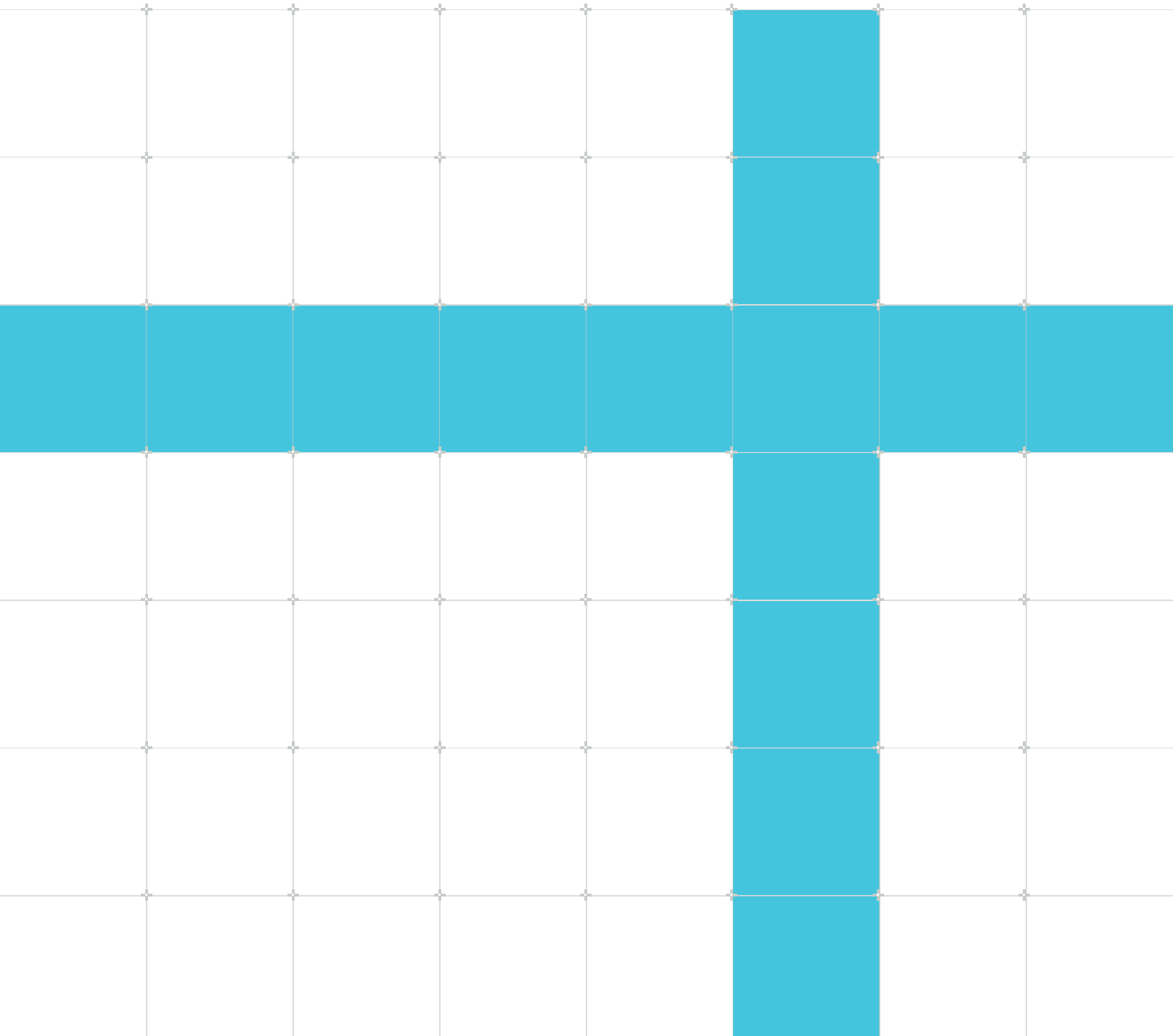
Arm® Neural Super Sampling

Version 1.0

Use Case Guide

Issue 01
111009_0100_01_en



Arm® Neural Super Sampling Use Case Guide

This document (111009_0100_01_en) was issued on 2025-08-12. There might be a later issue at https://developer.arm.com/documentation/111009

The product version is 1.0.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This document is for machine learning engineers, software developers, and system designers who are using the Arm® Neural Super Sampling to create frame upsampling solutions.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Contents

1. Introduction

This document provides a guide to the Neural Super Sampling (NSS) technology developed by Arm. The document outlines the following aspects of the Neural Super Sampling:

- Architecture
- Input and output structure
- Implementation strategies
- Expected performance

This document is intended for developers, engineers, and researchers working on real-time upsampling techniques, particularly on mobile platforms.

1.1 Overview of Neural Super Sampling

Mobile devices increasingly require more complex graphic experience, which increases pressure on GPU performance, power budgets, and memory bandwidth. Traditional rendering techniques are effective, but they are reaching their limits in delivering high-resolution, high-frame-rate graphics within the tight constraints of mobile hardware.

The Neural Super Sampling (NSS) addresses these limitations. The NSS is a temporal frame upsampling technique that improves visual fidelity by reconstructing high-resolution frames from lower-resolution inputs. The NSS is specifically optimized for mobile applications to achieve:

- High image quality
- Computational efficiency
- Minimal bandwidth usage

By integrating neural networks directly into the graphics pipeline, the NSS reduces rendering frames at lower resolution and upsamples the result to the target size. It generates clearer and smoother graphics using significantly less computational power than traditional methods such as full-resolution shading or brute-force anti-aliasing.

Neural Super Sampling is well-suited for integration to both game engines such as Unreal, and standalone graphics applications that implement their own rendering pipeline.

- The current version of NSS is available as an Unreal engine plugin. For more information, see *Arm® Neural Super Sampling in Unreal Engine learning path*.
- For more information about NSS network training, architecture, and inference, see *Arm® How Neural Super Sampling Works*. For more information about Neural Graphics, see *Arm® Neural Graphics Development Kit*.
- For more information about Mobile NSS design, see the *Arm® Mobile Neural Super Sampling* article.

We recommend that you use the latest version of NSS because both image quality and inference performance are continuously improving.

1.2 Background of Neural Super Sampling

The need for the Neural Super Sampling (NSS) is increasing because it addresses the following technical challenges:

- Mobile games now target resolutions beyond 1080p and frame rates of 90–120fps. Memory bandwidth and Thermal Design Power (TDP) have not scaled accordingly; NSS helps to overcome this gap.
- Traditional super sampling techniques such as Temporal Super Sampling (TSS) have limitations in ghosting, flickering, and complex heuristics. These are difficult to generalize or tune across games. The NSS resolves this performance gap.
- Neural graphics are now available on mobile devices. Mobile System-on-Chips (SoCs) increasingly include dedicated Neural Accelerators that can run neural networks.
- The NSS enables intelligent frame reconstruction using compact, low-power models that can run in real-time. Compared to traditional manually coded methods, this helps to achieve better temporal image stability, better image detail with fewer visual artifacts.

1.3 Key features

The Neural Super Sampling has the following key features:

- It enables temporal anti-aliasing-based upsampling.
- It optimises performance for Arm Neural Accelerators.
- It supports integer and non-integer scaling factors.
- It supports quantization for efficient inference on Arm GPUs with Neural Accelerators.

1.4 Notation

This document uses the following abbreviations:

H
: Image height

S
: Image upscaling factor

t
: Time of the current frame

t-1

Time of the previous frame

W

Image width

This document uses data type labels adopted from the Khronos Vulkan standards.

2. Neural Super Sampling architecture

The Neural Super Sampling (NSS) architecture combines classical and machine learning approaches for Temporal Super Sampling. The goal is to create a high-quality image upsampling solution to work efficiently on Arm AI-capable GPU architecture.

2.1 High-level architecture overview

Neural Super Sampling (NSS) is based on a parameter prediction network architecture for temporal super sampling, which is an adaptation of the Kernel Prediction Network (KPN). The system consists of the following main stages:

Pre-processing
: Prepares input data for processing by extracting motion and structural information.

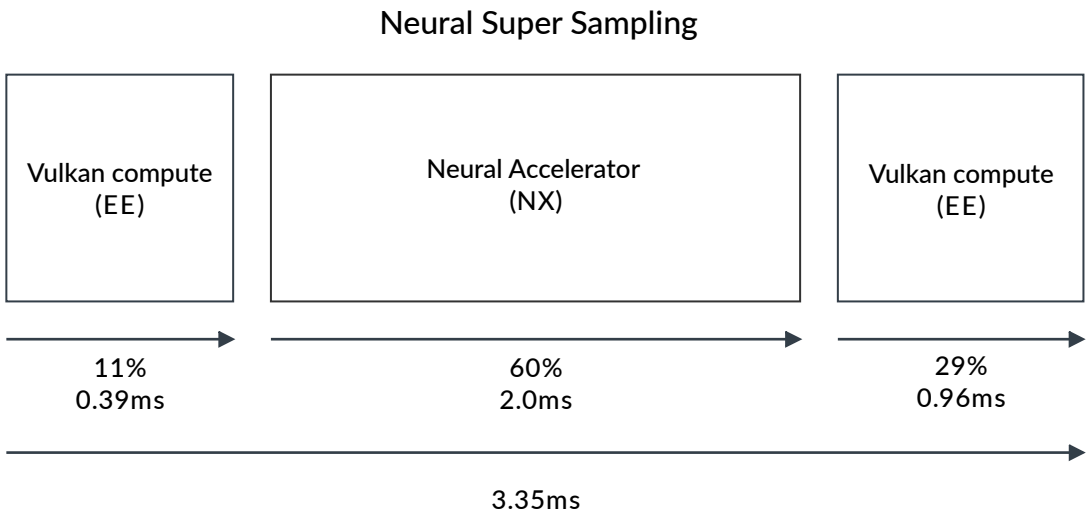
Filter predicting AutoEncoder
: Predicts upsampling parameters as a recurrent neural network.

Post-processing
: Enhances the upsampled frame using the upsampling parameters predicted by the neural network.

The NSS uses the Vulkan Application Programming Interface (API). The pre-processing and post-processing stages are implemented as Vulkan compute shaders. Running a neural network on Neural Accelerator within Arm GPU requires new ML extensions for Vulkan.

The following figure shows the three stages of the NSS architecture. The neural network inference stage is between the pre-processing and post-processing stages which are implemented as compute shaders. The latency numbers in the figure are early estimations for a low-frequency GPU that emulates a thermally limited device.

Figure 2-1: Neural Super Sampling stages



The following table shows more details on early performance predictions for running Neural Super Sampling on a GPU operating in a low frequency sustained mode.

Note

The performance values in the table are indicative only.

Table 2-1: Early performance predictions

Metric	Value
Reference system	Arm GPU with 8 Neural Accelerators and 14 Shader Cores, reference system
Core frequency	430 MHz
Resolution	From 540p to 1080p
Neural network inference time	Approximately 2.0 ms (using Neural Accelerator)
Pre-processing and post-processing time	Approximately 1.35 ms (combined compute shader execution)
Total upscaling time	Approximately 3.35 ms

2.2 Pre-processing stage

The pre-processing stage prepares the input data for the neural network. It uses attributes from the current frame provided by the rest of the rendering pipeline. The following points explain what is happening during the pre-process stage:

- Generates a disocclusion mask and luma derivative to improve temporal stability and to reduce ghosting.
- Downsamples and warps previous predictions to align with the current frame.
- Warps recurrent feedback features to align with the current frame.

2.2.1 Pre-processing stage inputs

The following table lists the pre-processing stage inputs.

Table 2-2: Pre-processing stage inputs

Input	Shape	Data type	Description
Jittered input frame (t)	1 x H x W x 3	R11G11B10	Rendered RGB frame at the current timestep.
Motion vectors (t)	1 x H x W x 2	R16G16 SFLOAT	Rendered motion vectors from the previous frame.
Depth buffer (t)	1 x H x W x 1	R32 FP32	Rendered depth values for occlusion handling.
Luminance derivative (t-1)	1 x H x W x 1	R8G8 UNORM	Luminance derivative from the previous frame to detect flickering areas.
Depth buffer (t-1)	1 x H x W x 1	R32 FP32	Rendered depth values for occlusion handling from the previous frame.
Feedback (t-1)	1 x H x W x 1	R8 SNORM	Recurrent feedback features, previous output from the network.
Depth offset (t-1)	1 x H x W x 2	R8 UINT	Dilated depth coordinates from the previous frame, converted to a position offset.
History (t-1)	1 x SH x SW x 3	R11G11B10	Previous output frame.

2.2.2 Pre-processing stage outputs

The following table lists the pre-processing stage outputs.

Table 2-3: Pre-processing stage outputs

Output	Shape	Data type	Description
Luminance derivative (t)	1 × H × W × 1	R8G8 NORM	Luminance derivative to be used for the next frame.
Intermediate upsampled frame (t)	1 x SH x SW x 3	R11G11B10	Upsampled version of the unjittered input frame to be refined using the output of the neural network.
Depth offset (t)	1 x H x W x 2	R8 UINT	Dilated depth coordinates from the previous frame, converted to a position offset. Passed to post-processing and used for the next frame.
Neural network input tensor	1 x H x W x 12	R8G8B8A8 SNORM	Neural network input data concatenated into a single tensor. INT8 data quantized using asymmetric per tensor quantization.

2.3 Filter predicting autoencoder

The filter predicting autoencoder stage is implemented as a recurrent U-Net-based neural network that leverages the feedback from previous frames.

The filter predicting autoencoder produces Neural Super Sampling (NSS) coefficients that guide the upsampling process in the post-processing stage. The current network version Neural Super Sampling is a refined architecture with increased parameter capacity for improved generalization.

2.3.1 Neural network inputs

The following table lists the neural network inputs.

Table 2-4: Neural network inputs

Input	Shape	Data type	Description
Jittered Current Frame (`t`)	1 × H × W × 3	Quantized INT8 VK_Tensor	RGB input frame
Downsampled Previous Prediction (`y_prev`)	1 × H × W × 3	Quantized INT8 VK_Tensor	Aligned previous frame
Disocclusion Mask	1 × H × W × 1	Quantized INT8 VK_Tensor	Identifies newly exposed areas
Luminance derivative	1 × H × W × 1	Quantized INT8 VK_Tensor	Highlights flickering features
Warped Recurrent Feedback Features	1 × H × W × 4	Quantized INT8 VK_Tensor	Previous output from the network

Neural network inputs are concatenated into a single tensor of size 1 x H x W x 12. The data type is the asymmetrically quantized INT8. In device memory, this tensor is represented as a single region of memory with VK_Tensor memory type.

2.3.2 Neural network outputs

The following table lists the neural network outputs.

Table 2-5: Neural network outputs

Output	Shape	Data type	Description
NSS Coefficients k0 (4 x4 predicted filter - column 0)	1 × H × W × 4	Quantized INT8 VK_Tensor (aliased as an R8G8B8A8 SNORM texture)	Learned weights of a 4x4 kernel that is used to filter the sparsely upsampled frame.
NSS Coefficients k1 (4 x4 predicted filter - column 1)	1 × H × W × 4	Quantized INT8 VK_Tensor (aliased as an R8G8B8A8 SNORM texture)	Learned weights of a 4x4 kernel that is used to filter the sparsely upsampled frame.
NSS Coefficients k2 (4 x4 predicted filter - column 2)	1 × H × W × 4	Quantized INT8 VK_Tensor (aliased as an R8G8B8A8 SNORM texture)	Learned weights of a 4x4 kernel that is used to filter the sparsely upsampled frame.
NSS Coefficients k3 (4 x4 predicted filter - column 3)	1 × H × W × 4	Quantized INT8 VK_Tensor (aliased as an R8G8B8A8 SNORM texture)	Learned weights of a 4x4 kernel that is used to filter the sparsely upsampled frame.
NSS Coefficients (temporal parameters)	1 × H × W × 4	Quantized INT8 VK_Tensor (aliased as an R8G8B8A8 SNORM texture)	Learned temporal upsampling parameters. Passed to the post-processing stage.
Feedback Features	1 × H × W × 4	Quantized INT8 VK_Tensor (aliased as an R8G8B8A8 SNORM texture)	Feedback tensor stored and passed to the input of the next frame for stability.

The neural network output data type is the asymmetrically quantized INT8.

2.4 Post-processing stage

The post-processing stage:

- Applies Neural Super Sampling (NSS) coefficients to refine the output.
- Supports bilinear and Catmull-Rom resampling to balance sharpness and artifact reduction against the desired compute performance.

2.4.1 Post-processing stage inputs

The following table lists the post-processing stage inputs.

Table 2-6: Post-processing stage inputs

Input	Shape	Data type	Description
NSS Coefficients (temporal)	1 × H × W × 4	R8G8B8A8 SNORM	Learned upsampling parameters
NSS Coefficients k0 (4x4 filter - col 0)	1 × H × W × 4	R8G8B8A8 SNORM	Learned weights of a 4x4 filtering kernel
NSS Coefficients k1 (4x4 filter - col 1)	1 × H × W × 4	R8G8B8A8 SNORM	Learned weights of a 4x4 filtering kernel
NSS Coefficients k2 (4x4 filter - col 2)	1 × H × W × 4	R8G8B8A8 SNORM	Learned weights of a 4x4 filtering kernel
NSS Coefficients k3 (4x4 filter - col 3)	1 × H × W × 4	R8G8B8A8 SNORM	Learned weights of a 4x4 filtering kernel
Intermediate upsampled frame (t)	1 × SH × SW × 3	R11G11B10	Intermediate upsampled frame before refinement
Jittered Input Frame (t)	1 × H × W × 3	R11G11B10	Rendered RGB frame at the current timestep
Depth offsets (t)	1 × H × W × 1	R8 UINT	Dilated depth coordinates converted to offset
Motion Vectors (t)	1 × H × W × 2	RG FP16	Rendered motion vectors from the current frame
History (t-1)	1 × SH × SW × 3	R11G11B10	Previous final upsampled frame

2.4.2 Post-processing stage outputs

The following table lists the post-processing stage outputs.

Table 2-7: Post-processing stage outputs

Output	Shape	Data type	Description
Final upsampled frame	1 × SH × SW × 3	R11G11B10	High-resolution output frame

3. Implementation details

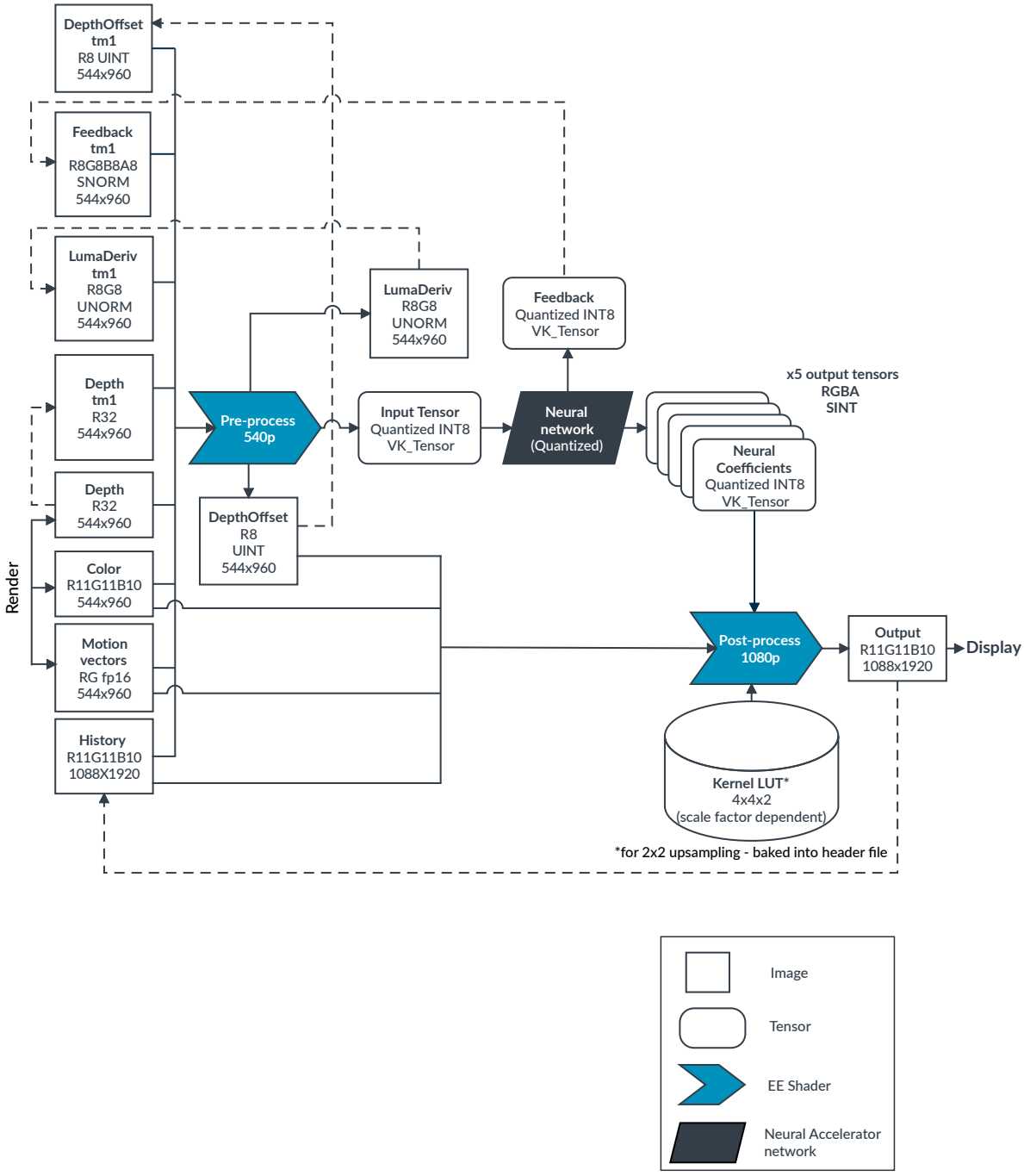
This section shows how the Neural Super Sampling (NSS) algorithm is implemented on Arm GPU with Neural Accelerators.

The neural network is optimized for execution on the Neural Accelerator. It is fully quantized for int8 inference, using the symmetric quantization of weight tensors and assymetric quantization of activations tensors. NSS is designed to be quantization-friendly. No significant differences have been observed in objective image quality measurements after performing quantization.

Both the pre-processing and post-processing steps are implemented as compute shaders. The shaders use fp16 calculations for higher performance. By default, bilinear resampling is used in post-processing. You can enable Catmull-Rom interpolation by defining the HISTORY_CATMULL parameter in the `2_post_process.comp` file. Catmull-Rom interpolation improves image sharpness and reduces the number of visual artifacts with higher computational cost. For more information, see the *Neural Super Sampling model on Hugging Face*.

The following figure shows the detailed Neural Super Sampling architecture. It demonstrates the NSS recurrent nature where the parameters which are computed at the current frame are taken into account at the next frame to improve the fidelity of the overall solution.

Figure 3-1: Neural Super Sampling block diagram



4. Model specification

For the best image quality, you must use the provided Neural Super Sampling (NSS) solution that is described in this section.

4.1 Brief specification

The following table shows the Neural Super Sampling (NSS) high-level specification.

The detailed specification of the inputs and outputs of each processing stage is provided in the High-level architecture overview.


Note

The performance values in the table are indicative only.

Table 4-1: Neural Super Sampling high-level specification

Parameter	Value
Standard upscaling ratio	2 x 2 (for example 540p to 1080p)
Minimum scaling	1 x 1
Maximum scaling	3 x 3
Trained on frame rate	30 Hz

4.2 Detailed constraints

When integrating the Neural Super Sampling (NSS) into the graphics pipeline, you must consider the additional constraints described in this section. Ignoring these constraints can result in low quality output images.

4.2.1 Supporting resolutions

The network has a sequence of three strided convolutions, each performing 2 x 2 downsampling. Therefore, the input to the network must be a multiple of 8 ($2^3 = 8$).

The input sizes that are not a multiple of 8 must be padded. For example, for full HD (1080p) output the 540p input is not a multiple of 8. Therefore, you should pad the 540p input to 544p, which when linearly upscaled by 2 x 2 makes the output 1088p.

4.2.2 Frame rate

The neural network has been trained and tested at a frame rate of 30 Hz. For the inference, a higher frame rate results in a better quality of output.

To allow the motion vectors to work, there are restrictions on the frame rate because of the way the Neural Super Sampling (NSS) algorithm uses the motion vectors. Any effective frame rate is acceptable, but a higher frame rate is better. For extremely low fps, for example 2 fps, we recommend reducing the camera speed, acceleration, or both so that the relative motion between frames mimics the application running at a higher frame rate.

Note

The motion vectors must run at the same frame rate as the application.

4.2.3 Ensuring visual consistency

To ensure the stability of the output image quality of the Neural Super Sampling (NSS) algorithm, you must provide accurate motion vectors to the inputs. To maintain visual consistency, reduce the speed of the camera movement in low fps scenarios.

5. Future enhancements

The following list shows the planned enhancements for future releases:

- Continued refinements to the Neural Super Sampling (NSS) algorithm and the neural network model versions.
- Improvements in image quality and upsampling efficiency.
- Introduction of fine-tuning tools for developer customization, including:
  - Guidance for building a custom training dataset.
  - Optimization for compute performance.

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is for a product under development (dev product).

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0100-01	12 August 2025	Non-Confidential	First non-confidential release for version 1.0

Change history

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in Document release information on page 20.

Table 2: Issue 0100-01

Change	Location
First non-confidential release for version 1.0	-

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

Caution

We recommend the following. If you do not follow these recommendations your system might not work.

Warning

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or harming yourself.

This information is important and needs your attention.

A useful tip that might make it easier, better or faster to perform a task.

A reminder of something important that relates to the information you are reading.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm® How Neural Super Sampling Works	How Neural Super Sampling Works	Non-Confidential
Arm® Mobile Neural Super Sampling	Mobile Neural Super Sampling	Non-Confidential
Arm® Neural Graphics Development Kit	Neural Graphics Development Kit	Non-Confidential
Arm® Neural Super Sampling in Unreal Engine learning path	Neural Super Sampling in Unreal Engine learning path	Non-Confidential
Arm® Neural Super Sampling model on Hugging Face	Neural Super Sampling model on Hugging Face	Non-Confidential