Job 2584773 - Gemma 2 9B IT Training (LoRA)

Informazioni Generali

- **Job ID**: 2584773
- **Nome**: gemma_te
- **Utente**: ediluzio
- **Status**: PENDING (in attesa priorità)
- **Partizione**: boost_usr_prod (GPU > 24GB)
- **Motivo Attesa**: (Priority) - Priorità inferiore rispetto ad altri job
- **Ambiente**: PyTorch 2.7.0+cu118 (riparato)

Descrizione Tecnica

Obiettivo del Job

Fine-tuning di Gemma 2 9B IT (Instruction Tuned) con LoRA per SVG captioning, continuando il training dal checkpoint esistente con ambiente PyTorch riparato.

Architettura del Modello

Gemma 2 9B IT Base Model

```
Architecture: Transformer Decoder-Only (Gemma 2 family)
- Layers: 42 transformer blocks
- Hidden Size: 3584
- Intermediate Size: 14336 (GeGLU activation)
- Attention Heads: 16
- Key-Value Heads: 8 (GQA - Grouped Query Attention)
- Vocabulary Size: 256,000 tokens
- Context Length: 8,192 tokens
- Total Parameters: 9.24B
- Instruction Tuned: Yes (IT variant)
```

Gemma 2 Innovations

```
# Key architectural improvements over Gemma 1
GEMMA2_FEATURES = {
    "sliding_window_attention": 4096,  # Local attention pattern
    "soft_capping": 50.0,              # Attention logits capping
    "query_pre_attn_scalar": 3584**-0.5,  # Query scaling
    "final_logit_softcapping": 30.0,   # Output logits capping
    "attn_logit_softcapping": 50.0,    # Attention logits capping
    "use_cache": True,                 # KV cache optimization
    "rope_theta": 10000.0              # RoPE base frequency
}
```

LoRA Configuration (Optimized for Gemma 2)

```
LORA_CONFIG = {
    "r": 64,                       # Rank of adaptation
    "lora_alpha": 128,             # LoRA scaling parameter (α/r = 2.0)
    "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
    "modules_to_save": ["embed_tokens", "lm_head"]  # Full fine-tuning for embeddings
}

# Trainable Parameters Calculation
# Base model: 9.24B parameters (frozen)
# LoRA adapters: ~75M parameters (trainable)
# Embeddings: ~896M parameters (trainable if modules_to_save)
# Total trainable: ~75M (0.81%) or ~971M (10.5%) with embeddings
```

Stack Tecnologico Riparato

Environment Stack

- **PyTorch**: 2.7.0+cu118 (risolve CVE-2025-32434)
- **Transformers**: 4.52.3 (supporto completo Gemma 2)
- **PEFT**: 0.15.1 (LoRA implementation)
- **Accelerate**: 1.7.0 (distributed training)
- **BitsAndBytes**: 0.41.3 (quantization)
- **Flash Attention**: 2.x (memory efficient attention)

Gemma 2 Specific Optimizations

```
# Attention optimizations
model_config.update({
    "use_flash_attention_2": True,     # Flash Attention 2.0
    "torch_dtype": torch.float16,      # Mixed precision
    "attn_implementation": "flash_attention_2"
})
```

```
# Memory optimizations for 9B model
torch.backends.cuda.matmul.allow_tf32 = True
torch.backends.cudnn.allow_tf32 = True
```

Training Configuration

Hyperparameters (Gemma 2 Optimized)

```
TRAINING_ARGS = {
    # Learning Rate & Optimization (Lower LR for larger model)
    "learning_rate": 1e-4,       # Reduced for 9B model stability
    "lr_scheduler_type": "cosine_with_restarts",
    "warmup_ratio": 0.05,        # Longer warmup for stability
    "weight_decay": 0.01,
    "adam_beta1": 0.9,
    "adam_beta2": 0.95,
    "adam_epsilon": 1e-8,

    # Training Dynamics
    "num_train_epochs": 2,       # Fewer epochs for larger model
    "per_device_train_batch_size": 1,  # Smaller batch for 9B model
    "per_device_eval_batch_size": 2,
    "gradient_accumulation_steps": 16,  # Higher accumulation
    "max_grad_norm": 0.5,        # Lower gradient clipping

    # 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 (More conservative)
    "early_stopping_patience": 10,
    "early_stopping_threshold": 0.0005,

    # Memory Optimization (Critical for 9B)
    "dataloader_pin_memory": True,
    "dataloader_num_workers": 2,  # Reduced for memory
    "remove_unused_columns": False,
    "fp16": True,
    "gradient_checkpointing": True,
    "optim": "adamw_torch_fused"  # Fused optimizer
}
```

Memory Management Strategy

```
# Advanced memory optimizations for 9B model
MEMORY_CONFIG = {
    "cpu_offload": False,           # Keep on GPU for speed
    "pin_memory": True,
    "empty_cache_steps": 10,        # Frequent cache clearing
    "max_memory_per_gpu": "28GB",   # Leave 4GB buffer
    "low_cpu_mem_usage": True,
    "torch_compile": True           # PyTorch 2.7 compilation
}
```

Gemma 2 Specific Features

Instruction Following Format

```
# Gemma 2 IT specific prompt format
def format_gemma_prompt(svg_xml, caption):
    prompt = "<bos><start_of_turn>user\n"
    prompt += "You are an expert at describing SVG images. "
    prompt += "Provide a detailed, accurate description of this SVG image:\n\n"
    prompt += f"{svg_xml}\n"
    prompt += "<end_of_turn>\n"
    prompt += "<start_of_turn>model\n"
    prompt += f"{caption}"
    prompt += "<end_of_turn><eos>"
    return prompt

# Special tokens for Gemma 2
SPECIAL_TOKENS = {
    "bos_token": "<bos>",
    "eos_token": "<eos>",
    "pad_token": "<pad>",
    "unk_token": "<unk>",
    "start_of_turn": "<start_of_turn>",
    "end_of_turn": "<end_of_turn>"
}
```

Attention Pattern Optimization

```
# Sliding window attention for long sequences
def configure_sliding_window():
    model.config.sliding_window = 4096
    model.config.max_position_embeddings = 8192

    # Soft capping for numerical stability
    model.config.attn_logit_softcapping = 50.0
    model.config.final_logit_softcapping = 30.0
```

Multi-GPU Training (Enhanced)

Distributed Strategy for 9B Model

```
# Enhanced distributed setup for larger model
accelerate_config = {
    "compute_environment": "LOCAL_MACHINE",
    "distributed_type": "MULTI_GPU",
    "num_processes": 2,
    "gpu_ids": [0, 1],
    "mixed_precision": "fp16",
    "gradient_accumulation_steps": 16,
    "deepspeed_config": {
        "zero_stage": 2,            # ZeRO-2 for memory efficiency
        "offload_optimizer": False,  # Keep optimizer on GPU
        "allgather_bucket_size": 5e8,
        "reduce_bucket_size": 5e8
    }
}
```

Memory Requirements (9B Model)

```
Per GPU (>24GB richiesta):
- Model (9B params): ~18GB (fp16)
- LoRA adapters: ~150MB (fp16)
- Gradients: ~150MB
- Optimizer states: ~300MB (AdamW)
- Activations: ~6-8GB (batch_size=1)
- KV Cache: ~2GB
- Buffer: ~2GB
Total per GPU: ~28-30GB (richiede GPU >24GB)
```

Checkpoint Management

Resume from Existing Checkpoint

```
# Checkpoint path (may not exist yet)
checkpoint_path = "experiments/xml_direct_input/outputs/gemma2_9b_it_test3_multi_gpu/c

# Checkpoint verification
def verify_gemma_checkpoint():
    if os.path.exists(checkpoint_path):
        try:
            # Test compatibility with PyTorch 2.7.0
            state_dict = torch.load(f"{checkpoint_path}/pytorch_model.bin", weights_on
            print("✅ Gemma checkpoint compatible")
            return True
        except Exception as e:
            print(f"❌ Checkpoint error: {e}")
            return False
    else:
        print("⚠️ Checkpoint not found, starting from base model")
        return False

# Training initialization
if verify_gemma_checkpoint():
    # Resume from checkpoint
    trainer.train(resume_from_checkpoint=checkpoint_path)
else:
    # Start fresh training
    trainer.train()
```

Gemma 2 Specific Optimizations

Flash Attention Integration

```
# Flash Attention 2.0 for memory efficiency
from flash_attn import flash_attn_func

class GemmaFlashAttention(nn.Module):
    def forward(self, query, key, value, attention_mask=None):
        # Flash attention with soft capping
        attn_output = flash_attn_func(
            query, key, value,
            dropout_p=0.0,
            softmax_scale=None,
            causal=True,
            window_size=(-1, -1),  # Full attention
            softcap=50.0           # Gemma 2 soft capping
        )
        return attn_output
```

Quantization Support

```
# BitsAndBytes quantization for memory efficiency
from transformers import BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_use_double_quant=True
)

# Load model with quantization
model = AutoModelForCausalLM.from_pretrained(
    "google/gemma-2-9b-it",
    quantization_config=bnb_config,
    device_map="auto",
    torch_dtype=torch.float16
)
```

Performance Monitoring

Gemma 2 Specific Metrics

```
# Additional metrics for Gemma 2
MONITORING_METRICS = {
    "attention_entropy": "Measure attention distribution",
    "soft_capping_activation": "Monitor soft capping usage",
    "sliding_window_efficiency": "Local attention effectiveness",
    "instruction_following_score": "IT model specific metric",
    "perplexity": "Language modeling quality"
}
```

Expected Performance

```
Training Characteristics:
- Slower convergence due to 9B parameters
- Better instruction following (IT variant)
- Higher memory requirements
- More stable training with soft capping

Performance Targets:
- BLEU-4: > 0.18 (higher than Llama due to IT)
- CLIP Score: > 0.28
- Instruction Following: > 0.85
- Training Time: 4-6 hours (larger model)
```

Troubleshooting & Status

Current Status

- **Environment**: ✅ PyTorch 2.7.0+cu118 ready
- **Model Support**: ✅ Transformers 4.52.3 supports Gemma 2
- **Memory Planning**: ✅ Optimized for >24GB GPUs
- **Queue Position**: ⏳ Priority-based waiting

Potential Issues & Solutions

1. **Memory Overflow**: Use gradient checkpointing + smaller batch size
2. **Slow Convergence**: Adjust learning rate schedule

3. **Instruction Format:** Ensure proper Gemma 2 IT formatting
4. **Attention Issues**: Monitor soft capping effectiveness

Questo job rappresenta il fine-tuning del modello più avanzato (Gemma 2 9B IT) con ottimizzazioni specifiche per l'architettura e capacità di instruction following.