Further accelerate OpenCV DNN on GPU using FP16 arithmetics

- Author: Vadim Pisarevsky
- Link: The feature request
- Status: **Draft**
- Platforms: Mostly GPUs supporting FP16 arithmetics
- Complexity: 1-2 man-months

Introduction and Rationale

Performance of DL inference is crucial nowadays when more and more tasks are effectively, but not always efficiently are solved using deep nets. OpenCV includes quite fast DL inference engine (OpenCV DNN) with OpenCL acceleration path. Even though we are able to load networks with FP16 and quantized coefficients, all the calculations are currently done in FP32. On CPUs this is fine, but on many modern GPUs FP16 version could run faster, often by factor of 2 or more. It would be unwise not to use this capability of those GPUs, especially given that in majority of cases network weights can just be truncated from FP32 to FP16 without any retraining, tuning etc.

Proposed solution

Since OpenCV DNN now includes several dozens of layers and the number is growing, it would be too time consuming to reimplement all those layers in FP16. Besides, not all the layers are implemented in OpenCL, so they should fall back to CPU where no FP16 processing instructions exist. So, the idea is to add FP16 path only to the convolution layer and maybe other layers immediately following it (e.g. ReLU, batch normalization etc.).

The network (`cv::dnn::Net` class) may provide a method to set `FP16` precision, which is propagated to each single layer (so that in principle each layer can have `FP16` branch). The convolution layer in this case converts the weights to FP16 (we should retain FP32 copy in this case in the case if user wants to switch to CPU branch for whatever reason) and then during the processing time it converts the input tensor (may be on per-block basis) from FP32 to FP16, does the convolution and optionally batchnorm + ReLU and then converts the results back to FP32. Of course, we loose a bit of performance on the conversion, but at least the second conversion can be done on fly without going to memory, and the first conversion can possibly be combined with caching chunks of input tensor in GPU's local memory. All-in-all, it should likely be a reasonable overhead.

Impact on existing code, compatibility

OpenCV DNN tests should likely be adjusted, because now they do quite strict checks of the final network results. Otherwise, since it will be explicitly selected and non-default execution mode, it should not affect any user code.

Possible alternatives

The above-described approach seem to be the easiest way to get 2x acceleration of OpenCV DNN on FP16-aware GPUs.

Yet, a possible alternative would be to enable GPU path of Intel DL Inference backend and somehow configure it to use FP16 instead of FP32 for inference

References

1. Some data on accelerated DL on the recent iGPU using FP16 arithmetics