White Paper

Vision
FPGA

Low-Precision Networks for Efficient Inference on FPGAs

Light retraining illuminates the way to meeting computer vision specifications

Authors

R. Abra
FPGA Deep Learning Retraining Lead
Intel Programmable Solutions Group

D. Denisenko
Deep Learning Software Engineer
Intel Programmable Solutions Group

R. Allen
Solutions Engineer
Intel Programmable Solutions Group

T. Vanderhoek
Engineering Software Manager
Intel Programmable Solutions Group

S. Wolstencroft
Royal Holloway University

M. Gibson
University of Reading

Table of Contents

Executive Summary

Neural networks are highly compute intensive. As a result, downsizing any of the calculations leads to significant savings in cost, time, and power. One way to downsize calculations is to reduce the size of the parameters. Quantization compresses the parameters in a neural network by reducing the number of bits used to represent them. This in turn reduces both the size of each calculation and the time and resources needed to move the values around the chip.

Implementing a low precision network in hardware provides numerous advantages when it comes to meeting specification. The increased flexibility allows the optimization of:

- Throughput
- Overall power consumption
- Resource usage
- Device size
- TOPs/Watt
- Deterministic latency

These have important benefits to the user experience, particularly where scaling and efficiency are inherent requirements of the application.

There are many options for quantization: small integers (weights and activations quantized to scaled integer values, e.g. *int8*), minifloats (e.g. FP11, FP9), or application-optimized numeric formats. Our end-to-end solution uses Block Floating Point (BFP), which has the advantage of easily halving the hardware footprint while maintaining accuracy at low precisions. This is possible due to BFP's high dynamic range.

Only two scenarios require a little more intervention. Modern, compact networks such as MobileNet and EfficientNet hold little in the way of redundancy, so accuracy can be lost at low precision. Additionally, even the larger, older networks (ResNet, VGG-SSDs) lose accuracy at very low precision. These losses can be recovered within a few retraining epochs when the quantization is modelled in the training flow. Accurate modelling of hardware arithmetic during training allows hardware accuracy to closely match that achieved in software.

The main aim of retraining is therefore to facilitate the compression of FP32 models to comply with application specifications, be they at the Edge or in the Cloud [1]. A reduction in the number of bits inherent in a network enables far more flexibility than is available at the larger size.

Quantizing Neural Networks for FPGAs

FPGAs have a major advantage where specific numeric formats are required [2]. For a given deep learning model, a format can be chosen that optimally balances compute and storage availability on the system with dynamic range and precision required by the model to easily maintain accuracy.

Besides integer and floating-point formats natively supported by Intel® FPGA digital signal processing (DSP) blocks[1] , FPGA logic can be used to implement additional numeric formats that are particularly well suited to deep learning models. BFP combines the extended dynamic range of floating-point format with the low-cost implementation of fixed point.

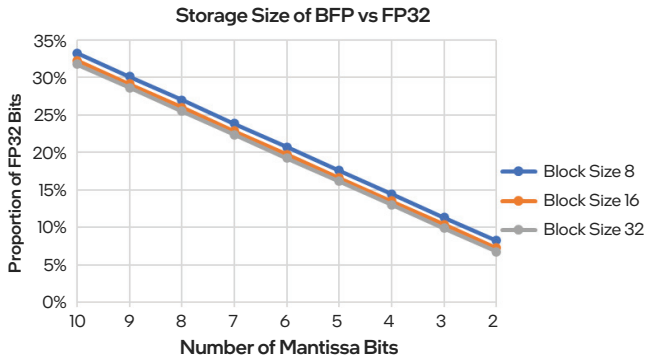

Figure 1. The effect of blocking and mantissa size on the number of bits in a neural network

What is BFP?

In BFP, values are grouped into blocks — nominally of size 32 — within which each value takes the same exponent. To achieve this, grouped numbers are aligned to suit the largest exponent. The mantissas can now be treated as signed integers in, for example, dot-product computation, and combined with the shared exponent in the accumulation step to convert the results to single precision floating-point representation.

Low-precision BFP reduces the payload by eliminating bits from two locations. The shared exponent removes the need for individual assignment, while low-bit mantissas are vastly smaller than the standard 23-bit fraction in FP32 values. Figure 1 shows the bit savings from using different block and mantissa sizes.

int7bfp consists of seven integer bits (including the sign) and five bits for the exponent. *int5bfp*, shown in Figure 2, consists of four integer bits with one sign bit and five exponent bits. Without the BFP blocking, this would be FP9 format, which uses one sign bit, an implicit mantissa bit, three explicit mantissa bits and five exponent bits.

Such small multiplication operations can be efficiently implemented on Intel® FPGAs. For example, a single DSP block in an Intel® Arria® 10 FPGA can implement two *int7bfp* multiplier or, with a few additional ALMs, four *int5bfp* multipliers. Since block size usually encompasses at least eight values, significant storage and compute savings result from utilizing BFP [7].

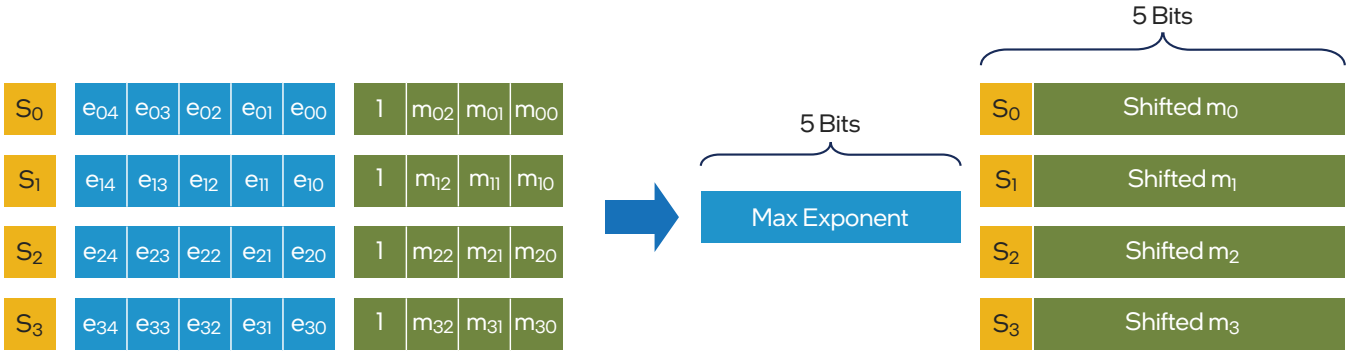

Figure 2. Blocking of four FP9 floating point values to *int5bfp*. The "1" in the left-most mantissa positions is the implicit 1 in floating-point format made explicit prior to conversion. Sign + mantissa bits are now in sign + magnitude integer format

Asymmetric BFP for Increased Savings

Dot product computations need not be symmetrical in BFP. A further reduction in storage size can be achieved if weights and activations are represented with different precisions. Many Convolutional Neural Networks (CNNs) maintain their accuracy even if their weights are represented with fewer bits than the activations. For example, *int5/4bfp* format can be used to store activations in *int5bfp* and weights in *int4bfp*. Dot product engines implemented using an Intel® FPGA would then perform 5 bit x 4 bit integer multiplies, which would achieve more efficient DSP block packing than using *int5bfp* for both weights and activations.

BFP vs Integer Quantization

BFP can be compared against integer quantization with favorable results. Each block of numbers in BFP gets its own scaling factor (2^{max_exp}), unlike integer quantization where such factors are arbitrary floats on a per layer basis. BFP provides an overall higher dynamic range, which can be further adjusted with the size of the block. Additionally, the BFP scaling factor is sized automatically as part of the computation, instead of having to run additional initialization steps.

[1] Some examples of natively supported numeric formats by Intel® FPGAs: 18-bit integer and FP32 on Intel® Arria® 10 FPGA [3][4]; *int8*, FP16, and bfloat16 on Intel® Agilex™ FPGA [5], and *int4* and *int8* tensor block on Intel® Stratix® 10 NX FPGA [6].

BFP for High Inference Accuracy

Many of the high parameter networks – the ResNets, Inception, VGG-based SSDs – quantize well to *int8bfp* and even *int7bfp* without any additional intervention, as shown in Table 1, where green highlights indicate a minimal loss of accuracy from the original FP32 model.

As expected, the drop in accuracy from applying quantization is more perceptible at very low precisions. This effect is exaggerated in the more modern, compact networks such as MobileNet and EfficientNet, which experience some accuracy drop even at higher precisions.

Fortunately, this penalty can be reversed easily. As few as four epochs of retraining – or a dozen for the more challenging networks – can recover the model's accuracy. Where this is insufficient, it is possible for the FPGA to accommodate an increase in the activation bit width for specific layers. Algorithmic techniques are available to determine and retrain a network to account for these changes in precision (see section on Mixed Precision below).

Why Block Floating Point?

- Small resource footprint
- Excellent compatibility with Intel® FPGA DSP blocks
- High dynamic range models weights and activations well at low precisions
- Simplicity in training: no parameter initialization

		Int5/4bfp		*Int7bfp*		*Int8bfp*	
Network	FP32 accuracy reference (%)	Accuracy (%) without retraining	Accuracy (%) with retraining	Accuracy (%) without retraining	Accuracy (%) with retraining	Accuracy (%) without retraining	Accuracy (%) with retraining
Classification (ImageNet)							
ResNet-18	69.76	55.69	69.13	69.67	n/a	69.60	n/a
ResNet-34	73.31	65.09	72.81	72.94	n/a	73.09	n/a
ResNet-50	76.13	60.32	75.60	75.75	n/a	75.95	n/a
Inception v3	77.32	32.70	78.34	77.11	n/a	77.31	n/a
EfficientNet_b0	75.86	0.34	71.96	64.37	75.45	70.48	75.47
MobileNet v2	71.81	6.00	68.99	67.28	71.65	71.12	n/a
SqueezeNet v1.1	58.18	33.09	54.90	57.73	58.15	58.10	n/a
Object Detection (VOC 2007 & 2012)							
SSD300	78.12	73.64	77.92	78.09	n/a	78.08	n/a
SSD512	80.26	74.72	80.00	80.19	n/a	80.08	n/a
Object Detection (COCO 2017)							
TinyYOLO v3	35.7	26.90	31.40	35.50	n/a	35.60	n/a
Semantic Segmentation (CamVid)							
UNet	71.95	63.95	72.36	71.66	n/a	71.89	n/a
ICNet	67.89	59.66	67.09	67.88	n/a	67.87	n/a

Table 1. Indicative Top-1 accuracies for networks both with and without retraining, at *int5/4bfp* (*int5bfp* activations and *int4bpf* weights), *int7bfp* and *int8bfp* – all at block size 32. n/a shows where retraining is not required

Blue : Full precision accuracy
Green : Achieves quantization accuracy within 1 percentage point of the full precision accuracy
Amber : Achieves quantization accuracy within around 5 percentage points of the full precision accuracy
Red : Achieves quantization accuracy significantly lower than full precision accuracy

BFP to Significantly Cut Resource Count

As already seen, the number of bits implicated in low precision quantization is much reduced from the original single precision implementation. This has a large knock-on effect on hardware resources. For instance, the 18 bit input hardened multipliers in an Intel® Arria® 10 FPGA (represented in Figure 3) can be used to implement a single 18 bit x 18 bit multiply, two 6 bit x 6 bit multiplies, or four 4 bit x 3 bit.

Pushing the sign bit into external logic therefore, it is possible to halve the number of DSP blocks used at *int12bfp* (equivalent to blocked FP16) by quantizing to a 7 bit mantissa (*int7bfp*) and halve it again using a 5 bit x 4 bit (*int5/4bfp*) configuration. These savings can be utilized to scale back the hardware footprint or increase throughput.

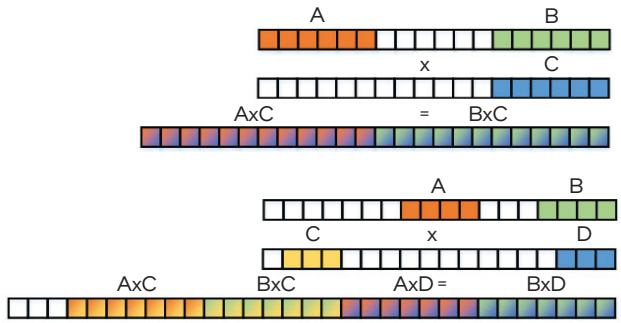

Figure 3. Packing 6x6 bit (*int7bfp*) or 4x3 bit (*int5/4bfp*) multipliers into an Intel Arria 10 FPGA 18 bit multiplier

Meaningful examples mirror real-life applications, which tend to employ variations of standard networks. As typical benchmarks, ResNet 50 and MobileNet v2 are used throughout this section to give an idea of the effects of quantization. The reference is *int12bfp* which is a good proxy for single precision in this context owing to the negligible accuracy loss from downsizing.

Simply by reducing the precision, the associated numbers of Adaptive Logic Modules (ALMs) and RAM blocks roughly follow the pattern seen in multiplier usage. This reduction is reflected in the number of DSP blocks and amplified by decreasing the block size, which – while limiting the throughput – has additional benefits for the footprint.

Also worthy of note is that the choice of network makes a big difference to the frame rate, with the much smaller MobileNet v2 attaining around twice the frame rate of ResNet 50 for the same footprint.

In the following figures, the baseline footprint (indicated in the leftmost columns of Figure 4) at *int12bfp* and block size 32 is:

- 816 M20K RAM blocks
- 551 DSP blocks
- 39,315 ALMs

In Figure 4, reductions in footprint result directly from reduced precision. With a single bitstream for both ResNet 50 and MobileNet v2, it is interesting to note the difference in frame rate.

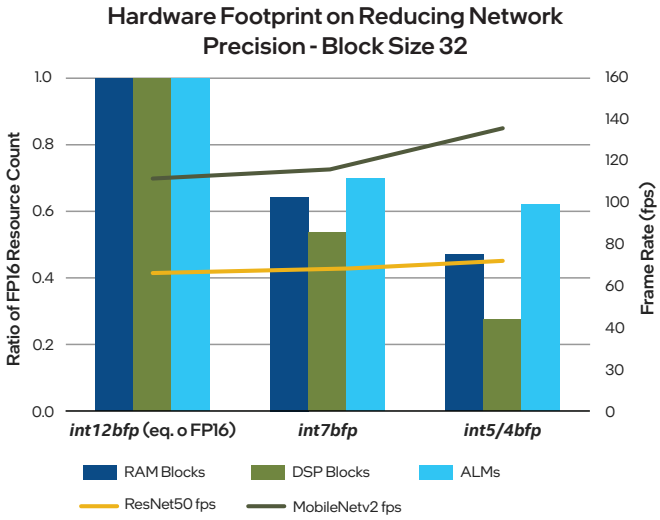

Figure 4. Resource count ratios and frame rate for ResNet 50 and MobileNet v2 inference at block size 32

Optimizing further, halving the block size leads to even greater savings. In Figure 5, while frame rate is halved for MobileNet v2 and reduced by two thirds for ResNet 50, using block size 16 halves RAM utilization and reduces the DSP block usage by two thirds.

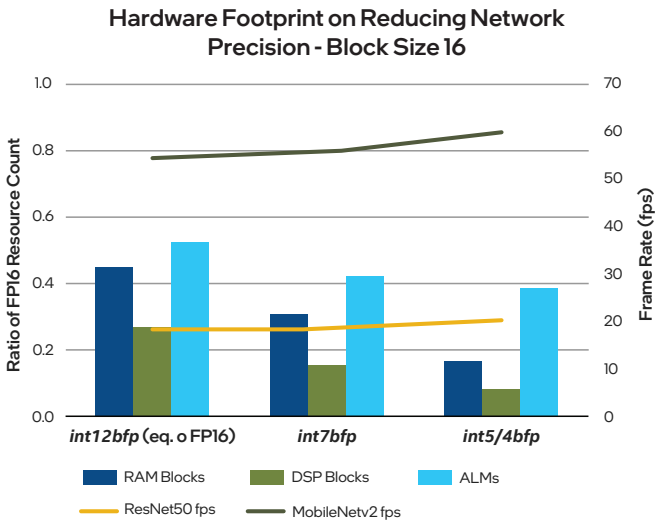

Figure 5. Resource count ratios and frame rate for ResNet 50 and MobileNet v2 inference at block size 16

An additional indication of available trade-offs is given in Figure 6. A realistic application may well use MobileNet v2 at a frame rate of 30 frames per second (fps). In this case, a block size of 8 is sufficient to fulfill the criteria which results in the use of 103 M20K RAM blocks, 31 DSP blocks and 12,635 ALMs at *int5/4bfp*.

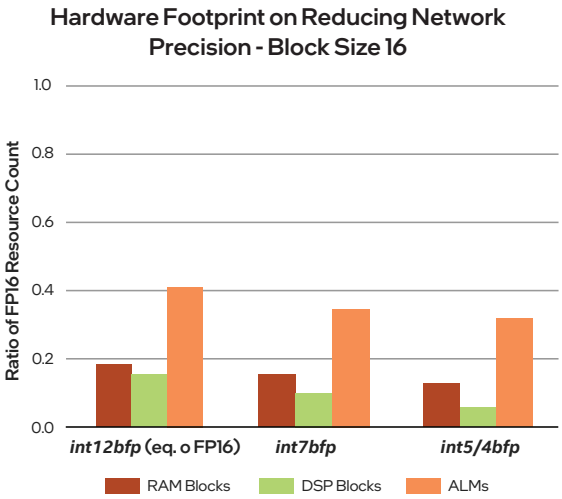

Figure 6. Resource count and accuracy for MobileNet v2 inference at 30 fps with block size 8

As seen, simply by reducing the precision, the associated numbers of ALMs and RAM blocks roughly follow the pattern seen in DSP block usage. This is amplified by decreasing the block size, which – while limiting throughput – compounds the benefits in the footprint. It is worth noting that doubling the number of hardware instances doubles the throughput — potentially useful in the case of multiple input streams or requirements for redundancy in the system.

Mixed Precision – The Advantages of Very Low Precisions with the Accuracy of Higher Precisions

Finally, for those situations where low precision quantization and retraining provides an unacceptable loss in accuracy, certain modes of mixed precision exist that have a cumulatively positive effect. These include "layer type" precision changes to incorporate higher precision hardware kernels, say for depthwise convolutions, and per-layer tensor precision doubling.

Although distinguishing different layer types is straightforward, accuracy uplift — much like training the original network — is determined by exploratory testing. This can be mitigated by using algorithms such as Hessian Aware Quantization (HAWQ), that determine the sensitivity of each convolutional layer to quantization. Each layer can be identified and the bit width of the weights, the activations or both can be doubled accordingly in retraining. In hardware, this augmentation can easily be effected by multiple passes through the PE array [8].

Figure 7 shows the results of training MobileNet v2 at BFP precisions determined by HAWQ. There is built-in flexibility to specify what proportion of the parameters in the convolution layers are doubled.

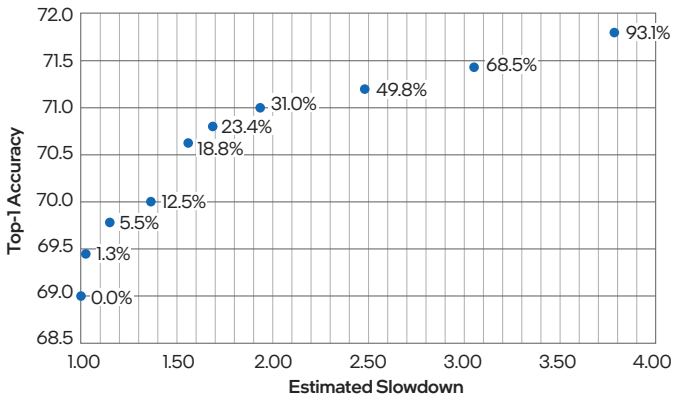

Figure 7. Accuracies achieved with mixed precision layers on MobileNet v2 with a base of int5/4bfp. Percentage figures show proportion of convolution layer parameters doubled

Conclusion

Many AI applications have stringent requirements that are complicated by additional functions needing to be in-lined, such as I/O, clipping, scaling, and dewarp. A big advantage of FPGAs is that these can be included as intellectual property (IP) cores on the same chip and combined as building blocks. While the functions themselves may claim heavy resource usage, the flexibility provided to neural network engines by BFP quantization can reduce the IP footprint and help to meet other specifications such as throughput or performance.

BFP quantization works very well on FPGAs due to the ability to pack integers of certain sizes efficiently into the DSP blocks, which very easily allows the footprint reductions shown on the previous page. A 50% reduction in DSP block usage is achieved simply by reducing the precision to *int7bfp*. This is replicated on a further precision reduction to *int5/4bfp*. Other logic elements follow a similar pattern of usage reduction. From here, changing the block size or repeating instantiations of the hardware enables tuning of the frame rate.

A further benefit of BFP quantization is its ability to store more network graphs in DDR. With a greater number of parameters available on-chip, the time and power to change from one graph to another reduces, enabling high-speed switching for different types of inference.

Where accuracy is concerned, BFP has a high dynamic range that is modifiable via the block size. This makes it very adept at retraining, even at very low precisions. A default block size of 32 is sufficient to allow older, larger networks to quantize at *int7bfp* without retraining (specifically ResNet, SqueezeNet, VVG-SSDs, TinyYOLO, UNet and ICNet). On newer, leaner topologies such as the MobileNets and EfficientNet, the resultant drop from quantization can be overcome with a few epochs of retraining. At precisions lower than *int7bfp*, these leaner networks still achieve good accuracy by enable precision-doubling for select critical layers.

The software model has several benefits. In addition to providing a neural network training facility, it allows:

- Testing the effects on accuracy of quantization before implementation
- Retraining to recoup accuracy lost through low precision quantization
- The ability to trial mixed precision configurations and higher precision kernels before building in hardware

All three points save a significant amount of time in speculative bitstream compilation and hardware engineering.

In summary, quantization is an easy way to significantly reduce hardware footprint while maintaining frame rate and keeping accuracy loss to a minimum.

References

For more information about Intel and low-precision inference on FPGAs, the following links are available:

[1] "A Configurable Cloud-Scale DNN Processor for Real-Time AI", https://www.microsoft.com/en-us/research/uploads/prod/2018/06/ISCA18-Brainwave-CameraReady.pdf

[2] "Harnessing Numerical Flexibility for Deep Learning on FPGAs", Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies, https://dl.acm.org/doi/10.1145/3241793.3241794

[3] "Intel Arria 10 Native Fixed Point DSP IP Core User Guide", https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_nfp_dsp.pdf

[4] "Intel Arria 10 Native Floating-Point DSP Intel FPGA IP User Guide", https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug-a10dsp.pdf

[5] "Intel Agilex Variable Precision DSP Blocks User Guide", https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/agilex/ug-ag-dsp.pdf

[6] "Intel® Stratix® 10 NX FPGA", https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/stratix-10-nx-technology-brief.pdf

[7] "Flexibility: FPGAs and CAD in Deep Learning Acceleration", Proceedings of the 2018 International Symposium on Physical Design, https://doi.org/10.1145/3177540.3177561

[8] US Patent application number 16/818889: "Floating-point Decomposition Circuitry with Dynamic Precision", https://uspto.report/patent/app/20200218508

The performance numbers presented herein are a mix of measured and estimated numbers generated using an Arria 10 PAC card at a batch size of 1, incorporating an A10-1150 speed grade 2 FPGA. The host is a Xeon E5-1650 v3 @ 3.5 GHz w/ 132 GB RAM. Some numbers were estimated based on the fmax of the compiled architecture.