Job 2584729 - Baseline Models Evaluation

Informazioni Generali

- **Job ID**: 2584729
- **Nome**: baseline_eval
- **Utente**: ediluzio
- **Status**: RUNNING (4:59:41 ore)
- **Partizione**: all_usr_prod
- **Nodo**: rezzonico
- **GPU**: Condivisa con CLIP evaluation

Descrizione Tecnica

Obiettivo del Job

Valutare le performance dei modelli baseline pre-trained (BLIP, BLIP2, ViT-GPT2, GIT-base, Ide Fix 3, Flores 2) su SVG captioning per stabilire un benchmark di confronto con i modelli fine-tuned.

Stack Tecnologico Aggiornato

- **PyTorch**: 2.7.0+cu126 (auto-upgrade da 2.2.0+cu118)
- **Transformers**: 4.52.3
- **Accelerate**: 1.7.0
- **Evaluation Metrics**: BLEU, METEOR, ROUGE, CIDEr, CLIP Score
- **Image Processing**: CairoSVG, Pillow

Architettura dei Modelli Baseline

1. BLIP (Bootstrapping Language-Image Pre-training)

```
Architecture: ViT + BERT
- Vision Encoder: ViT-B/16 (86M params)
- Text Decoder: BERT-base (110M params)
- Total Parameters: ~196M
- Pre-training: 129M image-text pairs
```

2. BLIP2 2.7B

```
Architecture: ViT + Q-Former + LLM
- Vision Encoder: ViT-g/14 (1.4B params)
- Q-Former: 188M params (32 queries)
- Language Model: OPT-2.7B
- Total Parameters: ~4.3B
- Innovation: Frozen LLM + learnable queries
```

3. ViT-GPT2

```
Architecture: ViT + GPT2
- Vision Encoder: ViT-B/16 (86M params)
- Text Decoder: GPT2-base (117M params)
- Total Parameters: ~203M
- Training: End-to-end fine-tuning
```

4. GIT-base (GenerativeImage2Text)

```
Architecture: ViT + Transformer Decoder
- Vision Encoder: ViT-B/16 (86M params)
- Text Decoder: 6-layer Transformer (140M params)
- Total Parameters: ~226M
- Pre-training: 0.8B image-text pairs
```

5. Ide Fix 3

```
Architecture: Custom SVG-aware model
- Specialized for technical diagrams
- SVG structure understanding
- Parameters: ~150M
```

6. Flores 2 Base

```
Architecture: Multilingual captioning
- Cross-lingual capabilities
- Parameters: ~200M
- Languages: 200+ supported
```

Pipeline di Evaluation

Fase 1: Environment Setup

1. **PyTorch Upgrade Process**:

```
# Automatic upgrade triggered
pip install torch==2.7.0 torchvision==0.22.0
# CUDA 12.6 compatibility check
# Memory optimization for shared GPU
```

2. **Model Loading Strategy**:
   - Sequential loading per evitare OOM
   - Model offloading dopo evaluation
   - Mixed precision per ottimizzazione

Fase 2: Data Preprocessing

1. **SVG Processing Pipeline**:

```
SVG → CairoSVG(dpi=300) → RGBA(1024x1024) → RGB(white_bg) → Resize(224x224)
```

2. **Test Set**: `data/processed/xml_format/test_set_final_xml_reduced_rgb.json`
   - **Size**: 10% del dataset originale (per velocità)
   - **Samples**: ~500 SVG-caption pairs
   - **Categories**: Technical diagrams, flowcharts, icons

Fase 3: Model-Specific Inference

BLIP Inference

```
# BLIP processing
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning

# Inference
inputs = processor(image, return_tensors="pt")
generated_ids = model.generate(**inputs, max_length=50, num_beams=5)
caption = processor.decode(generated_ids[0], skip_special_tokens=True)
```

BLIP2 Inference

```
# BLIP2 processing
processor = Blip2Processor.from_pretrained("Salesforce/blip2-opt-2.7b")
model = Blip2ForConditionalGeneration.from_pretrained("Salesforce/blip2-opt-2.7b")

# Q-Former + OPT-2.7B generation
```

```
inputs = processor(image, return_tensors="pt")
generated_ids = model.generate(**inputs, max_length=100, do_sample=True, temperature=0
```

ViT-GPT2 Inference

```
# ViT-GPT2 processing
feature_extractor = ViTFeatureExtractor.from_pretrained("nlpconnect/vit-gpt2-image-cap
tokenizer = GPT2TokenizerFast.from_pretrained("nlpconnect/vit-gpt2-image-captioning")
model = VisionEncoderDecoderModel.from_pretrained("nlpconnect/vit-gpt2-image-captionin

# Vision-to-text generation
pixel_values = feature_extractor(image, return_tensors="pt").pixel_values
generated_ids = model.generate(pixel_values, max_length=50, num_beams=4)
```

Fase 4: Metrics Calculation

BLEU Score (Bilingual Evaluation Understudy)

```
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction

# BLEU-1, BLEU-2, BLEU-3, BLEU-4
weights_1 = (1.0, 0, 0, 0)
weights_2 = (0.5, 0.5, 0, 0)
weights_3 = (0.33, 0.33, 0.33, 0)
weights_4 = (0.25, 0.25, 0.25, 0.25)

bleu_scores = {
    'BLEU-1': sentence_bleu([reference], candidate, weights_1),
    'BLEU-2': sentence_bleu([reference], candidate, weights_2),
    'BLEU-3': sentence_bleu([reference], candidate, weights_3),
    'BLEU-4': sentence_bleu([reference], candidate, weights_4)
}
```

METEOR (Metric for Evaluation of Translation with Explicit ORdering)

```
from nltk.translate.meteor_score import meteor_score

# Considers synonyms, stemming, paraphrases
meteor = meteor_score([reference], candidate)
```

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

```
from rouge_score import rouge_scorer
```

```
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference, candidate)
```

CIDEr (Consensus-based Image Description Evaluation)

```
from pycocoevalcap.cider.cider import Cider

# Measures consensus between candidate and multiple references
cider_scorer = Cider()
cider_score = cider_scorer.compute_score(references, candidates)
```

CLIP Score

```
import clip_score

# Semantic similarity between image and generated caption
clip_score_value = clip_score.compute_clip_score(image, caption)
```

Ottimizzazioni per GPU Condivisa

Memory Management

- **Sequential Model Loading**: Un modello alla volta
- **Explicit Memory Cleanup**: `torch.cuda.empty_cache()`
- **Model Offloading**: CPU offload tra evaluations
- **Batch Size Adaptation**: Ridotto per shared GPU

Performance Strategies

```
# Memory-efficient evaluation
@torch.no_grad()
def evaluate_model_batch(model, images, batch_size=4):
    results = []
    for i in range(0, len(images), batch_size):
        batch = images[i:i+batch_size]
        with torch.cuda.amp.autocast():  # Mixed precision
            outputs = model.generate(batch)
        results.extend(outputs)
        torch.cuda.empty_cache()  # Free memory
    return results
```

Output Structure

Results Directory: `experiments/baseline_evaluation/`

```
baseline_evaluation/
├── blip_results.json
├── blip2_results.json
├── vit_gpt2_results.json
├── git_base_results.json
├── ide_fix3_results.json
├── flores2_results.json
├── aggregated_metrics.csv
├── statistical_analysis.json
└── comparison_plots/
    ├── bleu_comparison.png
    ├── meteor_comparison.png
    ├── rouge_comparison.png
    ├── cider_comparison.png
    └── clip_comparison.png
```

Metrics per Model

```
{
  "model_name": "BLIP",
  "metrics": {
    "BLEU-1": 0.234,
    "BLEU-2": 0.156,
    "BLEU-3": 0.098,
    "BLEU-4": 0.067,
    "METEOR": 0.187,
    "ROUGE-1": 0.245,
    "ROUGE-2": 0.134,
    "ROUGE-L": 0.198,
    "CIDEr": 0.456,
    "CLIP": 0.234
  },
  "inference_time": 1.23,
  "memory_usage": "4.2GB"
}
```

Expected Performance Ranking

Anticipated Results (based on literature)

1. **BLIP2 2.7B**: Highest scores (large parameter count)
2. **GIT-base**: Strong performance on technical content
3. **BLIP**: Solid baseline performance
4. **ViT-GPT2**: Good vision-language alignment
5. **Ide Fix 3**: Specialized for technical diagrams
6. **Flores 2**: Multilingual but may struggle with technical SVGs

Statistical Analysis

Significance Testing

- **ANOVA**: Compare means across models
- **Post-hoc Tests**: Pairwise comparisons
- **Effect Size**: Cohen's d for practical significance
- **Confidence Intervals**: 95% CI for each metric

Questo job stabilisce il benchmark fondamentale per valutare l'efficacia dei modelli fine-tuned Llama e Gemma.