OpenCV GSoC 2025 Proposal Quantized models for OpenCV Model Zoo (90-175 hours)

Table of Contents

Name

David Magdy Nagib Yacoub

Contact Info/Personal Links

- [LinkedIn Account]
- [GitHub Account]
- Email: divomagdy@gmail.com

Synopsis

This project aims to **expand OpenCV Zoo's model collection** by adding and optimizing three high-demand, state-of-the-art models currently missing from the repository:

1. **YOLOv8-Nano** (Ultralytics): A lightweight object detector optimized for edge devices.
2. **PP-OCRv4** (PaddlePaddle): A significantly more accurate text detection/recognition model than existing Zoo options.
3. **EfficientNet-Lite** (Google): A mobile-friendly image classification model with built-in quantization support.

Key Innovations:

- First-time integration of these popular models into OpenCV's ecosystem.
- INT8 quantization for **YOLOv8-Nano** and **PP-OCRv4** to enable 2-4x faster CPU inference.
- Production-ready pipelines (conversion → validation → documentation) for seamless community adoption.

Impact:

- Bridges gaps in OpenCV's model coverage (e.g., no modern YOLO variants in Zoo).
- Empowers developers with efficient, ready-to-use versions of cutting-edge models.

Benefits

For OpenCV Users

- **Access to cutting-edge models:** Get production-ready versions of high-demand architectures like YOLOv8 and PP-OCRv4, currently missing from the Zoo.
- **Optimized for real-world use:** INT8-quantized variants of new models deliver **2-4x faster CPU inference** (tested on Kaggle) with minimal accuracy loss.
- **Smaller deployments:** Quantized models are **~4x smaller** than FP32 versions, ideal for edge/embedded systems.

For OpenCV Maintainers

- **Expanded model coverage:** Adds modern architectures (YOLOv8, PP-OCRv4) to fill critical gaps in the Zoo's offerings.
- **Reduced support burden:** Pre-optimized models with documented tradeoffs reduce user questions about performance.
- **Future-proof pipelines:** Clean conversion and quantization scripts make it easy to add more models later.

For the Community

- **Reproducible workflows:** Full documentation of model conversion, quantization steps, and benchmarking.
- **Educational resource:** Serves as a reference for developers learning model optimization techniques.
- **Hardware-aware:** Includes benchmarks across CPU tiers (Kaggle, consumer laptops) to set realistic expectations.

Deliverables

Type	Description	Notes
New Models	▪ **YOLOv8-Nano** (FP32 + INT8) ▪ **PP-OCRv4** (FP32 + INT8) ▪ **EfficientNet-Lite** (FP32)	*First-time additions to OpenCV Zoo*
Quantization Scripts	ONNX Runtime/PaddlePaddle scripts to convert FP32 → INT8 (for YOLOv8 & PP-OCRv4)	*Reusable for future models*
Validation Suite	▪ Accuracy tests (mAP for YOLOv8, F-score for PP-OCRv4) ▪ CPU speed benchmarks (Kaggle)	*Compares FP32 vs INT8*
Documentation	▪ Model cards (input/output specs, usage) **"How to Quantize New Models"** guide	*For Zoo contributors*
Code Contributions	Only if OpenCV DNN needs patches for new ops (e.g., YOLOv8's SiLU in INT8)	*Mentor-guided*

Project Details

Target Models to Add to OpenCV Zoo

(Prioritized by impact and feasibility)

Priority	Model	Why Include?	Key Metric	Data for Validation
1	YOLOv8 -Nano	▪ 3x faster than YOLOx on CPU	mAP@0.5	COCO (100-image subset)
2	PP-OCRv4	▪ 15% more accurate than Zoo's current `text_detection_db` ▪ PaddlePaddle's easy quantization tools	F-score (text)	ICDAR2017 (50 images)
3	Efficient Net-Lite	▪ Google-prequantized for mobile ▪ Missing from Zoo despite wide industry use	Top-1 Accuracy	ImageNet (100 samples)

(Backup: RT-DETR if YOLOv8 integration fails)

Execution Plan

1. **Model Conversion & Integration**
   - Convert models to OpenCV-friendly formats:
   - YOLOv8: PyTorch → ONNX (with dynamic batch support)
   - PP-OCRv4: PaddlePaddle → ONNX
   - Add to Zoo with:
   - Standardized inference scripts (Python/C++)
   - Metadata (input shapes, expected accuracy)
2. **Quantization (YOLOv8 & PP-OCRv4 only)**
   - Use ONNX Runtime PTQ for INT8 conversion
   - Calibrate with tiny datasets (50-100 images) to respect Kaggle limits
3. **Validation**
   - Accuracy: Compare FP32 vs INT8 using native OpenCV metrics
   - Speed: FPS tests on Kaggle CPU (single-threaded/multi-threaded)
4. **Documentation**
   - Model cards with:
   - Usage examples
   - Benchmark results
   - Quantization tradeoffs ("INT8 speeds up X% but drops accuracy Y%")

Risk Management

Risk	Solution
YOLOv8's `SiLU` op not supported in OpenCV DNN	Use ONNX Runtime as fallback backend
PP-OCRv4 text accuracy drops >5%	Keep FP32 as default, document INT8 as "experimental"
Kaggle timeouts	Pre-download calibration datasets to `/kaggle/working`

Project Timeline (175 Hours)

Start Date: July 1, 2025

Week	Task	Hours	Key Deliverables	Notes
Week 1	Setup + YOLOv8-Nano conversion	20	yolov8n.onnx in OpenCV Zoo format	Kaggle env ready
Week 2	YOLOv8 quantization (INT8)	15	yolov8n_int8.onnx, calibration script	COCO 100-image subset
Week 3	YOLOv8 validation + docs	25	mAP/FPS benchmarks, usage guide	CPU-only tests
Week 4	PP-OCRv4 conversion	20	ppocrv4.onnx added to Zoo	ICDAR50 images
Week 5	PP-OCRv4 quantization (INT8)	15	ppocrv4_int8.onnx, text accuracy tests	F-score metrics
Week 6	Finalize PRs for both models	20	OpenCV Zoo PRs (YOLOv8 + PP-OCRv4)	Includes docs
Weeks 7–8	Buffer/testing	30	Address mentor feedback, fix issues	Critical review period
Weeks 9–12	*Stretch goal*: EfficientNet-Lite	30	Add FP32 model to Zoo	Only if on track

Total Hours: 175 (reduce to 90 by removing Weeks 9–12 + half of buffer time)

When can you begin work?

- 01/July/2025, since my final exams finish at the end of June.

Do you have any possible school or work conflicts with your schedule?

- No, there aren't any known conflicts currently, but I will make sure to communicate directly If this changes.

Short Bio

About me, what I am studying, what I am passionate about, and what makes me the best candidate:

I am David Magdy, a third-year Computer Science student at **Ain Shams University** in Egypt, During my first year of university, I focused mainly of being fluent in writing code and be good at problem solving. I tried a decent amount of technologies (like game development and web) but my main passion was toward Artificial Intelligence. During summer break I entered a summer training sponsored by my college and held by **ApplAI** student activity. I started learning machine learning fundamentals and participated in the training's competition and achieved 5th place among a variety of 50+ different year students.

I always wanted to try getting into AI since high-school and this training was my kick-off to start pursuing my career in AI. I started by asking connections for learning resources and courses and started to get into advanced topics day by day, till I realize I am more into computer vision as a specialized field.

In the beginning of the second year I wanted to share this knowledge with freshmen to ease up their journey and spreading the knowledge of how AI is more fun than just some matrix-multiplications.

Having guided **30+ trainees** through their AI and computer vision journeys, I've seen firsthand how **accessible tools and clear examples** accelerate learning. This OpenCV project resonates deeply because it aligns with my dual goals:

1. **Advancing my own expertise** in model optimization and open-source contribution.
2. **Creating a blueprint** for newcomers—by documenting challenges and best practices, I can empower future contributors to navigate open-source CV projects confidently.

This OpenCV project excites me because it combines my love for efficient AI systems with my goal to create accessible tools for students and developers alike.

Additional info on what makes me the best candidate for this project?

While I may not yet be an expert in ONNX Runtime or OpenCV's DNN module, I bring three crucial strengths that make me uniquely qualified:

1. **Proven ability to rapidly master technical concepts**: When I discovered AI, I went from fundamentals to training my own computer vision models in just 2 months, eventually placing 5th in ApplAI's competition. I'll apply this same focused learning approach to master quantization tools.
2. **Real-world teaching experience**: Having trained 30+ students in AI fundamentals, I understand both the theory and how to make complex topics accessible. This means I'll not only deliver working INT8 models but also create clear documentation to help others reproduce and build on this work.
3. **Alignment with OpenCV's mission**: My passion lies in making computer vision practical and accessible. This project excites me because quantized models directly help developers working with resource constraints - the same challenges my students face when deploying their first CV projects.

I've already begun preparing by:

- Running OpenCV's DNN module tutorials in Kaggle notebooks
- Prototyping basic model quantization using Python
- Studying ONNX Runtime documentation to accelerate my Week 1 productivity

What I lack in specific tool experience, I make up for in dedication, teachability, and a track record of delivering results quickly when motivated by meaningful problems.

Relevant coding projects that I have worked on:

1. **COCO:** Educational-purpose mini-deep learning framework specialized in computer vision.
2. **Social-Media Sentiment Analysis:** A complete analysis and prediction of social media user sentiment.