# CVPR 2026 Compute Reporting Form (CRF) - Example

---

## Instructions

**Important Disclaimer:**This compute reporting is collected for transparency, research insights, and recognition purposes only. **Compute reporting data will NOT be used in the peer review process or publication decisions.**Your paper's scientific merit is evaluated independently of computational resource usage.

This form is designed to collect information about computational resources used for your research. All fields marked with (Required) are **required** . Optional fields help provide more complete compute impact assessments.

**Submission Guidelines:**

- CRF is due on the **Supplementary Materials Deadline**
- Complete this form for the **primary experiments** reported in your paper
- If multiple training runs were conducted, report the **final model** used for results
- For ensemble methods, report **combined compute** across all component models
- For multiple models, report the **most competitive ("strongest")** model
- For multiple baselines, report the **most competitive ("strongest")** baseline
- For multiple evaluation metrics, report the **most relevant**metric. Note the plurality of metrics and note your choice of **most relevant** in Efficiency calculation method/reasoning question

**Note on W&B Logs:**Anonymized W&B logs are optional but encouraged. If you choose to submit logs, please upload them on the **Supplementary Materials submission page** , not through this form. See Section 4 for anonymization instructions.

**Compute Metrics:**

- **GPU+CPU Hours**: Easier to track, reflects actual resource usage, hardware-dependent
- **FLOPs**: Hardware-independent, better for theoretical comparison, requires calculation tools
- Choose the metric most appropriate for your work. FLOPs are particularly useful for architectural comparisons and when hardware varies significantly.

---

## Section 1: Hardware Specifications (Required)

### 1.1 Processing Units

**CPU Information:**

CPU Model: AMD EPYC 7282

Number of CPU cores used: 32

**GPU Information:**

GPU Type(s): A100 PCIe 40/80GB

Number of GPUs: 8

GPU memory per unit (GB): 80

**Memory and Storage:**

System RAM (GB): 512

Storage type: NVMe SSD

### 1.2 Infrastructure Type

**Select your primary infrastructure:**
Cloud - Commercial cloud provid

**If Cloud, specify provider:**

Provider: AWS

Instance type(s): N/A

---

## Section 2: Task and Compute Reporting (Optional, Highly Encourared)

**Instructions:**Report ONLY the compute needed to reproduce the specific comparison reported in your paper (your best model vs. strongest baseline on the primary dataset/task).

### 2.1 Task Category

Object Detection

**Primary Task Category:**

If Other, specify here:

## 2.2 Task Evaluation

**Performance Comparison:**

- Performance metric name: FID
- Your method - metric value: 2.01
- Baseline method name: E2-Rectified Flow (Liu et al., 202
- Baseline - metric value: 4.85
- Performance metric Percentage Improvement: 58.56

**Dataset Context:**

Primary dataset: ImageNet-1K

## 2.3 Compute for Reported Results

**Select Compute Metric:**

(•) GPU+CPU Hours ( ) FLOPs (Floating-Point Operations)

**Note:** If reporting FLOPs, see Section 6 for instructions on computing FLOPs.

**Your Method - Model Information:**

Model size (number of parameters): 350M

**Your Method - Compute Cost:**

Total compute (hours or FLOPs): 1000

**If training involved:**

- Training set size (number of samples): 1281167
- Training compute (hours or FLOPs): 960
- Number of epochs: 400
- Batch size: 2048

**If reporting FLOPs, also provide:**

- FLOPs per forward pass:

**If inference involved:**

- Test set size (number of samples): 50000

- Inference compute per 1000 instances (hours or FLOPs): 0.8
- Total inference compute (hours or FLOPs): 40

**If reporting FLOPs:**
- FLOPs per inference instance:
- FLOPs calculation tool used: ptflops
- If Other or Custom, describe:

### 2.4 Compute Efficiency

**Efficiency Calculation:**

Compute per Performance Metric Percentage Improvement: 17.08

(If using hours: hours per percentage point. If using FLOPs: FLOPs per percentage point)

Efficiency calculation method/reasoning:

Computed as 1000 GPU+CPU hours / 58.56 percentage point improvement = 17.08 hour

---

## Section 3: Additional Computational Details (Optional)

### 3.1 Total Development Compute

**Instructions:**If you tracked your complete development process, report the TOTAL GPU+CPU hours spent to achieve the results in your paper, including all experiments, failed attempts, and iterations.

**Total Development GPU+CPU hours:** 3500

**Breakdown by Development Stage:**
(Provide approximate percentages that sum to 100%)

Training from scratch: 40 %
- GPU+CPU hours: 1400

Fine-tuning pretrained models: 15 %
- GPU+CPU hours: 525

Model distillation/compression: 0 %

- GPU+CPU hours: 0

Hyperparameter search: 20 %
- GPU+CPU hours: 700
- Configurations tested: 48

Ablations: 15 %
- GPU+CPU hours: 525
- Configurations tested: 12

Inference-based development: 5 %
- GPU+CPU hours: 175

Failed experiments/debugging: 5 %
- GPU+CPU hours: 175

Other: 0 %
- GPU+CPU hours: 0

**Development Timeline:**
- Project start date: 2024-06-01
- Final results date: 2025-01-15
- Total calendar time: 7

**Additional Context:**

We conducted extensive hyperparameter tuning for learning rate schedules and architect

## 3.2 Code Efficiency

Framework used: PyTorch

Mixed precision training: Yes - FP16

Distributed training: (•) Yes ( ) No

If yes, specify method: DDP (DistributedDataParallel) wi

---

# Section 4: Weights & Biases Log Submission (Optional)

## 4.1 W&B Log Upload

We encourage submitting anonymized Weights & Biases logs to enable more precise compute and impact calculations. This submission is **entirely optional** but helps provide more objective data.

**Privacy and Anonymization:**

Your privacy is our priority. Before submitting any W&B logs:
1. **Use the provided anonymization tool** to remove all personally identifiable information (PII)
2. **Review anonymized logs** to ensure no sensitive information remains
3. **Only submit logs** that you're comfortable sharing for research purposes

## 4.2 How to Anonymize Your W&B Logs

**Step 1: Install the Anonymization Tool**

```
git clone https://github.com/vipavlovic/wandbanonymizer.git
cd wandbanonymizer
```

**Step 2: Anonymize Your Logs**

For basic anonymization (fast):

```
python wandb_anonymizer.py /path/to/your/wandb/logs -o /path/to/anonymized/logs
```

For enhanced anonymization (recommended, more accurate):

```
# Install additional dependencies
pip install presidio-analyzer presidio-anonymizer
python -m spacy download en_core_web_lg

# Run enhanced anonymization
python presidio_wandb_anonymizer.py /path/to/your/wandb/logs -o /path/to/anonymized/lo
gs --stats
```

**Step 3: Review Anonymized Output**

- **Check anonymized files** in the output directory
- **Verify that sensitive information** has been properly anonymized:

- Names → Person_abc123
- Emails → anon_xyz123@example.com
- File paths → /dir_anon_*/anon_*
- Hostnames → host-anon123.example.com

- **Ensure model performance metrics** remain unchanged

## 4.3 What Gets Anonymized

The tool automatically anonymizes:

- Personal names, emails, phone numbers
- File paths and hostnames
- Git repository URLs and API keys
- IP addresses and system usernames
- W&B run IDs and project names
- Credit card numbers and SSNs (if present)

**Technical metrics and model performance data remain unchanged.**

## 4.4 Upload Your Anonymized Logs

**File Upload:**

☑ I have anonymized my W&B logs using the provided tool

☑ I have reviewed the anonymized output and confirmed no PII remains

☑ I consent to sharing these anonymized logs for research purposes

**Upload anonymized files here:** File upload will be handled separately on the **Supplementary Materials** submission page.

**Log Summary:**

Log file size (approximate): 245 MB

Date range of experiments: June 2024 - January 2025

Additional notes:

Logs include main training runs, final hyperparameter sweep, and best model checkpoints

## 4.5 Benefits of W&B Log Submission

Submitting anonymized logs enables more precise efficiency calculations based on actual GPU utilization patterns and provides better resource optimization insights using real-time hardware data. Your contribution supports community benchmarking for computational efficiency and generates valuable research insights into training efficiency across different methodological approaches. Additionally, W&B log submissions enhance your eligibility for "CVPR Compute Gold Star" and other recognition awards by providing objective data that demonstrates computational excellence.

**Your anonymized data helps the entire ML community understand and improve computational efficiency while increasing your chances for conference recognition.**

---

## Section 5: Verification and Submission

### 5.1 Author Verification

☑ I confirm that the information provided accurately represents the computational resources used for the experiments reported in this paper

☑ I understand that this information may be used for efficiency analysis, community benchmarking, and recognition awards

☑ If submitting W&B logs, I confirm they have been properly anonymized and I consent to their use for research purposes

☑ I understand my paper is eligible for computational efficiency recognition based on this submission

**Paper ID:** 12345

**Date:** 2025-02-28

### 5.2 Additional Information

**Comments or Special Circumstances:**

Our training was conducted on a shared cluster with variable GPU availability, which may

**Estimated completion time for this form: 10-15 minutes (excluding W&B log preparation)**

---

## Section 6: Instructions for Computing FLOPs

FLOPs (floating-point operations) provide a hardware-independent measure of computational cost. If you selected FLOPs as your compute metric in Section 2.3, use the following guidelines to calculate and report FLOPs.

## Computing FLOPs Using Python Libraries

**Recommended Libraries:**

**1. ptflops** (`pip install ptflops`)

```
from ptflops import get_model_complexity_info
macs, params = get_model_complexity_info(model, (3, 224, 224),
                                         as_strings=True,
                                         print_per_layer_stat=True)
# Note: 1 MAC (Multiply-Accumulate) ≈ 2 FLOPs
```

**2. fvcore** (`pip install fvcore`)

```
from fvcore.nn import FlopCountAnalysis
flops = FlopCountAnalysis(model, input_tensor)
print(f"Total FLOPs: {flops.total()}")
```

**3. thop** (`pip install thop`)

```
from thop import profile
flops, params = profile(model, inputs=(input_tensor,))
```

**4. DeepSpeed Flops Profiler**

```
from deepspeed.profiling.flops_profiler import get_model_profile
flops, macs, params = get_model_profile(model, input_shape=(batch_size, channels, heig
ht, width))
```

## Reporting Guidelines

**General:**

- Report total FLOPs in scientific notation (e.g., 1.5e15 for 1.5 PetaFLOPs)
- Include both training FLOPs (if applicable) and inference FLOPs
- Be consistent in your calculation method across all reported values

**For Training:**

```
Training FLOPs = FLOPs_per_forward_pass × num_training_samples × num_epochs × 3
```

The factor of 3 accounts for forward pass, backward pass, and gradient update.

**For Inference:**

```
Inference FLOPs = FLOPs_per_forward_pass × num_test_samples
```

**Common FLOPs Units:**

- KFLOPs: 10^3 FLOPs (1e3)
- MFLOPs: 10^6 FLOPs (1e6)
- GFLOPs: 10^9 FLOPs (1e9)
- TFLOPs: 10^12 FLOPs (1e12)
- PFLOPs: 10^15 FLOPs (1e15)

### Important Notes

- Different libraries may report slightly different FLOPs counts due to different counting conventions
- Specify which tool you used in Section 2.3
- For custom operations not captured by standard libraries, document your calculation method
- MACs (Multiply-Accumulate operations) are often reported; use the conversion: 1 MAC ≈ 2 FLOPs

---

## Recognition Program

### Computational Efficiency Awards

By participating in this compute reporting initiative, your paper becomes eligible for several recognition opportunities that celebrate resource-conscious research and computational innovation.

**Awards and Recognition:**

Papers demonstrating exceptional computational efficiency may receive the **"Efficient CVPR" Badge** based on GPU+CPU hours or FLOPs per accuracy point gained over established baselines in their respective domains. This objective metric allows fair comparison within similar task categories, recognizing research that achieves meaningful performance improvements while minimizing computational resources.

For truly outstanding efficiency achievements, papers may earn the **"CVPR Compute Gold Star" Designation** . This prestigious recognition is reserved for papers showing the lowest GPU+CPU hours or FLOPs per accuracy point ratio in their category, representing the highest level of computational excellence relative to performance gains achieved.

Papers that contribute high-quality anonymized W&B log submissions will receive **"CVPR Compute Transparency Champion"**recognition. This award acknowledges authors who provide comprehensive, detailed computational logs that enable precise efficiency analysis and contribute valuable data to the community's understanding of training dynamics.

Throughout the conference, recognized papers will receive **Special Presentation Recognition**including dedicated poster session highlights, mentions in oral presentation introductions, and featured placement on the conference website's sustainability showcase section.

**Benefits of Participation:**

Participation in this program increases visibility for resource-conscious research approaches and positions authors as community leaders in sustainable AI practices. The submitted data provides valuable benchmarking information for your research domain and can lead to potential collaboration opportunities with other efficiency-focused researchers in the community.

**Award Selection Process:**

Awards are determined by the Compute Reporting Committee using objective metrics and data quality assessments. The **"Efficient CVPR" Badge** and **"CVPR Compute Gold Star"**are awarded based on GPU+CPU hours or FLOPs per accuracy point gained over domain-specific baselines, ensuring fair comparison within similar task categories (e.g., object detection, language modeling, generative models). The **"CVPR Compute Transparency Champion"**award recognizes the quality and completeness of submitted W&B logs, including training metrics, resource utilization data, and experimental documentation. Award results will be announced during the conference, giving recipients immediate recognition before the broader research community.

Your participation helps recognize and promote computational sustainability in AI research while providing valuable insights to the broader community.

---

## Technical Support

**Questions about compute reporting?** Contact: crf_cvpr2026@computer.org

**Issues with W&B anonymization?**

- Check https://github.com/vipavlovic/wandbanonymizer for documentation
- Report issues at the repository issues page

**Privacy concerns?** Contact: crf_cvpr2026@computer.org

---

*This form is part of our initiative to improve transparency in computational resource usage and promote efficiency in machine learning research. Thank you for your participation in making our field more efficient and accountable.*