Pixelmania

Table of Contents

Preamble

1.1 Disclaimer

This document and the software is a work in progress, Pixelmania reserves the right to update and change the procedures and processes described in this documentation at any time. We will try our best to inform you when these changes happen with the hopes of minimising any impact on production.

1.2 Copyright

In addition to those names set forth on this page, the names of other actual companies and products mentioned in this User Guide (including and not limited to, those set forth below) may be the trademarks or service marks, or registered trademarks or service marks, of their respective owners in the United States and/or other countries, states and/or provinces. No association with any company or product is intended or inferred by the mention of its name in this User Guide.

Nuke is a trademark of The Foundry Visionmongers Ltd.
Linux is a registered trademark of Linus Torvalds.
CentOS is a trademark of Red Hat, Inc.
Rocky Linux is a trademark of Rocky Enterprise Software Foundation (RESF).
"Microsoft (R) Windows (R)" is a registered trademark of Microsoft Corporation in the United States and other countries.
Reprise License Manager is a trademark of Reprise Software, Inc.
Vimeo is a trademark of Vimeo, Inc.

1.3 Acknowledgements and citations

We like to thank the authors of the following research work and publications:

```
@InProceedings{wang2018esrgan,
    author = {Wang, Xintao and Yu, Ke and Wu, Shixiang and Gu, Jinjin and Liu, Yihao and Dong,
              Chao and Qiao, Yu and Loy, Chen Change},
    title = {ESRGAN: Enhanced super-resolution generative adversarial networks},
    booktitle = {The European Conference on Computer Vision Workshops (ECCVW)},
    month = {September},
    year = {2018}
}

@misc{wang2020basicsr,
  author = {Xintao Wang and Ke Yu and Kelvin C.K. Chan and Chao Dong and Chen Change Loy},
  title = {BasicSR},
  howpublished = {\url{https://github.com/xinntao/BasicSR}},
  year = {2020}
}

@article{tecogan2020,
  title = {Learning temporal coherence via self-supervision for GAN-based video generation},
  author = {Chu, Mengyu and Xie, You and Mayer, Jonas and Leal-Taix{\'e}, Laura and Thuerey, Nils},
  journal = {ACM Transactions on Graphics (TOG)},
  volume = {39},
  number = {4},
  pages = {75--1},
  year = {2020},
  publisher = {ACM New York, NY, USA}
}
```

2. Installation

2.1 System Requirements

- 64 bit Linux Operating System (for example CentOS 7.x)*, or
- 64 bit Windows Operating System (for example Windows 10 Pro)
- Nuke 11.3 or higher (Linux), or Nuke 12.0 or higher (Windows), with a valid license. Nuke Indie is supported from NNSuperResolution v2.5.0 and above (to run the plugin in Nuke Indie, you have to use the latest version of Nuke13.x, Nuke14.x or Nuke15.x).

* It is possible to use NNSuperResolution on Rocky8.x but you might need to set a certain environment variable **NUKE_ALLOCATOR=TBB** for it to work with the current builds (we are currently building all our plugins on CentOS7). Please see the FAQ, section 4 in this document, to read more about Rocky 8 compatibility.

2.2 Plugin Installation

NNSuperResolution relies on the standard Nuke process for finding and using plug-ins. Hence you only need to unzip the files in the downloaded zip archive (that matches your OS, Nuke version and CUDA version) to a directory of your own choice.
If you are using Windows, you do need to set an additional environment variable, please see the info below.

You then append that full path to the **NUKE_PATH** environment variable. This will tell Nuke to load the supplied **init.py**, **menu.py** and **NNSuperResolution.so** (**NNSuperResolution.dll** on Windows) files at startup. To read more about the **NUKE_PATH** environment variable, visit the documentation page by Foundry:
https://learn.foundry.com/nuke/content/comp_environment/configuring_nuke/defining_nuke_plugin_path.html

Linux:
The simplest way to create this environment variable is to put a single line like this in your **.bashrc** file in your user's home folder:
export NUKE_PATH=/full/path/to/plugin/install/folder

Windows:
In Windows 10, you have to add the environment variable using the system preferences. The easiest way to get to the right place is using the search bar on the Start Menu. Search for "environment variables", and choose "**Edit the system environment variables**". Add the "**NUKE_PATH**" variable and set the value to the full path to the folder where the plugin is installed.

For a more detailed guide of how to do this, you can search on Google for "windows 10 set environment variable permanently", and you will get links to pages like:
https://www.architectryan.com/2018/08/31/how-to-change-environment-variables-on-windows-10/
https://www.opentechguides.com/how-to/article/windows-10/113/windows-10-set-path.html

Important! In Windows, you also have to set the **PATH** environment variable to the full path of the folder where the plugin is installed (in addition to the NUKE_PATH environment variable already mentioned). This is needed for the plugin to be able to find the additional provided libraries that it is dependent upon (in Linux, this is handled automatically by the system). You shall not replace your current PATH environment variable set by the system, but make sure to append it to the list of paths already there. The different paths in the list are separated by semi-colons (;).

An alternative way to using the **NUKE_PATH** environment variable to specify for Nuke where to look for plugins, is to use the python command:
nuke.pluginAddPath("/full/path/to/plugin/install/folder")
This can be added to your **init.py** file located in your user's "**.nuke**" folder. Read more at:

https://learn.foundry.com/nuke/developers/13.0/pythonreference/_autosummary/nuke.pluginAddPath.html

2.3 Environment Variables

PIXELMANIA_NNSUPERRESOLUTION_ABORT_ON_LICENSE_FAIL
This environment variable is deprecated/removed since NNSuperResolution v.3.3.0. This is because we don't allow batch rendering in demo mode anymore, i.e. when the node doesn't find a license. Basically if you are running the node without any license to try it out, it will only work in Nuke's GUI mode (and will be watermarked with noise). If you want to make test renders of sequences, it's possible by rendering locally using GUI Nuke. If you want to test out the plugin without the watermark noise we have free trial licenses available by request from our website.

PIXELMANIA_NNSUPERRESOLUTION_DONT_RENDER_USING_GUI_LICENSES
(This environment variable is new as of NNSuperResolution v3.3.0.)
The default behaviour of the node in batch/render mode is to look for dedicated NNSuperResolution render licenses. If it can't find a render license, it will instead look for a GUI license. If it finds a GUI license, it will use it to render the node's output. This behaviour is not preferred if you only got a few GUI licenses available, i.e. you don't want the render farm to steal the artists' GUI licenses. To enforce the node to not use GUI licenses, please set the environment variable PIXELMANIA_NNSUPERRESOLUTION_DONT_RENDER_USING_GUI_LICENSES to the value of "1".
Please note that if you already got a site license, you don't want to set this environment variable since you already got an unlimited number of floating GUI licenses, so it's fine for the node to use those for rendering as well as in GUI mode.

PIXELMANIA_SUPPRESS_KERNEL_WARNINGS
(This environment variable is new as of NNSuperResolution v3.4.0.)
As of NNSuperResolution v3.4.0, the node will print out a warning during creation of the node if it's being dependent on JIT-compiled processing kernels. Basically this is happening if you have a very modern GPU that is using a compute capability that is higher than the native compute capability support of the specific build of NNSuperResolution that you are running. The warning is there to inform you (the first time you run the plugin) that the kernels will get JIT-compiled which will take about half an hour (and appear like the computer is just hanged, even if it's heavily processing). The node will also check the environment variable CUDA_CACHE_MAXSIZE to see that it's set with a good value that will make it possible for the JIT-compilation to succeed (please see below for more info about this).
With all the above in mind, the new environment variable PIXELMANIA_SUPPRESS_KERNEL_WARNINGS can turn these warning messages off if it's set to the value of "1".

CUDA_CACHE_MAXSIZE
If you are using one of the CUDA10.1 versions of the NNSuperResolution plugin (as an example), and you are having a modern compute capability 8.0 or 8.6 GPU, i.e. the Ampere generation of cards (for example RTX3080Ti), you need to set the CUDA CACHE

for JIT-compiled kernels to a high value using the **CUDA_CACHE_MAXSIZE environment variable**. We recommend setting it to **4000000000** because that is close to the maximum limit of the setting (4Gb).
You need to do this because you are using an older version of CUDA that was released before the 8.0/8.6 cards were released. You will then rely on JIT compiling the internal PTX code instead of using the natively compiled kernels for your GPU architecture. These kernels will only need to be compiled once, and will be saved to the CUDA CACHE in your user's home directory. But for this operation to succeed the cache size needs to be set higher than the default value.
The kernel compilation usually takes around 30 minutes or so, and then you are set to use the plugin without needing to do this again. Be aware of this need though, since the process is silent and might be experienced as a strange hanging without any progress bars and such.

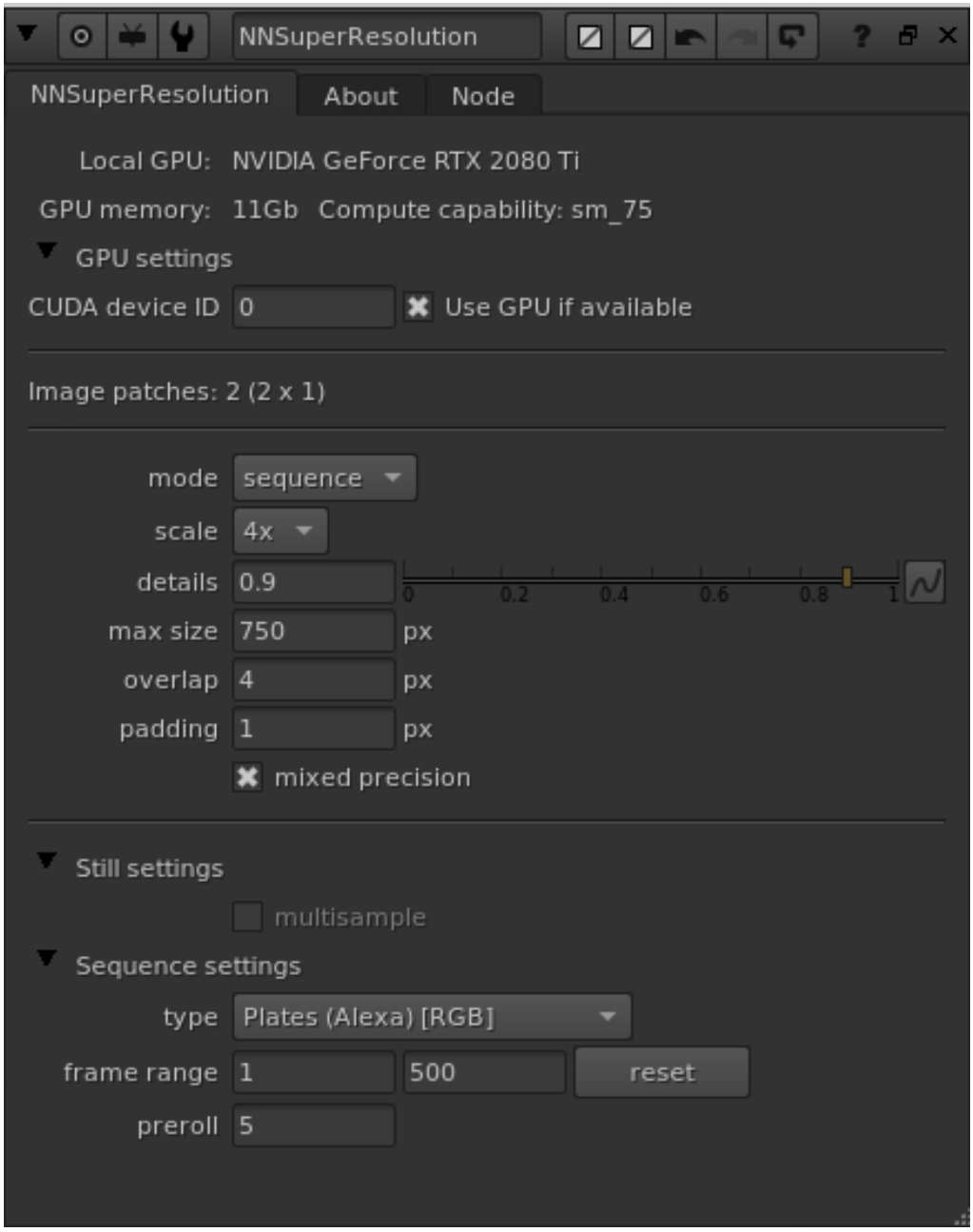
If you don't want to do all this, you should download and use one of the CUDA11.1/ CUDA11.2/CUDA11.8 compiled versions of NNSuperResolution instead. These will run directly and potentially even faster on your modern GPU.
The only downside then is if you are using Nuke13.x/Nuke14.x and Foundry's own machine learning toolset like the CopyCat node. Nuke 13.x is only shipped with support for CUDA10.1/cuDNN8.0.5, Nuke 14.0 is only shipped with support for CUDA11.1/ cuDNN8.4.1 and Nuke 14.1 is only shipped with support for CUDA11.8/cuDNN8.4.1. If you are using NNSuperResolution with another version of CUDA/cuDNN you can not run them in the same script without problems/crashing.

When it comes to the newly released Nuke 14.1, we are only shipping NNSuperResolution compiled against the exact same CUDA version and cuDNN version as Nuke is using internally. This is because Foundry have updated to a CUDA version that natively supports the RTX40xx series of NVIDIA GPUs. NNSuperResolution for Nuke 14.1 is hence using CUDA 11.8 and cuDNN 8.4.1 to fully be compatible with Nuke's own AIR tools.

3. Usage and Knob Reference

3.1 Usage of the Node

The algorithm that calculates the high resolution output using the lower resolution input material is pre-trained on lots and lots of images/sequences during the creation of the software. The plugin is capable of producing a 2x or a 4x upscale of the input material. The scale factor is controlled by a user knob called "scale". To produce the upscaled result, you simply feed it a RGB(A) image/sequence to the only input of the node. The node supports processing of any RGB layer, which means that you can feed multiple layers of RGB material through it at the same time, and write it out using a standard Write node set to "channels: all".
As of v2.5.0 it is also possible to process CG with alpha channels (RGBA) using sequence mode. You have to set the "type" knob to either "CG" or "Hybrid" to get the alpha processed together with the RGB channels. Please note that you have to have a pre-multiplied type of alpha channel to get the expected output. If you do have pixels that represent light (with no matching alpha), such as light glow, you get better results if that is upscaled separately using the "plates" options instead of "CG".
As of v3.0.0 you have a new user knob called "details". This knob controls the amount of generated sharp detail by blending between two differently trained neural networks, one softer PSNR loss based networks and one sharper GAN loss based network. The setting



0.0 uses the PSNR loss based network only, 1.0 uses the GAN loss based network only, while all values in between is a blend of the two. Older versions of NNSuperResolution (pre v3.0.0) always used solely the GAN loss based network. While this is producing superior results overall, it does quite often generate over-sharpened local results. To give users the ability and flexibility to handle these cases, the details knob was introduced.

3.2 Knob Reference

The knobs that do exist on the plugin to tweak by the artist are mostly related to being able to scale up pretty large resolution images/sequences using limited VRAM on the graphics card. The neural network requires pretty large amounts of memory even for small resolution input images/sequences. To be able to scale up, for example, a 1K input to 4K the image needs to be split up into several passes/patches to fit into memory. This is all done and handled transparently “under the hood” so the artist can focus on more important things. You might need to tweak the settings though depending on the use case and

available hardware on your workstation. For a video demo of the available knobs (as of v2.5.0), please watch: https://www.youtube.com/watch?v=Oe1ilZCP_gY

mode
The default mode is still mode. This is best for upscaling still photos, textures, patches and similar material. When you are processing sequences, i.e. filmed video material, you want to change the mode knob to “sequence”. This activates a different way of processing than when in still mode. While the sequence mode doesn’t produce as detailed and sharp still frames as the still mode, it does create temporally stable results instead. This is hugely beneficial when you got moving material, for example when doing VFX work.

scale
Choose between an upscale factor of 2x or 4x.

details
This knob controls the amount of generated sharp detail by blending between two differently trained neural networks, one softer PSNR loss based networks and one sharper GAN loss based network. The setting 0.0 uses the PSNR loss based network only, 1.0 uses the GAN loss based network only, while all values in between is a blend of the two. This setting does not impact performance, it rather pre-computes what exact neural network to use when later performing the upscale processing.

max size
Max size sets the maximum dimension, in one direction, that an image patch can have and is the single most important knob on the plugin. The default is 500, which means that the max dimensions an input patch would be allowed to be is 500×500 pixels. From our experience that will use up to around 8Gb of VRAM on your graphics card. If you haven’t got that available and free the processing will error out with a CUDA memory error, and the node in Nuke will error in the DAG. To remedy this, and also to be able to input resolutions much higher than a single patch size, you can tweak the max size knob to adapt to your situation. You can lower it to adapt to having much less VRAM available. The plugin will split the input image into lots of smaller patches and stitch them together in the background. This will of course be slower, but it will make it possible to still run and produce much larger results. There is a text status knob above the “max size” knob (in between the two dividers), that will let you know how many image patches the plugin will run to create the final upscaled image.

overlap
Since the plugin will run multiple patches through the neural network, there will be lots of edges of image patches present. The edges doesn’t get as satisfying results as a bit further into an image from the edge. Because of this, all these multiple patches that get processed are done so with some spatial overlap. The overlap knob value sets the number of pixels (in the source image resolution space) of the patches’ overlap. The default of 12, which in the resulting high res image is 48 pixels (when upscaling 4x), is usually a good value.

padding
The padding is very connected to the overlap above. While the overlap sets the total amount of pixels the patches overlap, the padding then reduces the actual cross fading area where the patches are blended to not use the very edge pixels at all. The padding is

also specified in pixels in the source resolution, so the default value of 1 actually means that the 4 edge pixels in the high res result of each patch will be thrown away (when upscaling 4x). This way of blending all patches together has proven very successful in our own testing.

mixed precision
This is a new knob since v3.2.0. When this is “on" (which is default) the plugin will try to use mixed precision on the GPU for calculations. What this means is that for certain supported calculations, the GPU will process data using half floats (16 bit) instead of normal full floats (32 bit). This results in a bit less VRAM usage and faster processing, for the little cost of a slightly less accurate result. You usually won’t notice the difference in quality at all, but the resulting processing speed can be, for example, 15% faster. The actual speed difference depends on your exact GPU model.
If you are getting CUDA processing errors, please try to process the same material but turning off mixed precision.

Still mode: multisample
The default value of the multisample knob is “off” which means that each image patch is run through the upscale algorithm once. This usually produces nice and sharp still images. Sometimes it’s more beneficial to get a bit less sharp but smoother results instead. When multisample is “on” it will instead run each image patch through the upscale algorithm 4 times but first pre-process the patch by orienting it differently (using rotations and mirrors). This will of course make the process much slower, but it will in some situations be worth it.

Sequence mode: type
Using the type knob, you can specify what type of material you want to process. If you want to process filmed material, you can choose between “Plates (Alexa) [RGB]” and “Plates (Vimeo) [RGB]” (this option was in earlier versions called “Plates (legacy) [RGB]”). We recommend the “Plates (Alexa) [RGB]” option since that is using a newer and better performing neural network that was released in v.2.5.0, producing sharper and better detailed results. This has been made possible by retraining the whole neural network solution using a lot more, and a lot higher quality, real filmed plates shot using the Arri Alexa motion picture camera.
If you want to process computer generated images, i.e. rendered 3D graphics with alpha channel support, you can choose between “CG, [RGBA]” and “Hybrid (Alexa+CG), [RGBA]”. The first option is using our trained CG neural network for all channels. The second “hybrid” option is using the “Plates (Alexa)” solution mentioned above for the RGB channels, while the A channel is produced using the CG trained neural network solution.

Sequence mode: frame range
When you are using sequence mode, it’s very important to set the frame range of the input material correctly. Since the algorithm need to gather neighbouring frames it needs to know the extend of the material it can use to best be able to produce a good result. If you are trying to view/process a frame outside of the specified frame range it will render black.

Sequence mode: preroll
The preroll is specifying how many frames before the current frame that is used to produce the current frame’s upscaled result. Since the algorithm is temporal it is a bit more complex than that. Basically the plugin will use the number of preroll frames if you jump straight into a random frame in a sequence. It will then need to run the whole upscale process for all

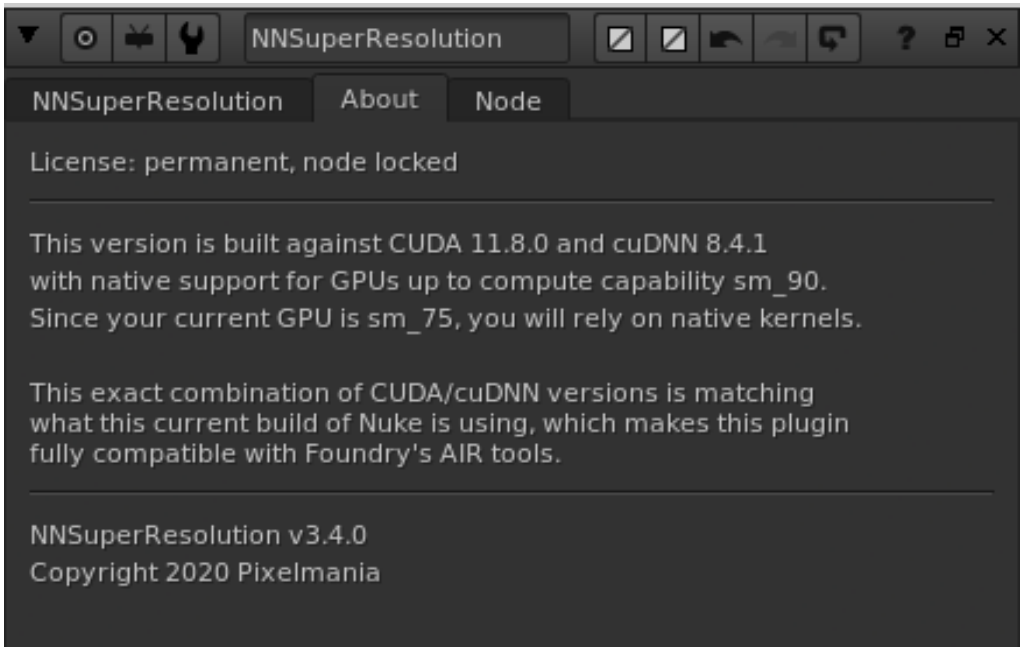
the preroll frames before it can produce the current high res frame. This is how it's making sure the result is highly detailed and also temporally stable. Doing this takes a lot of processing power and is not something we want to do for every frame. To be efficient the current frame's high res result is cached internally in the plugin. So if you then step to the next frame in the timeline it will use the cache and will directly be able to process the frame.
To put it short and clear: The plugin will need to process the preroll amount of frames before the current one if the previous frame hasn't been processed just before (and hence resides in the cache). Because of this the first frame will take much longer to process, but if you then step to the next frame one at a time the processing will be faster. Because of this, you do want to keep a pretty high batch count (frame range of a processing chunk) when you upscale material using the plugin in a farm environment. If you are rendering locally it will just work as long as you are rendering all consecutive frames in a sequence.

CUDA device ID
This is a new knob since v3.2.0. This knob specifies what GPU to use in the system by its CUDA device ID. It's only relevant if you got multiple GPUs installed in the system. The default value is 0, which is the default CUDA processing device, usually the fastest/most modern GPU installed. Please refer to the output of running the command "nvidia-smi" in a terminal for retrieving the info of the specific GPU device IDs you have assigned to your GPUs in your particular system.

Use GPU if available
There is another knob at the top of the plugin called "Use GPU if available", and it's "on" by default (recommended). This knob is only present if you've installed a GPU version of the plugin. This knob is not changing the upscale behaviour, but rather how the result is calculated. If it is "on" the algorithm will run on the GPU hardware of the workstation, and if it's "off" the algorithm will run on the normal CPU of the workstation. If the plugin can't detect a CUDA compatible GPU, this knob will automatically be disabled/greyed out. This knob is similar to the one you'll find in Foundry's own GPU accelerated plugins like for example the ZDefocus node.

We highly recommend always having this knob "on", since the algorithm will run A LOT faster on the GPU than on the CPU. To give you an idea of the difference, we've seen calculation times of the same input image be around 10 secs using the GPU and about 6 minutes(!) on the CPU. These numbers are just a simple example to show the vastly different processing times you will get using the GPU vs. the CPU. For speed references of processing, please download and test run the plugin on your own hardware/system.

The "About" tab:
There is an additional tab called "About" in the knob panel for NNSuperResolution. This tab prints out some additional info that can be very helpful for trouble shooting. The first section is printing out what type of license that is currently in use, and also how many more days it's valid for.
The next section (if you are running a GPU based build) is printing out what CUDA and cuDNN this version is built against. It also lists what compute capability your current GPU is supporting and what max compute capability the current build of the plugin is natively supporting. If also writes out if your current configuration will use native or JIT-compiled kernels. In addition to that, it also writes out if the current plugin build is matching what



versions of CUDA and cuDNN that the current Nuke build is using, and hence if it's compatible with the AIR tools or not.
Finally it's listing what version of the plugin that is running.
(the screenshot above is just a snapshot of how this info can look, captured on one of our development workstations)

4. Frequently Asked Questions

Is Nuke Indie supported?
Yes, Nuke Indie is supported from v.2.5.0 and above. To have the plugin working in Nuke Indie you have to use the latest version available from Foundry of Nuke 13.x, Nuke 14.x or Nuke 15.x.

How are licenses consumed?
All of Pixelmania's licenses are per host, i.e. you can have multiple jobs using NNSuperResolution on the same host and only use a single license. If you query the license server it may report multiple handles for those jobs, but it's still only using a single license token.
As of NNSuperResolution v3.3.0 we do support dedicated render licenses as well. The default behaviour when rendering using Nuke's batch mode (i.e. command line rendering), is to check out render licenses. If it can't find a render license, it will instead try and check out a GUI license. This behaviour is preferred if you got a node locked license, or if you got a site license. It is not preferred if you have only a few GUI licenses and then a bunch of render licenses. To stop the node from using GUI licenses when rendering, you can set the environment variable PIXELMANIA_NNSUPERRESOLUTION_DONT_RENDER_USING_GUI_LICENSES to the value of "1" (please see more about this in the section "2.3 Environment Variables").

Do you offer educational discount?
As of now, we are not offering any default educational discount. We've chosen to sell our licenses for a low price instead to accommodate most people and companies to afford a license anyway. But you are of course always welcome to contact us directly and state your case.

Am I able to transfer my NNSuperResolution license to another machine?
Yes, please fill out the License Transfer Form available at this address:
https://www.pixelmania.se/license-transfer-form
The form should be filled in, signed, scanned and emailed to licenses@pixelmania.se, and we will send you a new license file as soon as we can.

Do you really need CUDA installed to be able to use the GPU?
You don't need to have the CUDA Toolkit installed to use our plugins and have them GPU accelerated. All of our plugin downloads come bundled with the needed CUDA and cuDNN libraries from NVIDIA. What you need to have them work correctly is a supported GPU (please see below) and a modern enough NVIDIA graphics driver version installed.

You can download the latest graphics drivers from NVIDIA's website here:
https://www.nvidia.com/Download/index.aspx

What NVIDIA graphic card architectures are supported?
The currently supported CUDA architectures are the following compute capabilities (see https://en.wikipedia.org/wiki/CUDA):

- 3.5 (Kepler)
- 5.0 (Maxwell)
- 5.2 (Maxwell)
- 6.0 (Pascal)
- 6.1 (Pascal)
- 7.0 (Volta)
- 7.5 (Turing)
- 8.6 (Ampere)*
- 8.9 (Ada Lovelace)**

* The Ampere architecture, i.e. the NVIDIA RTX30xx series of cards, works with all CUDA variants of NNSuperResolution (**CUDA10.1, CUDA11.1, CUDA11.2, CUDA11.8)**. The CUDA11.1/CUDA11.2/CUDA11.8 versions got native support and will work directly. The CUDA10.1 versions will need to compile the CUDA kernels from the PTX code. Please see more info above in the environment variables section.

* The Ada Lovelace architecture, i.e. the NVIDIA RTX40xx series of cards, works with all CUDA variants of NNSuperResolution (**CUDA10.1, CUDA11.1, CUDA11.2, CUDA11.8)**. The CUDA11.8 versions got native support and will work directly. The CUDA10.1/ CUDA11.1/CUDA11.2 versions will need to compile the CUDA kernels from the PTX code. Please see more info above in the environment variables section.

I'm getting crashes using NNSuperResolution on Rocky8.x
We have noticed, and also received reports about, NNSuperResolution crashing when being used on Rocky8.x in Nuke13.2v2 or later. After thorough investigation, and also discussions with Foundry, we've found that it is related to the internal memory system used in Nuke. Foundry changed that from "Threading Building Blocks" (TBB) back to the default system way in Nuke13.2v2. This seems to not work correctly with our builds created on CentOS7. There is a workaround available though, and that is to set an environment variable that forces Nuke to still use the TBB memory handler:
NUKE_ALLOCATOR=TBB
With this environment variable set, everything seems to be working fine, at least according to our own tests.

We are now compiling the Nuke15.0 build of NNSuperResolution in Rocky8. Please let us know if you are experiencing any strange behaviour. Please report any bugs directly using the following email address: contact@pixelmania.se

Why is the plugin so large?
This is a result of including a lot of needed static libraries for neural network processing, and also for supporting lots of different graphic cards for acceleration. The plugin could have been much smaller if all these pieces of software used were required to be installed as dependencies on the system instead. That would kind of defeat the nice ecosystem of having self contained plugins that work simply by themselves, hence the plugin need to be a rather large file.

I'm having problems running NNSuperResolution in the same Nuke environment as KeenTools FaceTracker/FaceBuilder
We have noticed that trying to load both NNSuperResolution and FaceTracker in the same Nuke environment makes the plugins clash with each other. The error you get is that NNSuperResolution is not finding the GOMP_4.0 symbols in the libgomp.so shared library that is loaded. This is because both NNSuperResolution and FaceTracker are using the libgomp.so shared library, but KeenTools have decided to ship their plugin with a version of this library. This specific library version they are shipping is a rather old version of the library. So what happens is that NNSuperResolution is trying to use the version from FaceTracker but it's expecting a newer version, and it's failing. The fix we have found working well is simply to delete the libgomp.so file that is shipped with KeenTools. You probably already got a newer libgomp.so file installed on your system. If that is the case both NNSuperResolution and FaceTracker will pick up the newer libgomp.so shared library from the system, and since this library is backwards compatible everything will work just fine. If you haven't got libgomp.so installed on your system, you need to install it to run the plugins. How that is done is of course dependent on your operating system and such. On CentOS you can install it by simply running the command "yum install libgomp".

I'm having problems running NNSuperResolution in the same Nuke environment as Kognat's Rotobot
Please make sure to match versions of both NNSuperResolution and Rotobot to what exact CUDA version they have been built against. We have synced our efforts together with Kognat, and made sure that we both got compatible releases with each other, built against either **CUDA10.1/cuDNN8.0.5, CUDA11.2/cuDNN8.1.0, CUDA11.1/cuDNN8.4.1** or **CUDA11.8/cuDNN8.4.1**.

I'm having problems/crashing when using NNSuperResolution together with Foundry's AIR nodes in Nuke13.x
You are highly likely using the CUDA11.2/cuDNN8.1.0 build or the CUDA11.8/cuDNN8.4.1 build if this is happening. These builds of NNSuperResolution are for modern GPUs like the RTX3080 or RTX4080 cards, and will work fast and nice except for when you need to process NNSR upscaling together with for example CopyCat nodes. If you need to support this, you have to use the CUDA10.1/cuDN8.0.5 build instead. Please see more info about this above in the environment variables section.

I'm having problems/crashing when using NNSuperResolution together with Foundry's AIR nodes in Nuke14.0
You are highly likely using the CUDA11.8/cuDNN8.4.1 build if this is happening. This build of NNSuperResolution is for modern GPUs like the RTX4080 card, and will work fast and nice except for when you need to process NNSR upscaling together with for example CopyCat nodes. If you need to support this, you have to use the CUDA11.1/cuDN8.4.1 build instead. Please see more info about this above in the environment variables section.

5. Change Log

v3.4.3 - January 2024
- Fixed a regression bug where the RGBA mode was broken.
- Added support for Nuke15.0 (which also is natively built against CUDA 11.8 and cuDNN 8.4.1, on Rocky8).

v3.4.2 - November 2023
- Fixed a bug that created a 1 px wide black line between the image patches when upscaling 2x and only when the padding was set to 1.
- Increased the default value for the knob "overlap" to 12 (previously set to 4), to improve the image quality in the patch blending regions.

v3.4.1 - November 2023
- Corrected a bug that made the node crash when a GPU build was used to process on a machine without an installed GPU. Now it automatically falls back to using the CPU instead, as intended.

v3.4.0 - November 2023
- More efficient releasing of used GPU memory while using the node in a live Nuke GUI session, i.e. you don't have to restart Nuke anymore to get some of the GPU memory back. Also doing a more thorough GPU memory cleanup when the node is fully deleted (usually when you run "File/Clear" in the Nuke GUI).
- Added native support for the NVIDIA RTX40xx series of GPUs, i.e. compute capability 8.9 (also supporting 9.0), with a version built against CUDA 11.8 and cuDNN 8.4.1.
- Added support for Nuke14.1 (which also is natively built against CUDA 11.8 and cuDNN 8.4.1)
- Added a check for GPU compatibility, i.e. disable the GPU processing options if the current GPU is not supported (instead of crashing).
- Improved the error logging. It's now much more clear if the node errors because you are running out of GPU memory.

- Added text information knobs to the “About” tab that prints out what CUDA/cuDNN versions the current build is against. Also what compute capability that is natively supported and what compute capability your current GPU supports and info about if you will or will not use JIT-compiled kernels. We’ve also added info about compatibility against Nuke’s own AIR tools in relation to the current build configuration.
- Added a warning to the terminal during node creation if you are relying on JIT-compiled kernels. Just so you get some kind of heads up the first time the JIT-compilation kicks in (which usually takes just above half an hour). There are also warnings about the CUDA_CACHE_MAXSIZE environment variable if it’s not set correctly. These warning can be suppressed with a new environment variable PIXELMANIA_SUPPRESS_KERNEL_WARNINGS if you want.
- Fixed a bug so the plugin can now upscale single channel layers, for example when you have a depth channel present in a multichannel CG render and you are upscaling all channels.
- Improved internal processing and memory handling. This improves the performance overall. According to our own tests, the performance increase is between 20-25% in sequence mode when upscaling 4x.

v3.3.0 - March 2023
- Added support for dedicated render licenses.

v3.2.1 - Sept 2022
- Fixed a regression where the overscan handling stopped working in sequence mode, but instead resulted in a crash. This was working in v3.0.0, but broke in the optimization release v3.2.0. It’s now fully corrected and working as it should again.

v3.2.0 - May 2022
- Speed optimizations overall. According to our own internal testing, sequence mode is now about 30% faster to render.
- Added an option for processing in mixed precision. This is using a bit less VRAM, and is a quite a lot faster on some GPU architectures that are supporting it (RTX).
- Added an option for choosing what CUDA device ID to process on. This means you can pick what GPU to use if you got a workstation with multiple GPUs installed.
- Disabled initial heuristics to fix a slow down issue on certain GPU architectures that happened on the first processing frame.
- Optimized the build of the neural network processing backend library. The plugin binary (shared library) is now a bit smaller and faster to load.
- Built the neural network processing backend with MKLDNN support, resulting in a vast improvement in rendering speed when using CPU only. According to our own testing it’s sometimes even less than 25% of the render time of v3.0.0 (in sequence mode).
- Updated the NVIDIA cuDNN library to v8.0.5 for the CUDA10.1 build. This means we are fully matching what Nuke13.x is built against, which means our plugin can co-exists together with CopyCat nodes as well as other AIR nodes by Foundry.
- Built the neural network processing backend with PTX support, which means that GPUs with compute capability 8.0 and 8.6, i.e. Ampere cards, can now use the CUDA10.1 build if needed (see above). The only downside is that they have to JIT compile the CUDA kernels the first time they run the plugin. Please see the documentation for more information about setting the CUDA_CACHE_MAXSIZE environment variable.

- Internal checking that the bounding box doesn't change between frames in sequence mode (it's not supported having animated bboxes). Now it's throwing an error instead of crashing.
- Bug fixes to the "frame range knobs" handling and the "reset frame range" button
- Better render status logging for what layer is processing to the terminal
- Better error reporting to the terminal
- Added support for Nuke13.2

v3.0.0 - November 2021
- Totally retrained all sequence mode networks from scratch with a higher neural network capacity, to improve the overall upscale results.
- Trained all the 2x versions of the networks, for both still and sequence mode. Added the scale knob to control the upscale factor.
- Added the details knob to control the blend between the PSNR and GAN networks.
- Renamed the sequence type option “Plates (legacy) [RGB]” to “Plates (Vimeo) [RGB]”.
- Fixed the handling of formats to support upscaling of anamorphic formats.

v2.5.0 - June 2021
- Totally retrained plate (RGB) solution with sharper and more detailed results.
- CG (RGBA) upscale option available (beta).
- Nuke Indie support (requires the latest version of Nuke12.2 or Nuke13.0).

v2.0.1 - March 2021
- Bug fix to the progress bar that didn’t show the right thing when processing the preroll in sequence mode using several image patches.
- Bug fix to a crash that happened when switching between GPU and CPU mode in the same Nuke session.

v2.0.0 - March 2021
- Release of sequence mode.
- Support for additional layers, i.e. if you got multiple layers and render them through with a Write node set to “channels: all” the plugin will run the upscale separately on each layer. Please note that there is no alpha channel (4th layer channel) support yet in this version.
- Renamed the status label from “Inference runs” to “Image patches” to be more clear to end users and also make sense in sequence mode.
- Fixed a bug relating to processing images using multiple scanline compression.

v1.0.0 - December 2020
- Initial release (still mode)