Job 2584772 - Llama 3.1 8B Training (LoRA)

Informazioni Generali

- **Job ID**: 2584772
- **Nome**: llama_te
- **Utente**: ediluzio
- **Status**: PENDING (in attesa risorse)
- **Partizione**: boost_usr_prod (GPU > 24GB)
- **Motivo Attesa**: (Resources) - Nessuna GPU > 24GB disponibile
- **Ambiente**: PyTorch 2.7.0+cu118 (riparato)

Descrizione Tecnica

Obiettivo del Job

Fine-tuning di Llama 3.1 8B con LoRA (Low-Rank Adaptation) per SVG captioning, continuando il training dal checkpoint esistente con ambiente PyTorch riparato.

Architettura del Modello

Llama 3.1 8B Base Model

```
Architecture: Transformer Decoder-Only
- Layers: 32 transformer blocks
- Hidden Size: 4096
- Intermediate Size: 14336 (SwiGLU activation)
- Attention Heads: 32
- Key-Value Heads: 8 (GQA - Grouped Query Attention)
- Vocabulary Size: 128,256 tokens
- Context Length: 131,072 tokens (128K context)
- Total Parameters: 8.03B
```

LoRA Configuration

```
LoRA_CONFIG = {
    "r": 64,                    # Rank of adaptation
    "lora_alpha": 128,          # LoRA scaling parameter
    "target_modules": [
        "q_proj", "k_proj", "v_proj", "o_proj",  # Attention projections
        "gate_proj", "up_proj", "down_proj"       # MLP projections
```

```
    ],
    "lora_dropout": 0.1,        # Dropout for LoRA layers
    "bias": "none",             # No bias in LoRA layers
    "task_type": "CAUSAL_LM"    # Causal language modeling
}

# Trainable Parameters Calculation
# Base model: 8.03B parameters (frozen)
# LoRA adapters: ~67M parameters (trainable)
# Percentage trainable: 0.83%
```

Stack Tecnologico Riparato

Environment Stack

- **PyTorch**: 2.7.0+cu118 (risolve CVE-2025-32434)
- **Transformers**: 4.52.3 (compatibile con PyTorch 2.7.0)
- **PEFT**: 0.15.1 (Parameter Efficient Fine-Tuning)
- **Accelerate**: 1.7.0 (multi-GPU training)
- **BitsAndBytes**: 0.41.3 (quantization support)
- **CUDA**: 11.8 (sistema compatibility)

Risoluzione Problemi Precedenti

```
# Problema risolto: torch.load weights_only=True
# PyTorch 2.7.0 supporta weights_only=True senza errori CVE
torch.load(checkpoint_path, weights_only=True)  # ✅ Funziona

# Compatibilità CUDA verificata
torch.cuda.is_available()  # True
torch.version.cuda         # '11.8'
```

Training Configuration

Hyperparameters

```
TRAINING_ARGS = {
    # Learning Rate & Optimization
    "learning_rate": 2e-4,
    "lr_scheduler_type": "cosine",
    "warmup_ratio": 0.03,
    "weight_decay": 0.01,
    "adam_beta1": 0.9,
    "adam_beta2": 0.95,
    "adam_epsilon": 1e-8,
```

```
    # Training Dynamics
    "num_train_epochs": 3,
    "per_device_train_batch_size": 2,
    "per_device_eval_batch_size": 4,
    "gradient_accumulation_steps": 8,
    "max_grad_norm": 1.0,

    # Evaluation & Checkpointing
    "eval_strategy": "steps",
    "eval_steps": 10,
    "save_strategy": "steps",
    "save_steps": 10,
    "logging_steps": 1,
    "load_best_model_at_end": True,
    "metric_for_best_model": "eval_loss",

    # Early Stopping
    "early_stopping_patience": 5,
    "early_stopping_threshold": 0.001,

    # Memory Optimization
    "dataloader_pin_memory": True,
    "dataloader_num_workers": 4,
    "remove_unused_columns": False,
    "fp16": True,  # Mixed precision training
    "gradient_checkpointing": True
}
```

Dataset Configuration

```
DATASET_CONFIG = {
    "train_file": "data/processed/xml_format/train_set_final_xml_rgb.json",
    "eval_file": "data/processed/xml_format/test_set_final_xml_reduced_rgb.json",
    "train_samples": ~4500,  # 90% split
    "eval_samples": ~500,    # 10% split
    "max_length": 512,
    "image_size": (224, 224),
    "format": "xml_direct_input"
}
```

Data Processing Pipeline

Input Format (XML Direct)

```
<svg>
  <rect x="10" y="20" width="100" height="50" fill="blue"/>
  <text x="60" y="45">Button</text>
</svg>
```

Tokenization Process

```
# Llama 3.1 Tokenizer
tokenizer = AutoTokenizer.from_pretrained("meta-llama/Meta-Llama-3.1-8B-Instruct")
tokenizer.pad_token = tokenizer.eos_token

# Input formatting
def format_input(svg_xml, caption):
    prompt = f"<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n"
    prompt += "You are an expert at describing SVG images in detail.<|eot_id|>\n"
    prompt += f"<|start_header_id|>user<|end_header_id|>\n"
    prompt += f"Describe this SVG image:\n{svg_xml}<|eot_id|>\n"
    prompt += f"<|start_header_id|>assistant<|end_header_id|>\n{caption}<|eot_id|>"
    return prompt

# Tokenization
inputs = tokenizer(
    formatted_text,
    max_length=512,
    truncation=True,
    padding="max_length",
    return_tensors="pt"
)
```

Multi-GPU Training Strategy

Distributed Training Setup

```
# Accelerate configuration
accelerate_config = {
    "compute_environment": "LOCAL_MACHINE",
    "distributed_type": "MULTI_GPU",
    "num_processes": 2,  # 2 GPUs richieste
    "gpu_ids": [0, 1],
    "mixed_precision": "fp16",
    "gradient_accumulation_steps": 8
}

# Data parallelism
model = torch.nn.DataParallel(model, device_ids=[0, 1])
```

Memory Requirements

```
Per GPU (>24GB richiesta):
- Model (8B params): ~16GB (fp16)
- LoRA adapters: ~134MB (fp16)
- Gradients: ~134MB
- Optimizer states: ~268MB (AdamW)
```

```
- Activations: ~4-6GB (batch_size=2)
- Buffer: ~2GB
Total per GPU: ~23-25GB
```

Checkpoint Management

Resume Training

```
# Checkpoint esistente
checkpoint_path = "experiments/xml_direct_input/outputs/llama31_8b_test3_multi_gpu/che

# Verifica compatibilità
def verify_checkpoint_compatibility():
    try:
        # Test torch.load con weights_only=True (risolto in PyTorch 2.7.0)
        state_dict = torch.load(f"{checkpoint_path}/pytorch_model.bin", weights_only=T
        print("✅ Checkpoint compatible with PyTorch 2.7.0")
        return True
    except Exception as e:
        print(f"❌ Checkpoint error: {e}")
        return False

# Resume training
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    resume_from_checkpoint=checkpoint_path
)
```

Checkpoint Structure

```
checkpoint-23900/
├── pytorch_model.bin          # LoRA adapter weights
├── adapter_config.json        # LoRA configuration
├── adapter_model.bin          # PEFT adapter
├── training_args.bin          # Training arguments
├── trainer_state.json         # Training state
├── optimizer.pt               # Optimizer state
├── scheduler.pt               # LR scheduler state
└── rng_state.pth             # Random state
```

Monitoring & Logging

WandB Integration

```
wandb_config = {
    "project": "svg_captioning_llama",
    "name": f"llama31_8b_lora_resume_{timestamp}",
    "tags": ["llama", "lora", "svg", "resume", "pytorch2.7"],
    "config": {
        "model": "meta-llama/Meta-Llama-3.1-8B-Instruct",
        "lora_r": 64,
        "lora_alpha": 128,
        "learning_rate": 2e-4,
        "batch_size": 16,  # effective batch size
        "pytorch_version": "2.7.0+cu118"
    }
}
```

Metrics Tracked

- **Training Loss**: Cross-entropy loss
- **Evaluation Loss**: Validation set loss
- **Learning Rate**: Cosine schedule
- **GPU Memory**: Per-device utilization
- **Training Speed**: Samples/second
- **Gradient Norm**: For stability monitoring

Expected Training Behavior

Training Progression

```
Epoch 1: Resume from step 23900
- Initial eval_loss: ~1.2 (from previous training)
- Target eval_loss: ~0.8-1.0
- Early stopping if no improvement for 5 steps

Convergence Criteria:
- eval_loss improvement < 0.001 for 5 consecutive evaluations
- Maximum 3 epochs from resume point
- Best model selection based on lowest eval_loss
```

Performance Expectations

- **Training Speed**: ~2-3 samples/second (multi-GPU)
- **Memory Usage**: ~23GB per GPU
- **Convergence Time**: 2-4 hours (depending on early stopping)
- **Final Performance**: BLEU-4 > 0.15, CLIP Score > 0.25

Troubleshooting Risolto

Problemi Precedenti (Risolti)

1. ✅ **PyTorch Incompatibility**: Aggiornato a 2.7.0+cu118
2. ✅ **CVE-2025-32434**: torch.load weights_only=True funziona
3. ✅ **CUDA Mismatch**: PyTorch cu118 compatibile con CUDA 11.8
4. ✅ **Environment Corruption**: Ambiente ricostruito da zero

Status Attuale

- **Environment**: ✅ Completamente funzionante
- **Checkpoint**: ✅ Compatibile e verificato
- **GPU Queue**: ⏳ In attesa risorse boost_usr_prod

Questo job rappresenta la continuazione del fine-tuning Llama con ambiente completamente riparato e testato.