"])
tokenized_eval  = eval_dataset.map(tokenize_function, batched=True,␣
 ↪remove_columns=["text"])
# Set up data collator for language modeling
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)
```

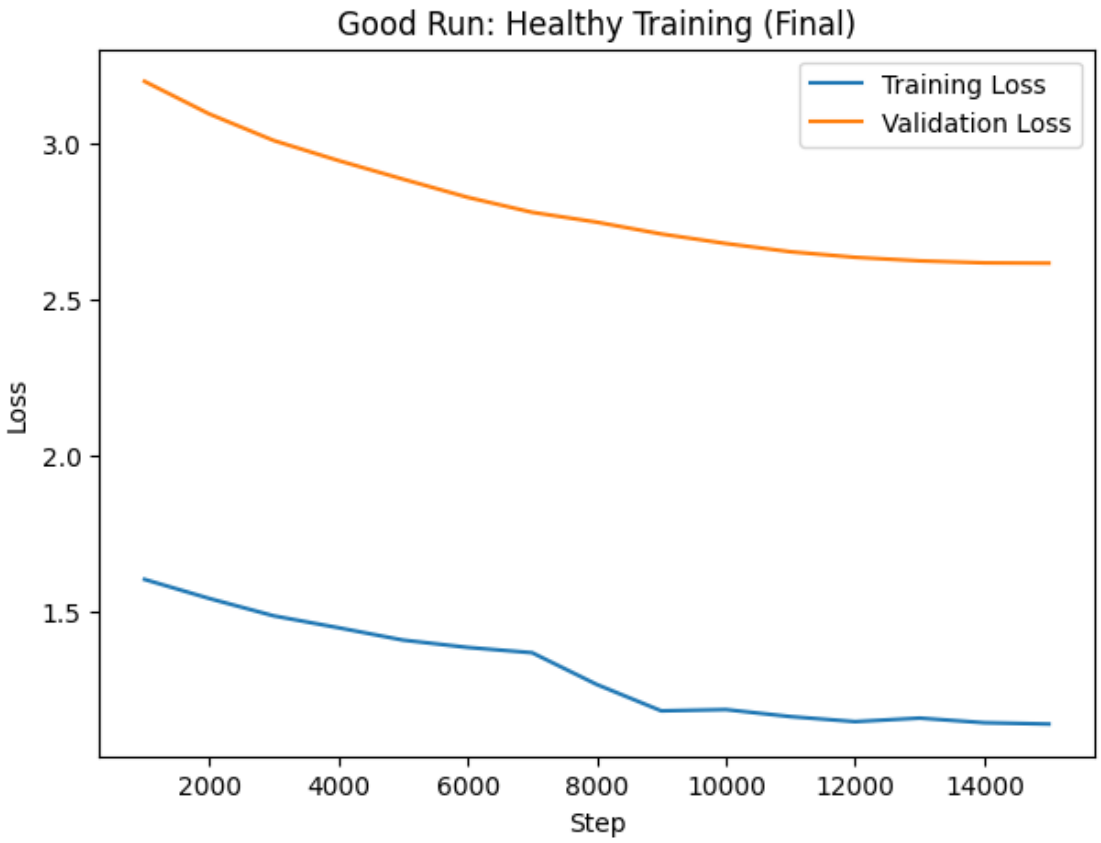
[ ]:
```python
# === HELPER FUNCTIONS (PLOTTING AND GENERATION) ===
def plot_loss(trainer, title):
```

```
    """Plots training and validation loss curves from trainer history."""
    logs = trainer.state.log_history
    df = pd.DataFrame(logs)

    # Separate train and eval loss
    train_loss = df.dropna(subset=["loss"])
    eval_loss = df.dropna(subset=["eval_loss"])

    # Plot curves
    plt.figure(figsize=(7,5))
    plt.plot(train_loss["step"], train_loss["loss"], label="Training Loss")
    plt.plot(eval_loss["step"], eval_loss["eval_loss"], label="Validation Loss")
    plt.xlabel("Step")
    plt.ylabel("Loss")
    plt.title(title)
    plt.legend()
    plt.show()
```

[ ]:
```
def show_output2(trainer, title, prompts, max_new_tokens=40):
    """Generates text from a simple prompt."""
    # Create a text generation pipeline
    gen_pipe = pipeline(
        "text-generation",
        model=trainer.model,
        tokenizer=tokenizer,
        device=0 if device=="cuda" else -1
    )
    print(f"\n=== {title} ===")
    # Generate and print text
    for prompt in prompts:
        generated = gen_pipe(prompt,␣
 ↪max_new_tokens=max_new_tokens)[0]["generated_text"]
        print(f"Prompt: {prompt}\nGenerated: {generated}\n{'-'*40}")
```

[ ]:
```
# === "GOOD RUN" (HEALTHY TRAINING) ===
# Define TrainingArguments for the final healthy run
final_args = TrainingArguments(
    output_dir="good-run-final",
    per_device_train_batch_size=8,
    learning_rate= 2e-4,     # A reasonable learning_rate
    num_train_epochs= 2,     # A reasonable num_train_epochs
    weight_decay= 0.01,      # Regularization a reasonable weight_decay
    lr_scheduler_type="cosine",

    eval_strategy="steps",
    eval_steps=1000,
    save_strategy="steps",
```

```
    save_steps=1000,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",

    logging_steps=1000,
    save_total_limit=2,
    report_to="none"
)

# Use Early Stopping to prevent overfitting
early_stop_callback = EarlyStoppingCallback(
    early_stopping_patience=1,
    early_stopping_threshold=0.0
)

# Initialize Trainer for the "Good Run"
final_good_trainer = Trainer(
    model=AutoModelForCausalLM.from_pretrained(model_id).to(device),
    args=final_args,
    train_dataset=tokenized_train,       #
    eval_dataset=tokenized_eval,
    processing_class=tokenizer,
    data_collator=data_collator,
    callbacks=[early_stop_callback]
)

# Start the final training
final_good_trainer.train()

# === SHOW RESULTS FOR "GOOD RUN" ===
# Plot the loss curve for the healthy run
plot_loss(final_good_trainer, "Good Run: Healthy Training (Final)")
# Show a sample generation from the healthy run
show_output(final_good_trainer, "Good Run Output (Final)")
```




```
Device set to use cuda:0

=== Good Run Output (Final) ===
Wall Street down on rising oil prices, weak earnings Wall Street lost ground
Wednesday, led by oil prices, as investors took profits to avoid the possibility
of higher
```

[ ]:
```
prompts = [
    "Q1",
    "Wall",
    "Google"
]
```

```
show_output2(final_good_trainer, "Text Generation Examples", prompts)

Device set to use cuda:0
```

```
=== Text Generation Examples ===
Prompt: Q1
Generated: Q14.7 million people without Internet connection in Zimbabwe, says UN
Security Council (AFP) AFP - More than 14 million people in Zimbabwe are without
Internet access, the UN Security Council reported
----------------------------------------
Prompt: Wall
Generated: Wall Street #39;s dollar tumbles on US jobs report NEW YORK (CBS.MW)
-- US economic growth is slowing, but the US dollar has seen gains in recent
trading.
----------------------------------------
Prompt: Google
Generated: Google, Google Search Together to Sell 74 Million Songs Google Inc.
and Apple Computer Inc. have announced that they will jointly sell 74 million
songs over the next 18 months,
----------------------------------------
```

[ ]:
```python
# === "BAD RUN" ===
model = AutoModelForCausalLM.from_pretrained(model_id).to(device)
bad_args = TrainingArguments(
    output_dir="bad-run",
    per_device_train_batch_size=8,
    learning_rate=4e-3,       #Too large learning_rate
    num_train_epochs=4,        # Too many num_train_epochs
    weight_decay=0.0,          # No regularization
    lr_scheduler_type="constant",
    eval_strategy="steps",
    eval_steps=1000,
    logging_steps=1000,
    save_strategy="no",
    report_to="none"
)
bad_trainer = Trainer(
    model=model,
    args=bad_args,
    data_collator=data_collator,
    train_dataset=tokenized_train,
    eval_dataset=tokenized_eval
)
bad_trainer.train()
# === SHOW RESULTS FOR "BAD RUN" ===
plot_loss(bad_trainer, "Bad Run: Oscillation")
```

```
show_output(bad_trainer, "Bad Run Output")
```

[ ]:
```
show_output2(bad_trainer, "Text Generation Examples", prompts)
```