Custom precision accelerators for energy-efficient image-to-image transformations in motion picture workflows

Emmet Murphy, Shashwat Khandelwal, Shanker Shreejith[a]

[a]Reconfigurable Computing Systems Lab, Trinity College Dublin, Dublin, Ireland

ABSTRACT

Image to Image (I2I) transformations have been an integral part of video processing workflows with applications in Image Synthesis for Virtual Productions, Segmentation, and Matting, among others. Over the years, deep learning-based approaches have been enabling new methods and tools for automating parts of the processing pipeline, reducing the human effort involved in post-production workflows. These compute-intensive models are often accelerated through on-premise or in-cloud GPU instances to improve the responsiveness and latency while expending large amounts of energy in performing these complex transformations. In this work, we present an approach for optimising the energy efficiency of I2I deep-learning models using quantised neural networks accelerated on a server-style FPGA. We use deep learning-based alpha background matting as the I2I application which is implemented using a U-Net conditional Generative Adversarial Network deep learning model. The model is trained and quantised using Vitis-AI flow from AMD/Xilinx and deployed on a data centre class Alveo U50 FPGA device. Our results show that the quantised model on the FPGA achieves a 1.14× higher throughput for inference acceleration while consuming 11× lower energy consumption per inference when compared to a GPU-accelerated version of the model on a 3080-Ti, while generating nearly identical results with an average IoU > 0.95 across multiple user images at 1080p and 4K resolutions. Additionally, offloads to the FPGA device can be seamlessly integrated into widely used motion picture tools like NUKE with minimal effort. With most cloud providers integrating heterogenous platforms (including FPGAs) into systems, we envision that this work paves the way for more efficient utilisation of custom precision deep-learning models and FPGA acceleration in deep learning-based motion picture workflows.

Keywords: Quantised Deep Learning, Image to Image Transformations, U-Net

1. INTRODUCTION

Motion picture workflows, like many other image-processing tasks, can be represented as a set of tasks that translate an input image (or video) sequence to a corresponding set of output sequences. The specialised nature of individual operators, such as image synthesis or segmentation has traditionally been tackled by often separate and specific algorithms to achieve optimal performance. The generalised nature of the transformation, from input pixel to output pixel, has led to the development and wider adoption of deep-learning-based machines and algorithms in motion picture workflows. Deep-learning tools, such as CopyCat[1] built into Foundry's motion picture software suite Nuke,[2] aid in reducing the amount of human effort in data-intensive tasks in motion picture workflows such as interactive segmentation.[3] With many commercial video production reportedly generating several terabytes of data each day,[4] such reduction can have a cumulative positive impact on the energy consumed over the production phase. Compute-intensive deep-learning models are often accelerated by Graphics Processing Units in video processing workflows, often deployed in server environments specialised for rendering tasks.[5] The use of off-the-shelf GPU accelerators is driven by the software-driven video processing workflow[6] and the native support for GPU offload offered by industry-standard video processing tools such as Foundry's Nuke[2] or Autodesk's FLAME. The massively increasing use of GPUs for existing DL solutions in such workflows is expected to present a significant multiplier growth in energy consumed by post-production tasks.[7]

In contrast, tailoring a circuits datapath to meet the needs of video processing tasks using application-specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs) has shown to provide superior energy

Further author information: (Send correspondence to S. Shreejith)
E-mail: shankers@tcd.ie

efficiency over CPUs and GPUs.[8] Specifically, FPGA-based deep-learning inference accelerators use custom micro-architectures to reduce control (logic) overheads combined with circuit and algorithmic optimisations such as quantisation and pruning to achieve similar inference accuracy at a fraction of the energy consumption.[9] Cloud providers have integrated FPGAs into their hardware resources to allow clients to lower the costs and energy consumption of their deep-learning models using custom-precision accelerators. However, the motion picture industry is yet to widely adopt FPGA-based deep-learning inference accelerators for their workflows to take advantage of the efficiency benefits it offers.

In this paper, we present a quantised deep learning FPGA accelerator and its seamless integration within a motion picture workflow, using Foundry's Nuke as the software platform. We use deep-learning-based alpha matting as the case study and use the pix2pix[10] network, a U-Net based conditional generative adversarial network (C-GAN), as the baseline deep learning model. The main contributions of this paper are

- A quantised pix2pix CGAN network and training flow for accelerating alpha background matting task.
- A network function-based offload from Nuke to the FPGA accelerator that implements the quantised pix2pix network.
- Analysis of the inference performance and energy consumption of the custom precision FPGA-based accelerator in comparison to standard GPU offload using an Nvidia 3080-Ti as the GPU accelerator.

We show that the model can be trained, quantised and optimised using off-the-shelf Pytorch and Vitis-AI tools to run on Xilinx's deep-learning processing unit (DPU) accelerator IP. A datacentre class Alveo U50 device from AMD is used to deploy the model, which talks to Nuke over a (virtual) network interface function, mimicking a network-connected accelerator in a cloud rendering station. Our results show that the quantised pix2pix model deployed on the Alveo U50 generates highly accurate mattes with an average IoU of >0.95 across multiple image sets, achieves $1.14\times$ higher throughput and consumes $11\times$ less energy per inference than the GPU-accelerated pix2pix, while still being invoked from within a native Nuke node. We use the openly available VideoMatte 240K dataset for training, validation and testing our model.[11]

2. RELATED WORKS

The ever-rising amount of image and video data generated and processed by post-production workflows is increasing the energy utilisation and carbon footprint of the media industry.[12] Initial work has been done towards analysing and quantifying the energy consumption of parts of the media processing pipeline, such as encoding and decoding tasks[13,14] and deep-learning inference accelerators in the cloud.[15] Major post-production tasks are executed on large-scale environments such as render farms, that are on-premise (modelled as a specialised data centre) or integrated through cloud service providers. It is estimated that cloud-based workloads have more than tripled between 2015 and 2022, and are dominated by data-driven computations such as deep learning training and inference. Despite the focused efforts of the cloud providers to improve their energy consumption by consistently adopting the most efficient computational resources (CPUs, GPUs, FPGAs and other specialised silicon accelerators) and network infrastructure, their energy usage has increased between 20 to 70% in the same period.[16] The increasing use of deep learning in motion picture workflows could significantly increase this energy consumption if relying on GPU and CPU acceleration.

Motion picture workflows predominantly use software solutions such as Foundry's Nuke[2] with many tools and plugins that can be integrated into a single processing pipeline. Hence, any custom accelerator built for motion picture tools or plugins should seamlessly integrate with such software programs. GPU-based acceleration has thus become the standard scheme for motion picture tools. Nuke's own deep-learning plugin, CopyCat,[1] is able to learn transformations from a small subset of (training) frames and be used to automatically replicate the effect for long sequences, offloading the inference to a GPU if available. In this case, the training task will overfit the model to a singular specific transformation based on a small training set with low variation, with the effect being applied to large amounts of data. Hence, such transformations will consume more energy during the inference (applying the effect) compared to generalised deep learning where the training is performed over a significant chunk of the data. This key distinction makes specialised inference accelerators for motion picture

I2I tasks more relevant. Deep learning inference using custom accelerators on FPGAs has shown to outperform inference on GPUs both in terms of throughput and energy efficiency across many application domains.[17,18] Custom inference engines on FPGAs make extensive use of operator-level optimisations and compact data types, with weights and activations mapped using custom precision representations (8-bits to as low as 1-bit).[17,19] Algorithmic optimisations to training and post-training fine-tuning of quantised deep learning models enable these inference accelerators on FPGAs to have limited impact on classification accuracy on datasets such as ImageNet,[20] MNIST[21] and SVHN.[22] However, the effect of quantisation on I2I transformations is relatively unexplored. Given the recent improvements in deep-learning-based I2I transformations,[23,24] it can be inferred that their application in motion picture tools is likely to increase in the near future.

2.1 Network models for I2I transformation

Convolutional neural network (CNN) continues to be the common workhorse model behind a wide range of applications in the area of image classification and prediction. Within the application setting, the training process causes the CNN to learn to minimise a specific loss function that is related to the transformation to be performed by the function. While designing effective loss functions for classification problems are easier, determining optimal loss function for I2I transformations are non-trivial and requires expertise.[25] Generative adversarial networks (GANs)[26,27] are a class of DNNs that learn to generate new data with similar statistical properties as the training dataset, with the training process minimising the deviation in the properties. By learning and adapting to the training data, GANs can be applied to a number of I2I transformations that would otherwise require specialised loss functions to learn the transformations. Conditional GANs (cGANs)[28] attach additional information to control and direct the data output generated by an unconstrained GAN model to mimic specific transformations more accurately, making them more suited for I2I transformations. Research has shown that conditioning can be performed on class labels,[29] part of data[30] or data from different modalities. When conditioned with images, these models have been used for image prediction,[31] future frame predictions[32] and for generalisable I2I transformations.[10] We build on the pix2pix network[10] in this work to show that I2I transformations in motion picture workflows can be efficiently accelerated on FPGAs and seamlessly integrated with the software tools.

2.2 Image matting

Image matting refers to the process of extracting the alpha matte that segments objects in the foreground from the background. Segmentation is a fast and efficient way to achieve this separation with many notable works such as Mask RCNN,[33] DeepLabV3+[34] and highly accurate interactive segmentation,[3] among others. Matting methods overcome the boundary artefacts generated by segmentation schemes, especially at large resolutions. Trimap-based methods[35–37] rely on manual annotation and learn to extract the alpha matte in the unannotated regions; however, the quality of the trimap impacts the performance of these methods. Approaches to determine alpha matte directly from the image (without a trimap) have also been proposed.[38] Their limited generalisability can be overcome by capturing additional information on the background.[39] This work builds on knowledge of the background image to extract alpha matte of HD and 4K sequences using the pix2pix network as a motion picture task case study.

2.3 Accelerating deep-learning at large scale on FPGAs

FPGA vendors have developed specialised families of products, design tools and integration flows for targeting large data problems that are typically executed in data centre scale settings. The AMD Alveo[40] family of data centre accelerator cards offer logic-dense FPGA fabric that is tightly coupled with high-bandwidth memory (on-chip and off-chip), interface resources (PCIe, 100GbE) and management engines (scheduler), allowing system designers to develop and integrate accelerators for offloading data-intensive tasks within a client-server model. For deep-learning workloads, the Vitis-AI development stack[41] from AMD takes high-level model descriptions in PyTorch or TensorFlow and compiles them to executables that can be run on their deep-learning processing unit (DPU) IP cores, which are specialised engines for accelerating deep learning inference. DPUs can be seamlessly deployed on data centre accelerators like the Alveo with optimisations to adapt to throughput, power consumption and parallelism. Vitis-AI offers both Post-training Quantisation (PTQ) flow as well as a Quantisation-aware Training (QAT) flow to allow designers to optimise their deep learning model for inference on the DPU.[42] While

PTQ is faster and requires only a small number of training samples, the performance of the quantised model is slightly inferior to the QAT flow. Other open-source tools such as FINN[43] and LUTNET[44] offer the ability to design custom datapath accelerators and different quantisation levels (down to 1-bit) for deep learning inference; however, their integration into motion picture workflows require low-level design and optimisation of interfaces and drivers, making them less attractive for seamless integration across multiple tools and software stacks.

In this work, we target a smaller Alveo U50 data centre card as our acceleration platform and utilise the QAT flow to minimise the accuracy loss incurred by quantisation. We show that our quantised pix2pix model on the Alveo U50 is able to generate nearly identical transformations as the ground truth images from the dataset while offering higher throughput and consuming a fraction of the energy when compared to accelerating pix2pix on a GPU. Both models are invoked from within a Nuke graph with the data centre card interfaced over a virtual network interface, mimicking a network-attached accelerator architecture in data centres.

3. SYSTEM ARCHITECTURE

3.1 UNet-Architecture

The U-Net architecture, first published in 2015, was aimed at medical segmentation and generated a class label for each pixel in the image. The key enhancement in the architecture compared to traditional deep-learning architectures was the idea of using feature reduction and expansion, as seen in Fig. 1, which mimics an encoder-decoder pipeline (as shown in Fig. 2), allowing high-resolution features from the encoder to be concatenated with decoder outputs for accurate classification. The authors used the 'intersection over union' (IoU) metric to compare outputs to ground truth labels and achieved a 0.7756 IoU compared to the then state-of-the-art method which achieved 0.46 IoU on the same dataset. Other authors have since refined the U-Net architecture and applied them to other image-based tasks. The authors in [45] analyses over 350 papers between 2017 and 2020 that were published based on the U-Net architecture and its variations like 3D U-Nets, Attention U-Nets and Residual U-Nets, clearly identifying the potential of U-Nets for image segmentation tasks as well as other applications.

The pix2pix U-Net that we adopt in this work is a cGAN network aimed at performing generalisable I2I translations and is released as an open-source repository. Community contributions to this repository have expanded on the original form of the network to add new I2I applications such as style transforms and image augmentation, among others. For this work, we train the pix2pix U-Net on the VideoMatte240K dataset[11] for performing alpha background matting transformation on a series of 1080p and 4K image sequences. The generator and the discriminator used in training are composed using blocks of Convolution-BatchNorm-RelU layers deployed consecutively where the convolutions use 4x4 filters with a stride of 2. Given *Ck* is one such block where *K* denotes the number of filters used, the *encoder* in generator comprises of *8* such blocks with the configuration: *C64-C128-C256-C512-C512-C512-C512-C512*. The decoder comprises of *7* such blocks with the configuration: *C512-C512-C512-C512-C256-C128-C64*. The convolutions in the encoder & decoder of the generator consecutively downsample & upsample by a factor of *2* respectively. After the training process, we discard the discriminator and use the generator for producing the high resolution output. The model is described with standard PyTorch nodes.

3.2 Network Offload Plugin

The *Nuke Machine Learning Plugin* is a built-in tool designed by Foundry to enable the use of custom deep learning architectures, represented in standard frameworks like Pytorch or Tensorflow. The plugin utilises GPU offloads for inference when invoked within a motion picture pipeline within the Nuke processing workflow. For integrating our custom deep learning model and accelerator, we adapt this flow to add support for offloading to an FPGA-based accelerator, utilising a *client to server* TCP network connection. To achieve this, AMD's Vitis-AI runtime (VART) API calls are integrated with the Nuke processing workflow using a server-side plugin to start an instance on the FPGA accelerator using this plugin by the client. This is modelled around Nuke's inbuilt support for GPU offload, allowing Nuke nodes to seamlessly interact with the ML server on the FPGA accelerator, move data between the client and the server and initiate inference tasks on the server. This approach also facilitates rapid integration of newer deep learning models on the server side with limited to no changes to the original Nuke workflow on the client side. Additionally, since the client is not expecting the server to be

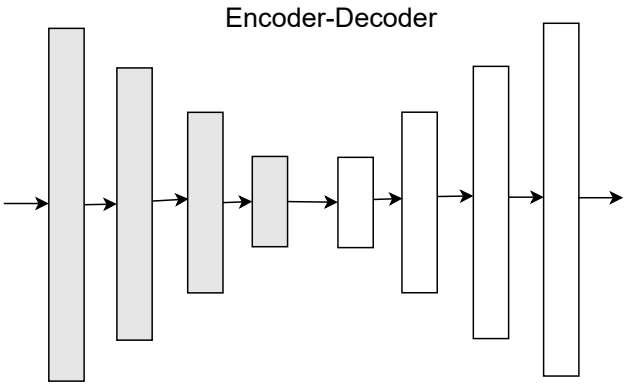

Figure 1. Encoder-Decoder Architecture

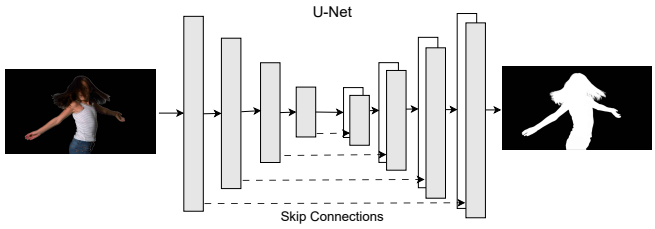

Figure 2. U-Net Architecture

hosted on the same machine, this approach can be easily scaled to interface multiple accelerators simultaneously, if required.

The server side of the application is managed from within a Vitis-AI docker container on the server that hosts the FPGA accelerator. The server application is a generic script that handles a TCP receive connection and a parser to unpack the received data into a NumPy array format for processing by the machine learning accelerator. Individual 'runners' are instantiated on the server side to move data between the server and the DPU and to manage the inference execution over the PCIe interface. Figure 3 shows the overview of the client (Nuke) and server (FPGA) integration architecture.

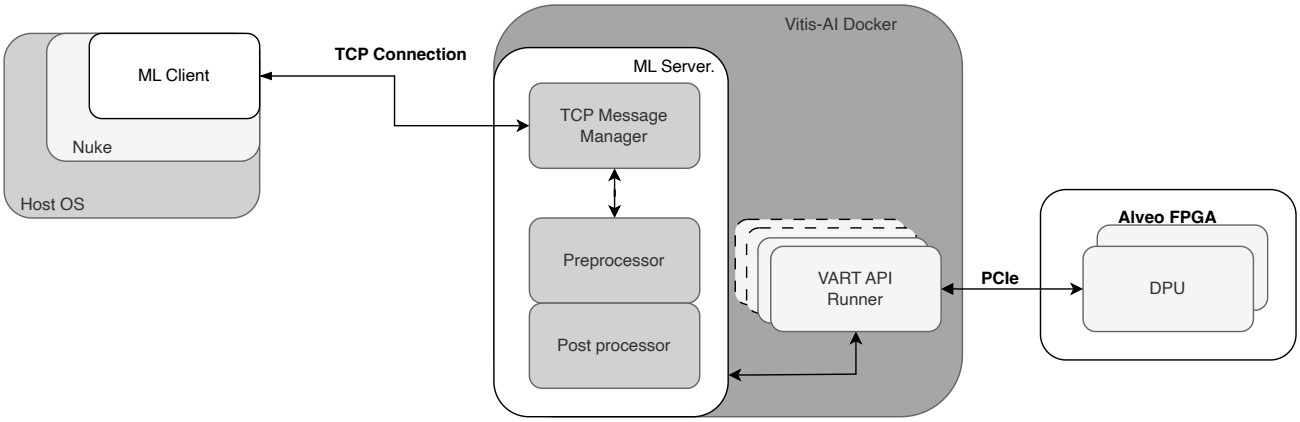

Figure 3. The figure shows the high-level architecture of the client and the server of the proposed network plugin with Nuke.

3.3 Dataset & Training

We use the VideoMatte240K dataset to train the pix2pix U-Net architecture for performing alpha matting and to test its performance.[11] The dataset is extracted from green screen stock footage and provides high-resolution

alpha matte and foreground video clips for 484 users, comprising 240,709 unique frames. The dataset provides 384 clips at 4K resolution while the remaining 100 are in HD (1080p) quality.

We trained & tested the network for *4* users in the dataset. Two users have frames in HD quality while the other two users have frames at 4K resolution. For each user we used 57, 11 & 7 frames for training, validation, and testing respectively, allocating a large section to training for efficient learning. Each frame is split into patches of 256 x 256 (see sec. 3.3.1) leading to a final split of $\sim$ 20000, $\sim$ 4000 & $\sim$ 2500 patches for HD and $\sim$ 34000, $\sim$ 6800 & $\sim$ 4300 patches for 4K resolution users for training, testing & validation respectively. A stride of 64 is used to generate the overlapped patches to prevent the model from learning only the local properties of these patches. During training, the foreground frames are fed as inputs to the model which is required to construct the alpha matte output as shown in the figure 4. Vitis-AI framework provides two both a post-training quantisation flow, (PTQ) where the model is quantised to INT8 precision after training at native precision and a quantisation aware training (QAT) flow, where the model is trained with quantised INT8 precision (for the inference pass). We explored both methods and found that the QAT flow enabled better model generalisation on the dataset. For each user, we train the floating-point model using the standard PyTorch training framework for *10* epochs. We used the *adam* optimizer with a learning rate set to 0.0002 and batch size of 1. The trained model at the end of each epoch is exported as a checkpoint for use with the QAT flow. The optimal checkpoint at the end of the floating point training is subsequently trained using the QAT flow, with 10 iterations within an epoch to fine-tune the weights to reduce the accuracy loss that could be incurred by the quantisation of weights and activations. A slower learning rate of 5×10^{-8} was used for the QAT flow.

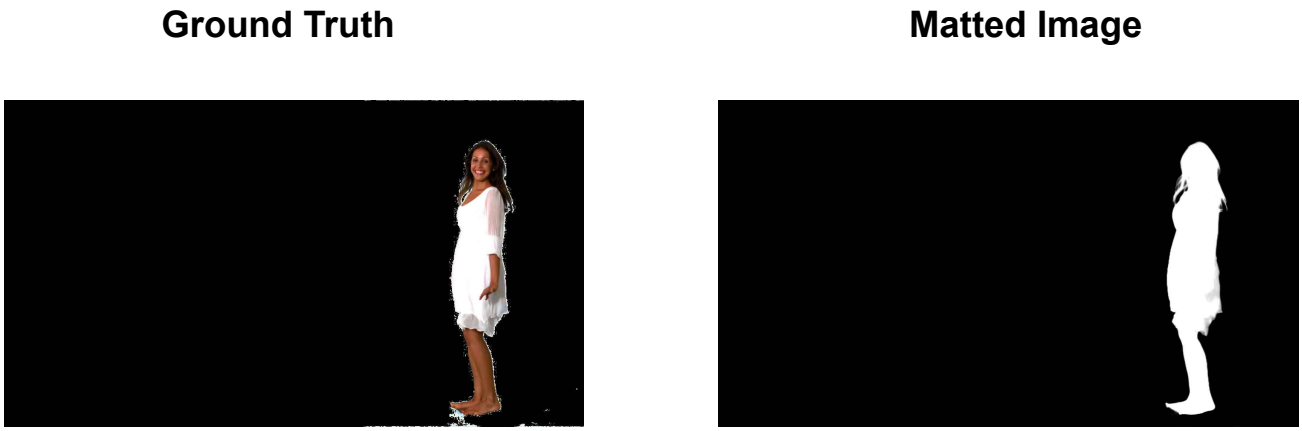

Figure 4. The image shows the input foreground images used to train the CQP2P network to produce alpha matte images.

3.3.1 Data Pre-Processing steps

The original pix2pix network uses image slices of 256x256 as its input. To use the high-resolution images in the dataset, preprocessing is applied these images ahead of time to scale them to the nearest integer multiple of 256 across both dimensions, before unpacking them into 256x256 patches for the pix2pix network. Subsequently, colour space conversion is applied to convert from the linear colour space used internally by Nuke to the sRGB colour space[46] using the prebuilt function in Nuke, using the relation:

$$C_{\text{in sRGB}} = \begin{cases} (1.055 \times C_{\text{in linear}}^{\frac{1}{2.4}}) - 0.055 & C_{\text{in linear}} > 0.0031308 \\ C_{\text{in linear}} \times 12.92 & \text{else} \end{cases}, C \in \{R, G, B\} \quad (1)$$

The colour space conversion is applied to each unpacked patch of the input image, which maps $[0, 1] \longrightarrow [0, 1]$. Subsequently, they are mapped into a fixed point representation using the relation

$$C_{\text{in}} = q_i \times (C_{\text{in sRGB}} - 0.5) \times 2, C \in \{R, G, B\}, q_i \in [1, 2, 4, 8, ..., 128] \quad (2)$$

which maps $[0, 1] \longrightarrow [-q_i, q_i]$. The multiplier q_i represents the quantisation step size in this mapping step. This forms the input patch for training the model using the QAT training flow.

3.3.2 Post-Processing steps

The model generates low-resolution patches corresponding to the input image, which are in the range $[-q_o, q_o]$, with $q_o \in \{1, 2, 4, ..., 128\}$ as the step size for the output's fixed point scale. The low-resolution patches are combined together to form a high-resolution output and transformed back to the floating point representation using the equation

$$C_{\text{out sRGB}} = ((C_{\text{out}}/q_o) + 1)/2, C \in \{R, G, B\} \tag{3}$$

which is in the range $[0, 1]$. Subsequently, the normalised high-resolution image is converted back to the linear colour space by the corresponding transfer function using the relation:

$$C_{\text{out linear}} = \begin{cases} (\frac{C_{\text{out sRGB}}+0.055}{1.055})^{2.4} & C_{\text{out sRGB}} > 0.004045 \\ \frac{C_{\text{out sRGB}}}{12.92} & \text{else} \end{cases}, C \in \{R, G, B\} \tag{4}$$

The colour space converted image is then passed to the TCP manager within the ML server for transferring back to the Nuke node that invoked the inference accelerator.

4. RESULTS

4.1 Training setup

For training, we train the model on the data from 4 unique user video clips of the matting dataset. Starting with a floating point variant of the pix2pix model, we use the training split of the data to train the model at native precision for *10* epochs. The pre-trained model is then exported as a checkpoint for passing to the Vitis-AI's QAT flow. The QAT flow performs a quantisation-aware tuning of the weights and biases to 8-bit precision over 10-15 iterations using the training split of the data, and exports the optimal quantised model as the final checkpoint. The quantised model is subsequently compiled into an *xmodel* executable file containing weights and instructions for the DPU accelerator IP.

4.2 Test setup

For testing the system, we use the smaller AMD U50 data centre accelerator card as our target FPGA, which is hosted on a workstation machine featuring an i7-4770 processor over a PCIe Gen3x16 link. For limiting the variability due to network performance, our Nuke instance is also running on the same machine, creating a virtual TCP connection to the ML server node that interfaces to the Alveo device using VART APIs. The Alveo device is configured with the DPUCAHX8H IP core, which is high throughput configuration of the DPU core optimised for convolutional operations. The DPU is instantiated in the user-configurable region of the U50 with additional logic for managing configurations, instructions and data movement. Two DPU cores are instantiated by the DPUCAHX8H IP core, each having 3 execution units. The default thread execution parameter is left unchanged for our tests.

From a nuke graph, each input/intermediate image is passed to the server and VART APIs for performing inference on the DPU in an asynchronous manner. The CPU task goes to idle as it waits for the processed image to be returned by the server API. However, this flow suffers from a very large initiation interval (II) as the next input is processed only after the complete execution of the first, reducing the overall throughput of the accelerator. Alternatively, passing multiple processing requests to the DPU allows it to optimally schedule the subtasks internally and to utilise its pipelined design to the fullest, significantly reducing the II of the flow, and thus achieving much higher throughput on the accelerator. Note that typical inference acceleration on GPUs (including our setup) utilises a similar scheme to extract higher throughput in the operation.

We report our results based on the matted high-resolution images returned by the quantised model on the Alveo U50. We also quantify the processing latency, initiation interval and energy consumption of the model on the Alveo device. The results are compared against a traditional acceleration flow within Nuke that utilises an Nvidia RTX 3080-Ti GPU hosted on a Ubuntu server with a high-end Intel 11700K CPU running the same version of Nuke tools and client-server interface.

4.3 Performance results

Figure 5 shows the comparison between the ground truth image and the output generated by our quantised model on the Alveo U50 for a set of input images. It can be observed that the outputs generated by the quantised model are visually identical to the ground truth from the matting dataset. To accurately quantify the variations in the generated output against the ground truth, we use the intersection over union metric as they are commonly used in other I2I research results. We also compute the mean squared error between the images, averaged over multiple runs for each user image. The results are shown in table 1, comparing it against the results from the floating point model on the Nvidia 3080-Ti GPU. We observe that the average IoU value across all users is > 0.95 for the quantised model and is comparable to the results obtained by the floating point model, which points to a highly accurate construction of the matted image corresponding to the test data samples in the matting dataset by the quantised model on the FPGA. The mean squared error results also show a close correlation between the images generated by the two platforms, with the quantised model achieving significantly lower MSE in the case of user 2.

Figure 5. Input, ground truth image and the output image from the quantised pix2pix model, when tested on the matting dataset.

Table 1. IoU and MSE scores for different user image sets averaged across multiple runs

Users	Quantised model on FPGA		fp32 model on GPU	
	IoU	**MSE**	**IoU**	**MSE**
User-1	0.981	1.64	0.984	1.83
User-2	0.973	2.6	0.985	3.35
User-3	0.968	2.15	0.99	2.11
User-4	0.989	1.81	0.991	1.65

4.4 Energy efficiency, latency and resources

To quantify the energy efficiency of the FPGA offload, we compare the model against an identical Nuke offload targeting an Nvidia 3080-Ti GPU (GPU accelerated) and the Intel 11700K CPU (GPU not available mode). The active power consumption of each platform is measured and averaged over multiple runs to eliminate outliers. For the Alveo, VART API's built-in utility function is used to report the power draw from the power rails on the device (12V for FPGA and 3.3V for the HBM) at runtime. In the case of the CPU, the Running Average Power Limit (RAPL) interface is used to isolate the CPU power consumption when running inference, while Nvidia tools are used to report the active power consumption on the GPU during inference. For the FPGA accelerator, the latency is measured as the total time from the invocation of the VART runner on the server node to the completion of the inference task on the DPU. In the case of the CPU and GPU inference, the decoding latency is measured as the wall clock execution time of each patch, when invoked from the server interface of the client-server model. The pre-/post-processing phase and the TCP interface delays are not factored into the latency calculations on the CPU, GPU and FPGA. The client-server model is shared by all accelerators with different low-level APIs invoked from the server interface for each platform (VART APIs for Alveo, Nvidia CUDA APIs for the GPU). The pre-/post-processing steps to generate smaller input image patches for the model are also identical for all platforms.

Table 2 shows the average results obtained for inference frames per second, inference latency, initiation interval of the inference task and average energy consumed by the inference task across the CPU, GPU-accelerated model and the quantised model on the FPGA. The results show that the custom quantised pix2pix model on the FPGA is able to achieve a steady state throughput of 394.32 inferences per second, which is 1.14× higher than a GPU offload. This is despite the higher latency (8×) per inference incurred by the FPGA accelerated model due to its heavily pipelined execution stages. However, the quantised model on the U50 consumes 11× lower energy (0.07 J) per inference when compared to the GPU and 131× lower energy than the CPU-based inference. The accelerator consumes nearly 35% of available LUTs, 30% of registers and 58% of DSP blocks on the Alveo U50. The results demonstrate that significant performance and energy efficiency gains can be achieved by offloading complex deep-learning-based motion picture tasks to custom accelerators on FPGAs with very little effort required for integration with existing software tools and workflows.

Table 2. Steady state results on different platforms averaged across multiple executions of a single patch of the input image.

Platform	**FPS**	**Latency (ms)**	**II (ms)**	**Energy (J)**
U-Net CPU	6.8	143.2	147.05	9.20
U-Net GPU	343.6	2.9	2.91	0.79
U-Net FPGA	394.32	23.1	2.5	0.07

5. CONCLUSION

Deep learning is increasingly finding new application avenues in motion picture workflows and GPUs continue to be the de facto platform of choice for accelerating these tasks. In this work, we investigate the feasibility of using

a custom deep-learning accelerator as an alternative platform for I2I transformation tasks and its integration with motion picture tools, using alpha background matting as an example use case, pix2pix cGAN as the deep learning model and Foundry's Nuke as the software platform. We show that the accelerator can be seamlessly interfaced with the Nuke graph using a simple client-server model utilising standard vendor libraries and scripts. The pix2pix model is quantised and trained on the VideoMatte240K dataset using standard PyTorch training flow and AMD's Vitis-AI flow for quantisation, fine-tuning and calibration. The quantised model is deployed as an accelerator attached to a TCP server on an Alveo U50 data centre accelerator platform and compared against Nuke's native deep learning accelerator modes using an Intel 11700K CPU and an Nvidia RTX 3080-Ti GPU. Our results show that the quantised pix2pix model achieves visually identical results to the ground truth with an average IoU of > 0.95 across multiple image sets while achieving $1.14\times$ higher throughput and consuming $11\times$ lower energy per inference than the GPU-accelerated inference. The results show that custom accelerators and their deeper integration into motion picture tools are a promising pathway for the energy-efficient acceleration of deep-learning-based I2I transformations in motion picture workflows.

ACKNOWLEDGMENTS

We would like to acknowledge many fruitful conversations with Dan Ring and Ben Kent from the R&D team at The Foundry, as well as the support of Prof. Anil Kokaram of the Sigmedia Group at the EEE Dept., Trinity College Dublin, for making this publication possible

REFERENCES

[1] The Foundry Visionmongers Limited, "Copycat: bringing machine learning into nuke's toolset." https://www.foundry.com/insights/film-tv/copycat-machine-learning-nuke.

[2] "NUKE — VFX and Film Editing Software." https://www.foundry.com/products/nuke-family/nuke.

[3] Forte, M., Price, B., Cohen, S., Xu, N., and Pitié, F., "Getting to 99% accuracy in interactive segmentation," *arXiv preprint arXiv:2003.07932* (2020).

[4] Blat, J., Evans, A., Agenjo, J., Kim, H., Imre, E., Hilton, A., Tefas, A., Nikolaidis, N., Pitas, I., Polok, L., et al., "IMPART: Big media data processing and analysis for film production," in [*IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*], 1–5, IEEE (2015).

[5] True, T. and Sylvester-Bradley, G., "COTS (Commercial-off-the-Shelf) Platform for Media Production Everywhere," *SMPTE Motion Imaging Journal* **132**(2), 15–25 (2023).

[6] Kernen, T. and True, T., "Tighter NIC/GPU Integration Yields Next-Level Media Processing Performance," *SMPTE Motion Imaging Journal* **132**(1), 59–68 (2023).

[7] Helzle, V., Spielmann, S., and Trottnow, J., "Green screens, green pixels and green shooting," in [*ACM SIGGRAPH 2022 Talks*], 1–2 (2022).

[8] Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., and Jones, P. H., "Comparing energy efficiency of cpu, gpu and fpga implementations for vision kernels," in [*2019 IEEE international conference on embedded software and systems (ICESS)*], IEEE (2019).

[9] Guo, K., Zeng, S., Yu, J., Wang, Y., and Yang, H., "A survey of FPGA-based neural network accelerator," *arXiv preprint arXiv:1712.08934* (2017).

[10] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A., "Image-to-image translation with conditional adversarial networks," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 1125–1134 (2017).

[11] Lin, S., Ryabtsev, A., Sengupta, S., Curless, B., Seitz, S., and Kemelmacher-Shlizerman, I., "Real-time high-resolution background matting," *arXiv* , arXiv–2012 (2020).

[12] Matthews, A., Snelson, T., and Finney, H., "A screen new deal: a route map to sustainable film production," tech. rep., albert (2020).

[13] Ramasubbu, G., Kaup, A., and Herglotz, C., "Modeling the hevc encoding energy using the encoder processing time," in [*2022 IEEE International Conference on Image Processing (ICIP)*], (2022).

[14] Kränzler, M., Wieckowski, A., Ramasubbu, G., Bross, B., Kaup, A., Marpe, D., and Herglotz, C., "Optimized decoding-energy-aware encoding in practical vvc implementations," in [*2022 IEEE International Conference on Image Processing (ICIP)*], (2022).

[15] Hodak, M., Gorkovenko, M., and Dholakia, A., "Towards power efficiency in deep learning on data center hardware," in [*2019 IEEE International Conference on Big Data (Big Data)*], (2019).

[16] International Energy Agency, "Tracking clean energy progress 2023." https://www.iea.org/reports/tracking-clean-energy-progress-2023 (2023).

[17] Li, Y., Liu, Z., Xu, K., Yu, H., and Ren, F., "A gpu-outperforming fpga accelerator architecture for binary convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems, Volume 14, Issue 2* (2018).

[18] Lammie, C., Olsen, A., Carrick, T., and Azghadi, M. R., "Low-power and high-speed deep fpga inference engines for weed classification at the edge.," *IEEE Access* **7** (2019).

[19] Sun, M., Li, Z., Lu, A., Li, Y., Chang, S.-E., Ma, X., Lin, X., and Fang, Z., "Film-qnn: Efficient fpga acceleration of deep neural networks with intra-layer, mixed-precision quantization," in [*Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*], (2022).

[20] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision* **115** (2015).

[21] Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al., "Comparison of classifier methods: a case study in handwritten digit recognition," in [*Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*], **2**, IEEE (1994).

[22] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V., "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082* (2013).

[23] Wang, T., Zhang, T., Zhang, B., Ouyang, H., Chen, D., Chen, Q., and Wen, F., "Pretraining is all you need for image-to-image translation," *arXiv preprint arXiv:2205.12952* (2022).

[24] Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M., "Palette: Image-to-image diffusion models," in [*ACM SIGGRAPH 2022 Conference Proceedings*], (2022).

[25] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A., "Context encoders: Feature learning by inpainting," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 2536–2544 (2016).

[26] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," *Advances in neural information processing systems* **27** (2014).

[27] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A., "Generative adversarial networks: An overview," *IEEE signal processing magazine* **35**(1), 53–65 (2018).

[28] Mirza, M. and Osindero, S., "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784* (2014).

[29] Denton, E. L., Chintala, S., Fergus, R., et al., "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in neural information processing systems* **28** (2015).

[30] Goodfellow, I., Mirza, M., Courville, A., and Bengio, Y., "Multi-prediction deep boltzmann machines," *Advances in Neural Information Processing Systems* **26** (2013).

[31] Wang, X. and Gupta, A., "Generative image modeling using style and structure adversarial networks," in [*European conference on computer vision*], 318–335, Springer (2016).

[32] Mathieu, M., Couprie, C., and LeCun, Y., "Deep multi-scale video prediction beyond mean square error," *arXiv preprint arXiv:1511.05440* (2015).

[33] He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask R-CNN," in [*Proceedings of the IEEE international conference on computer vision*], 2961–2969 (2017).

[34] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H., "Encoder-decoder with atrous separable convolution for semantic image segmentation," in [*Proceedings of the European conference on computer vision (ECCV)*], 801–818 (2018).

[35] Chen, Q., Li, D., and Tang, C.-K., "Knn matting," *IEEE Transactions on pattern analysis and machine intelligence* **35**(9), 2175–2188 (2013).

[36] Chuang, Y.-Y., Curless, B., Salesin, D. H., and Szeliski, R., "A bayesian approach to digital matting," in [*Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*], **2**, II–II, IEEE (2001).

[37] Hou, Q. and Liu, F., "Context-aware image matting for simultaneous foreground and alpha estimation," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision*], 4130–4139 (2019).

[38] Qiao, Y., Liu, Y., Yang, X., Zhou, D., Xu, M., Zhang, Q., and Wei, X., "Attention-guided hierarchical structure aggregation for image matting," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 13676–13685 (2020).

[39] Sengupta, S., Jayaram, V., Curless, B., Seitz, S. M., and Kemelmacher-Shlizerman, I., "Background matting: The world is your green screen," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 2291–2300 (2020).

[40] UG1120, *Alveo Data Center Accelerator Card Platforms User Guide* (2022).

[41] UG1414, *Vitis-AI User Guide* (2023).

[42] Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Van Baalen, M., and Blankevoort, T., "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295* (2021).

[43] Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., and Vissers, K., "Finn: A framework for fast, scalable binarized neural network inference," in [*Proc. Intl. Sym. on Field-Programmable Gate Arrays (FPGA)*], 65–74 (2017).

[44] Wang, E., Davis, J. J., Cheung, P. Y., and Constantinides, G. A., "LUTNet: Rethinking inference in FPGA soft logic," in [*Proc. Intl. Sym. on Field-Programmable Custom Computing Machines (FCCM)*], 26–34, IEEE (2019).

[45] Siddique, N., Paheding, S., Elkin, C. P., and Devabhaktuni, V., "U-net and its variants for medical image segmentation: A review of theory and applications," *Ieee Access* **9**, 82031–82057 (2021).

[46] Commission, I. E. et al., "Multimedia systems and equipment-color measurement and management-part 2-1," *Color management-Default RGB color space-sRGB* (1999).