Neural Texture Block Compression

Shin Fujieda, Takahiro Harada

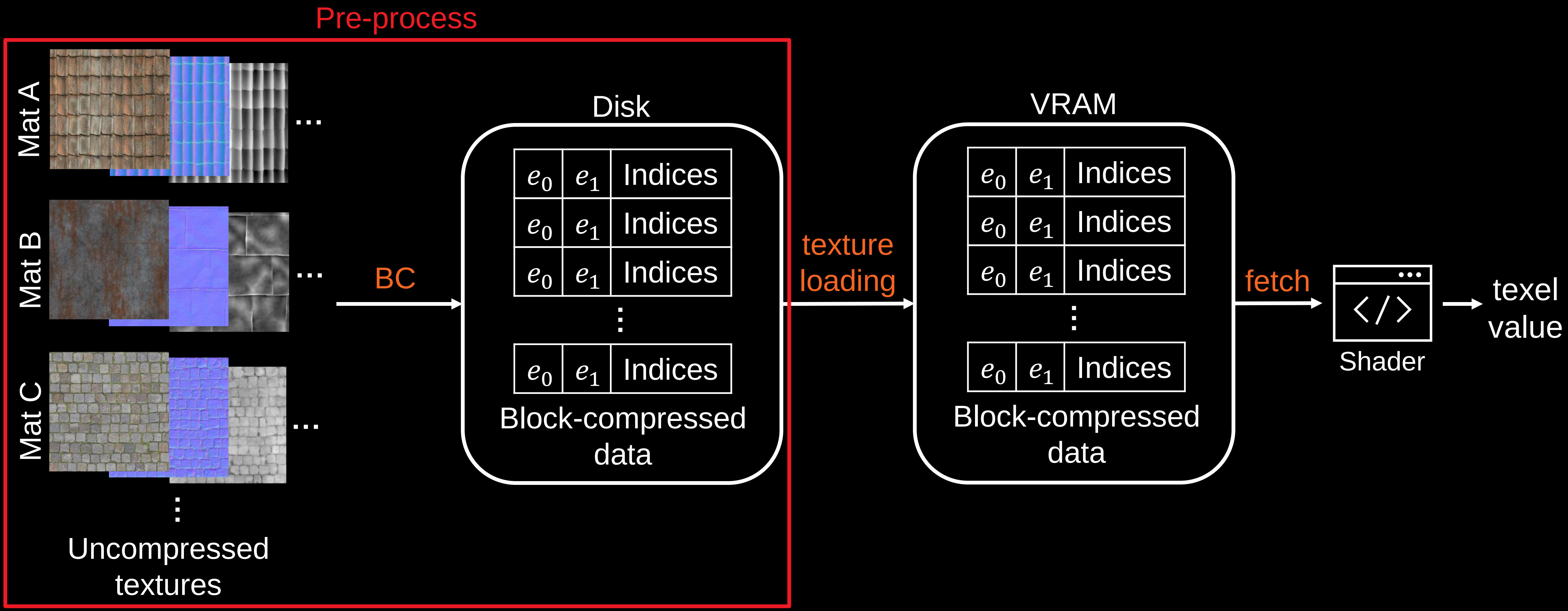
Motivation

- Texture is a key component to achieve high visual fidelity through material properties
  - High-resolution textures (e.g., 4K) require a lot of storage

- Block compression (BC) is one of the most popular techniques to compress textures
  - All variations, BC1 – BC7, compress each 4x4 texel block to a fixed number of bytes
  - BC1 and BC4: 8 bytes, others: 16 bytes
- Consumes 8MB (= 4096*4096 / (4*4) * 8 bytes) for a single 4K texture with BC1

- To reduce the storage costs, we propose Neural Texture Block Compression (NTBC)
  - Compress textures in BC1/BC4 formats using a multi-layer perceptron (MLP)
  - Not require any change in the shader execution

Pipeline - conventional



Pre-process

Mat A ...

Mat B ...

Mat C ...

⋮

Uncompressed textures

BC

Disk

e_0	e_1	Indices
e_0	e_1	Indices
e_0	e_1	Indices
⋮		
e_0	e_1	Indices

Block-compressed data

texture loading

VRAM

e_0	e_1	Indices
e_0	e_1	Indices
e_0	e_1	Indices
⋮		
e_0	e_1	Indices

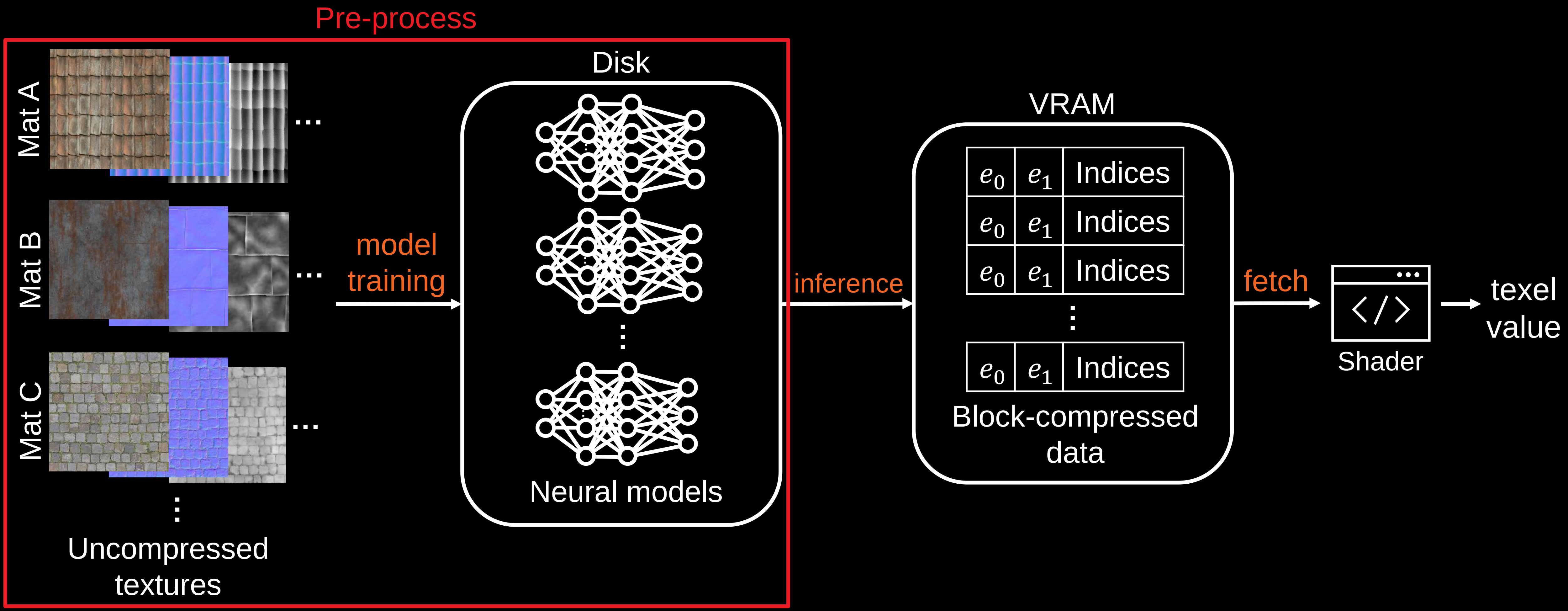
Block-compressed data

fetch

Shader

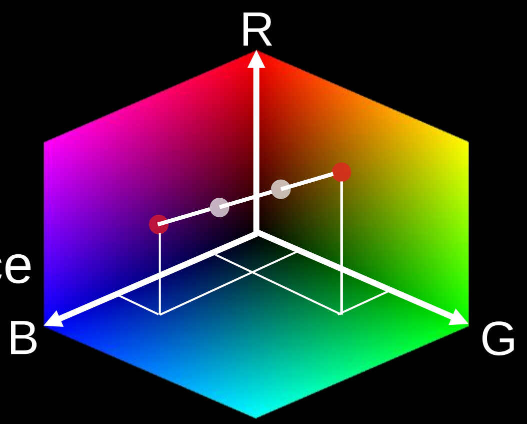
texel value

Pipeline - NTBC



e_0	e_1	Indices
e_0	e_1	Indices
e_0	e_1	Indices
⋮		
e_0	e_1	Indices

Block Compression



- Block compression (BC) encodes 4x4 texel blocks into a fixed-size structure
- Each block contains a color palette with colors on a line segment in RGB space
  - Two endpoints + linear-interpolations

BC1

- RGB images
- 2 RGB565 endpoints (4 bytes)
  - Palette has 4 entries
- 16 2-bit indices (4 bytes)
  - $0 \leq n \leq 3$

BC4

- Single-channel images
- 2 8-bit endpoints (2 bytes)
  - Palette has 8 entries
- 16 3-bit indices (6 bytes)
  - $0 \leq n \leq 7$

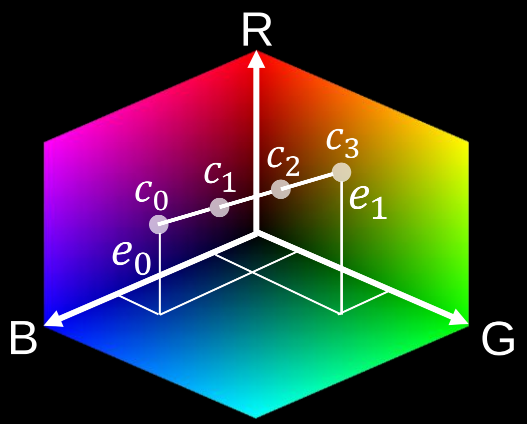
How can we encode these block-compressed textures using neural networks?

Naive approach

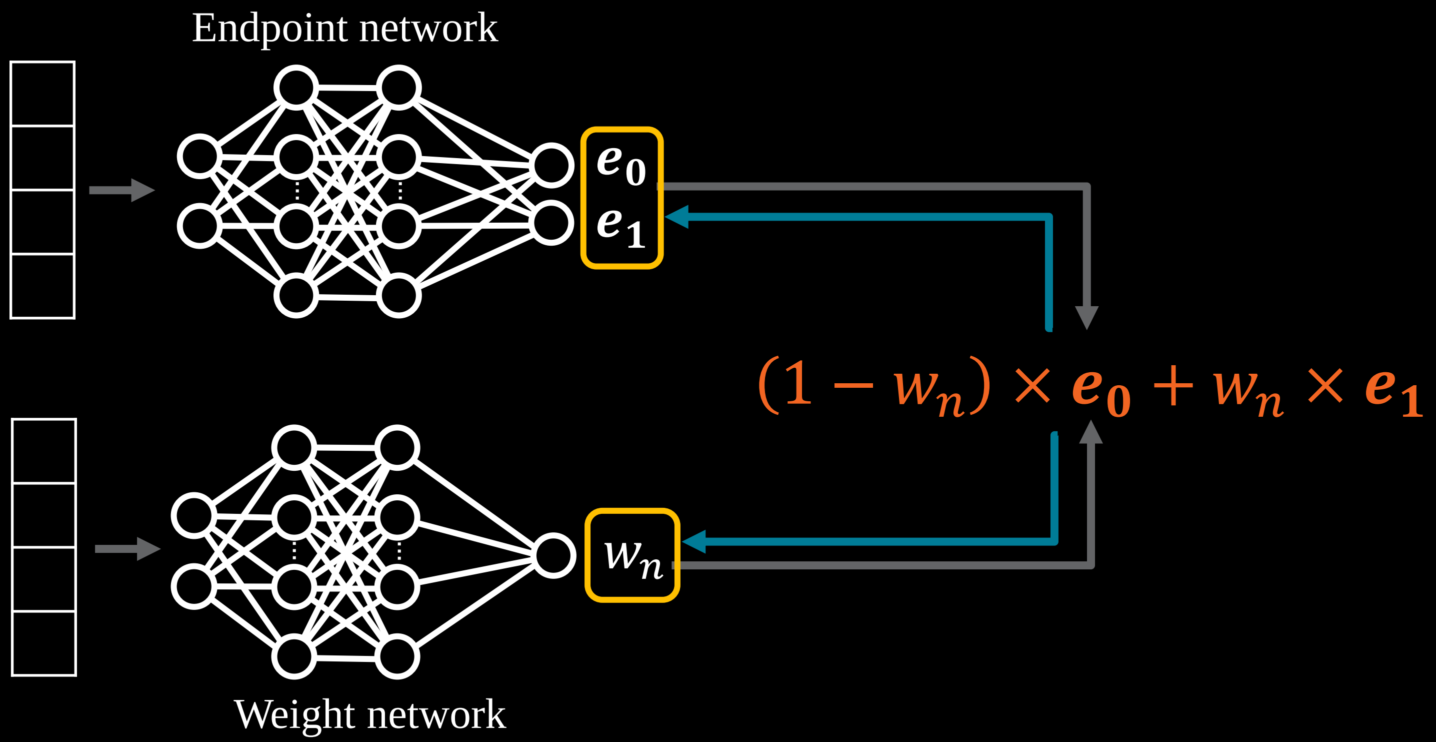
- Colors of the palette are evenly spaced on a line segment in RGB space:

$$\boldsymbol{c_n} = (1 - w_n) \times \boldsymbol{e_0} + w_n \times \boldsymbol{e_1}$$

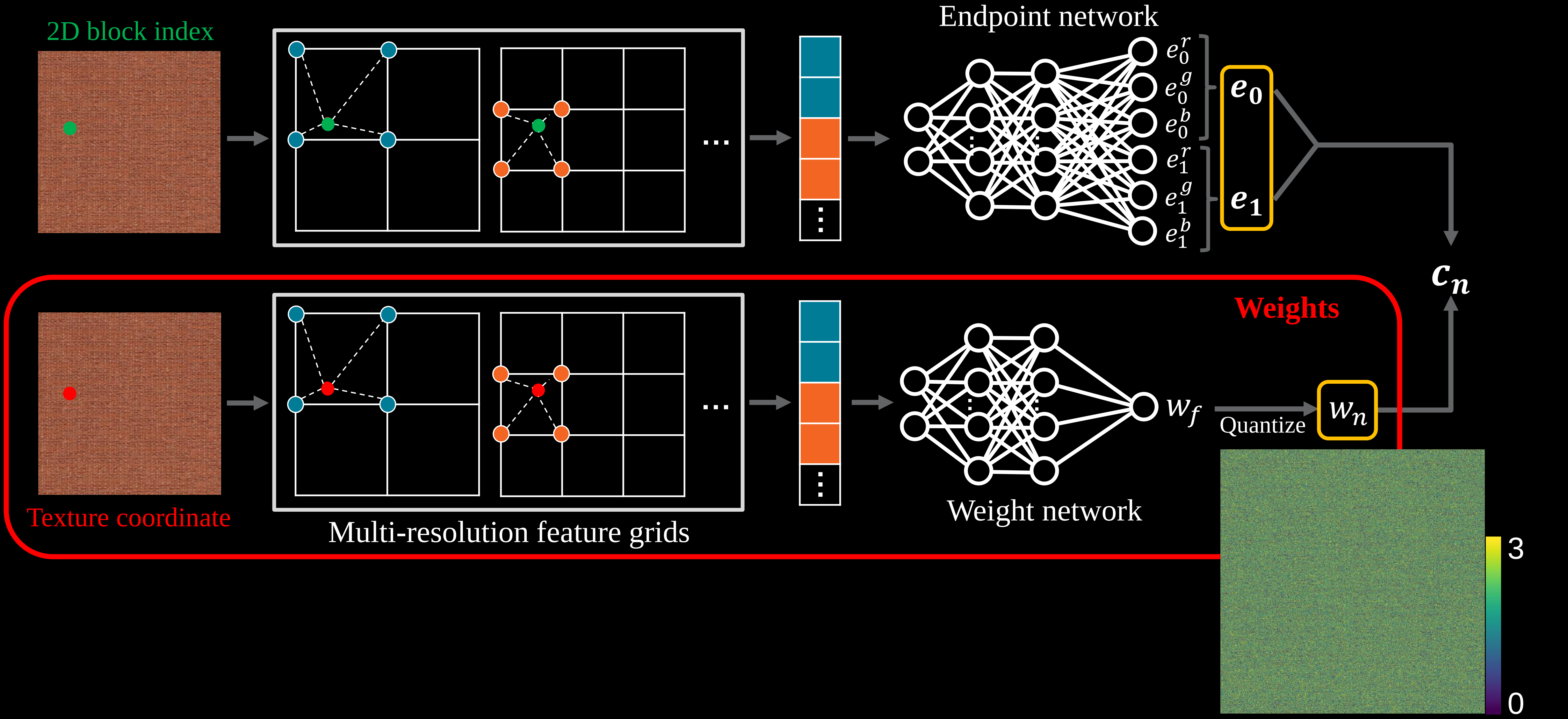
$\boldsymbol{e_0}, \boldsymbol{e_1}$: two endpoints, w_n: weights of $\frac{n}{3}$ (BC1), $\frac{n}{7}$ (BC4)



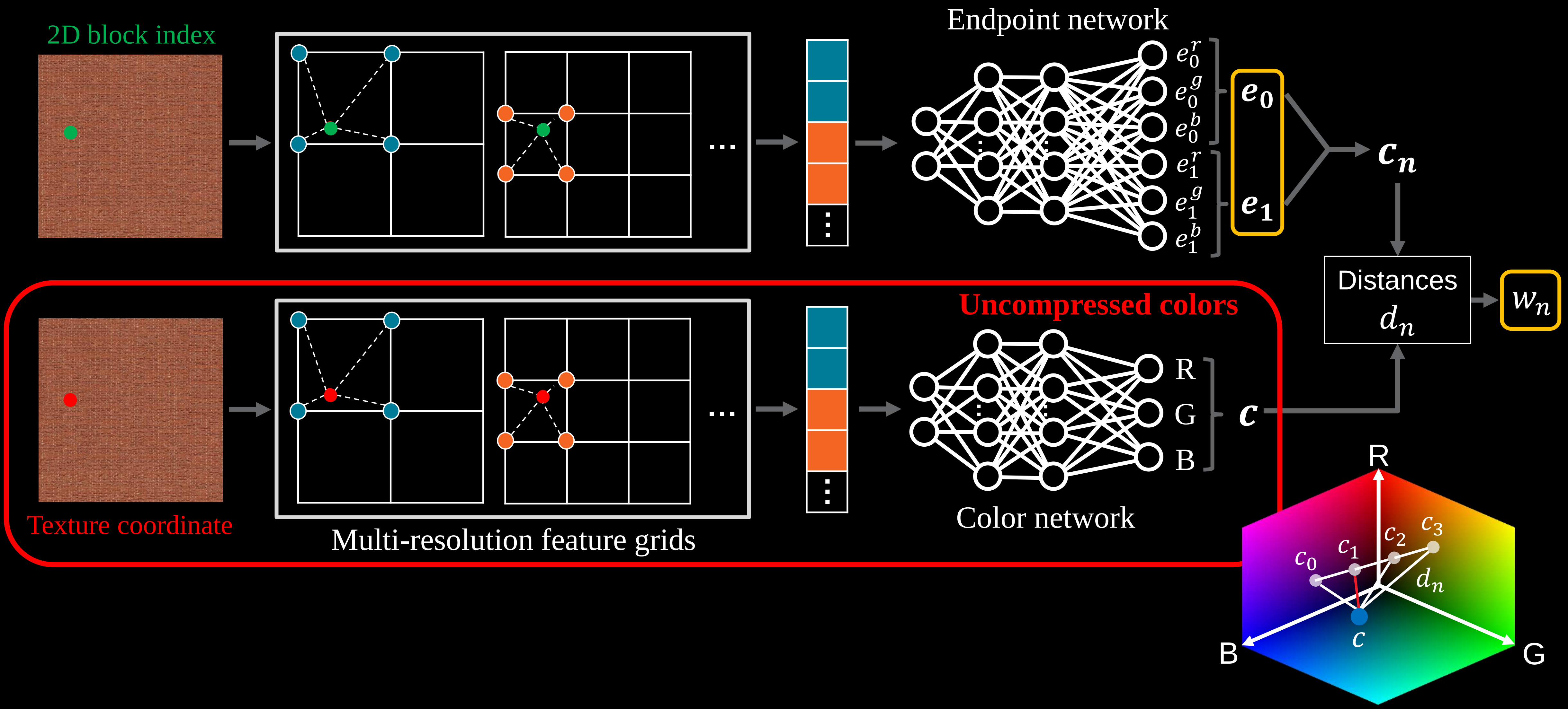
- Encoding e_0, e_1, w_n using NNs should be a straight-forward approach



Naive approach



Neural Texture Block Compression



NTBC – model configuration

- Multi-resolution feature grids

  Endpoint network
  - 7 levels
  - Resolutions from 16 to 1024
  - 2D features per level

  Color network
  - 8 levels
  - Resolutions from 16 to 2048
  - 2D features per level
- Small MLPs with FP16
  - 64 x 3 hidden layers
- Adam optimizer

NTBC – grid quantization

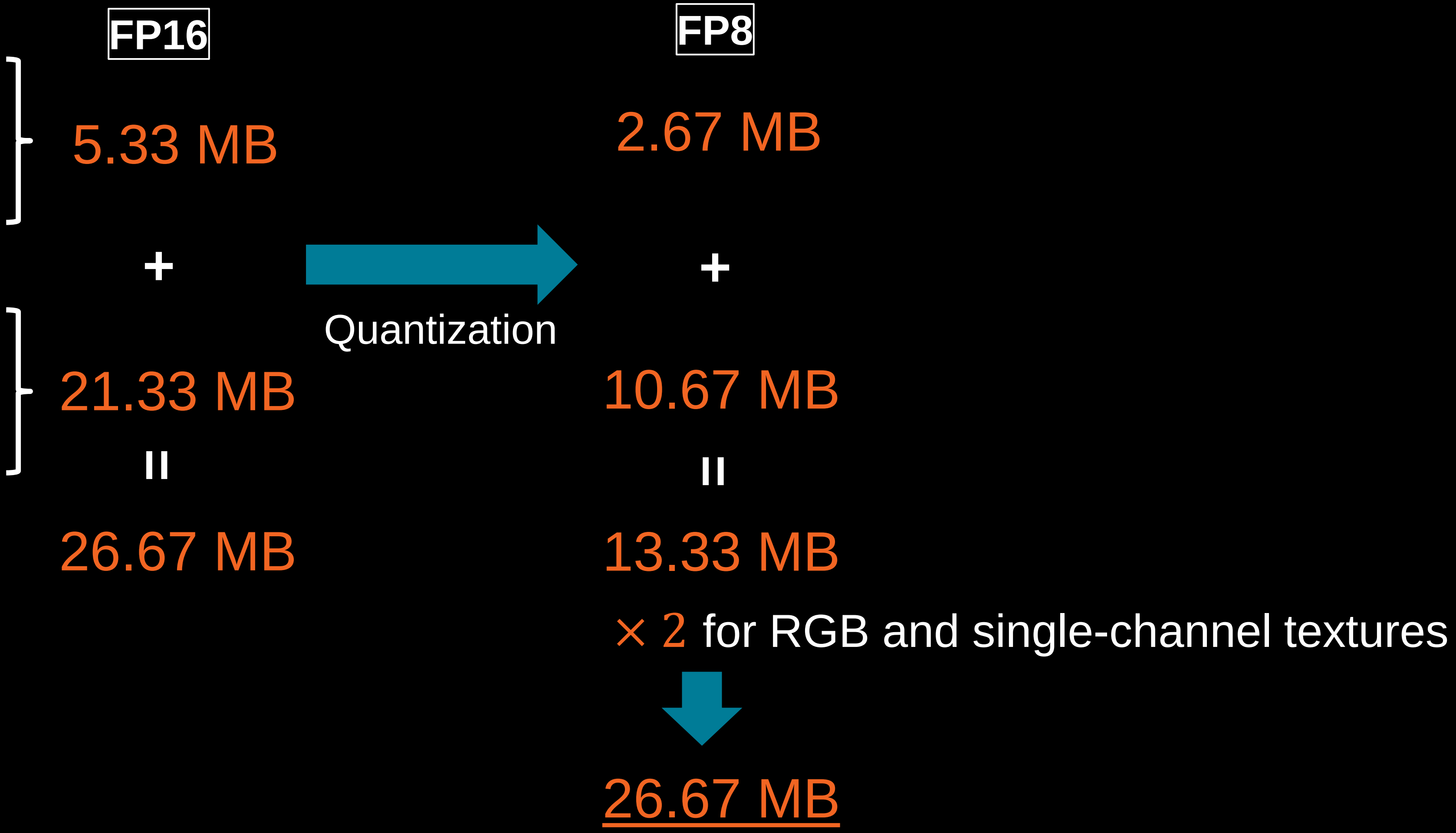
- Multi-resolution feature grids

<u>Endpoint network</u>

- 7 levels
- Resolutions from 16 to 1024
- 2D features per level

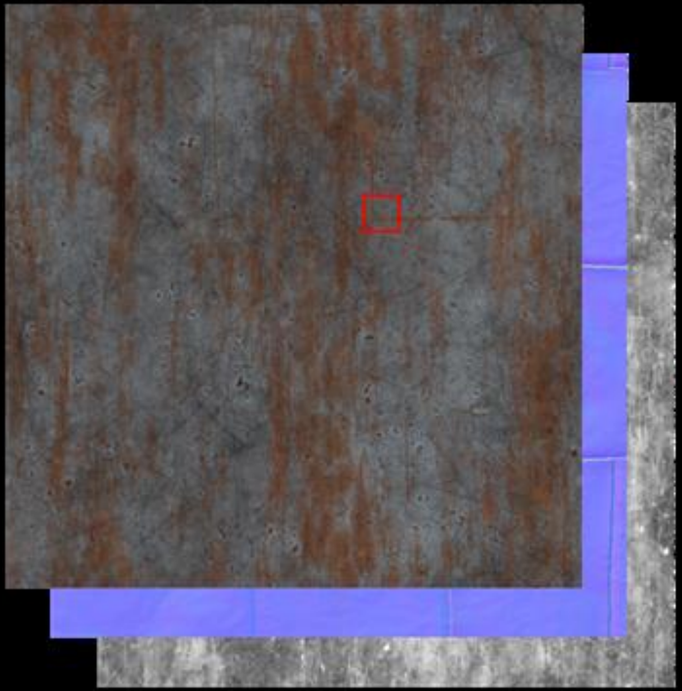
<u>Color network</u>

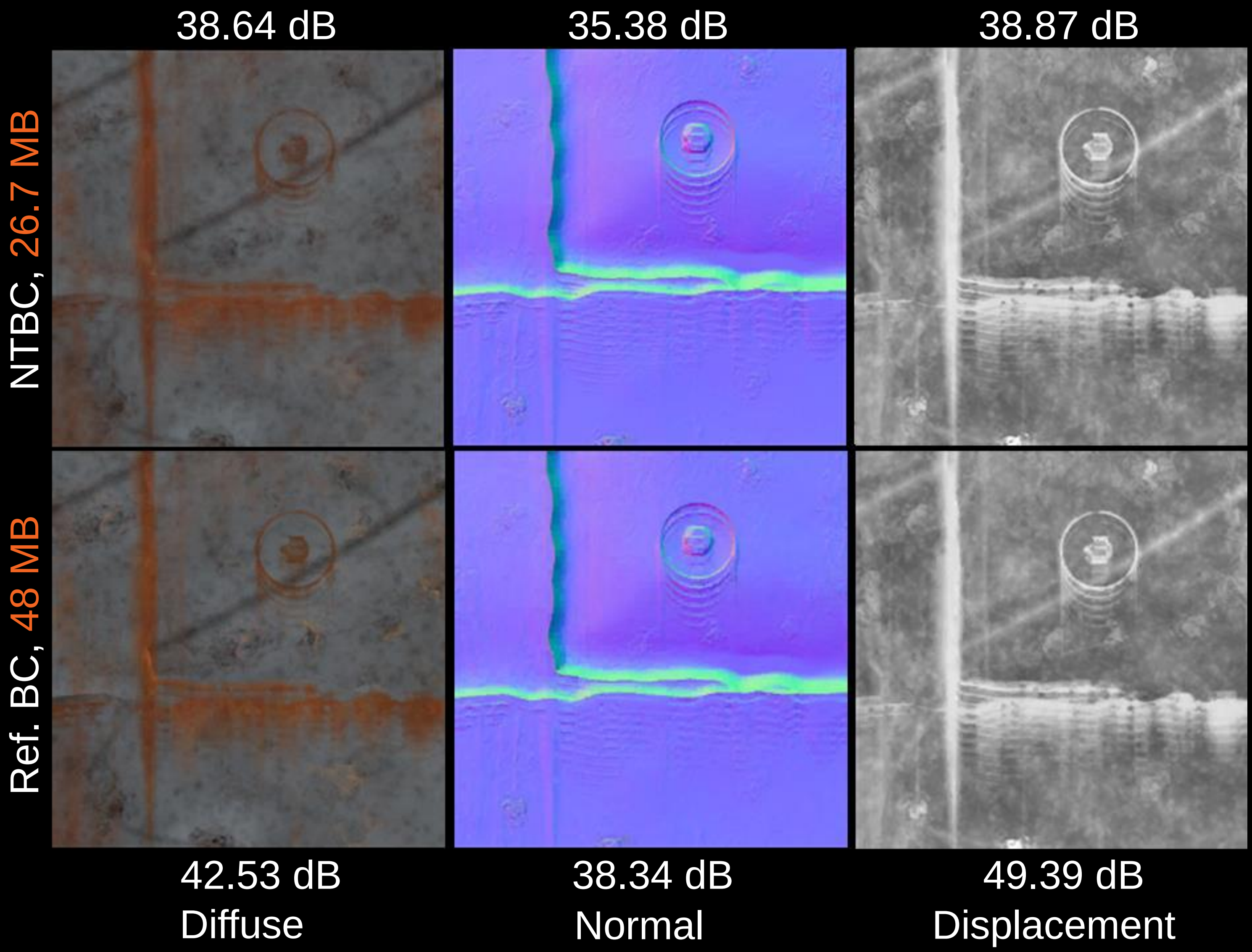
- 8 levels
- Resolutions from 16 to 2048
- 2D features per level



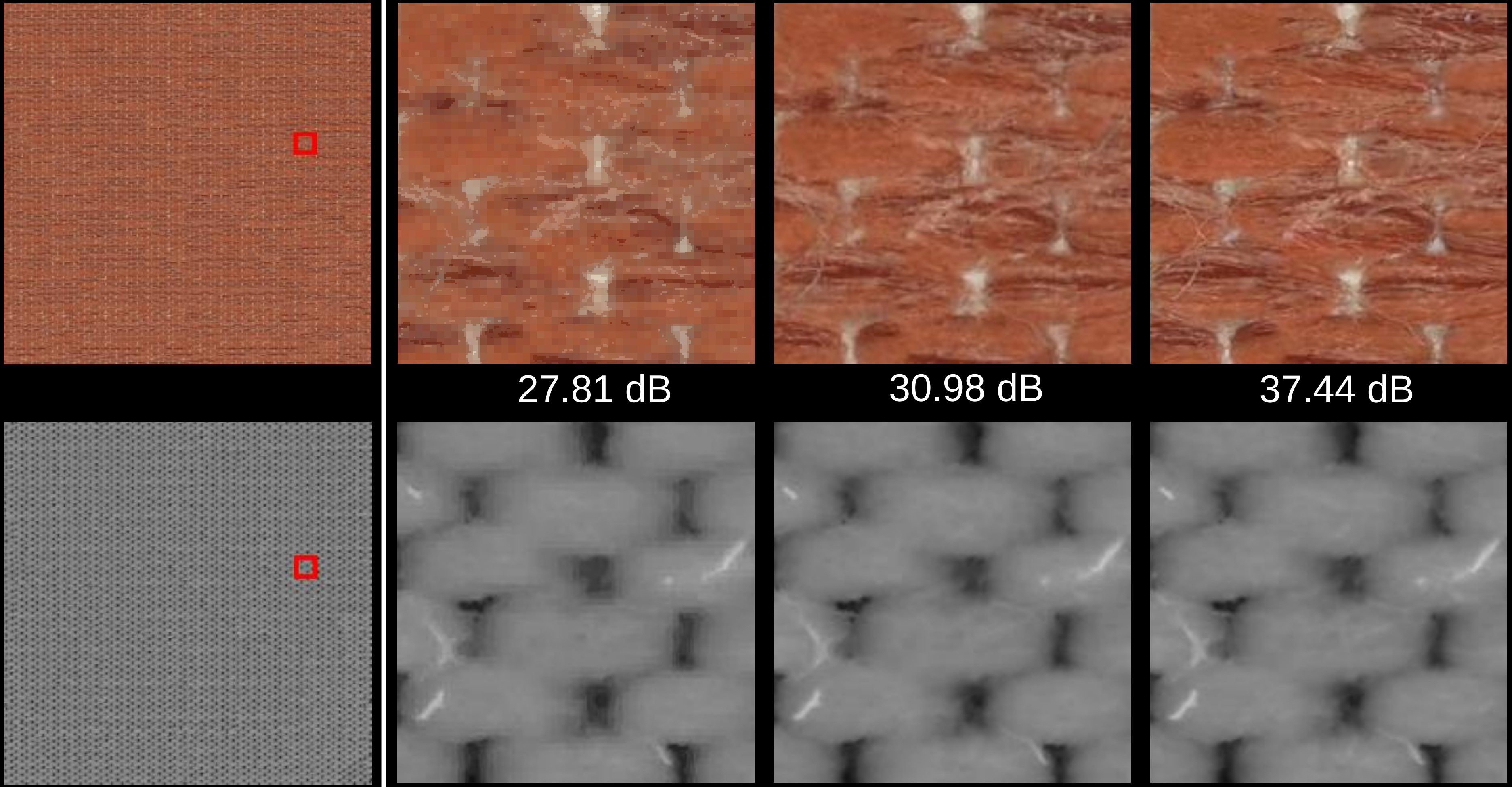
Comparisons

Uncompressed textures





Comparisons

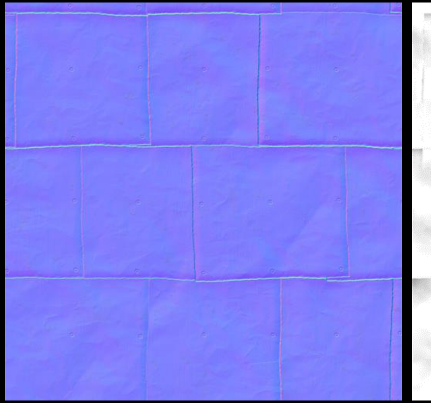


	Uncompressed textures	Naive approach 26.7 MB	NTBC 26.7 MB	Ref. BC 40 MB
Diffuse		27.81 dB	30.98 dB	37.44 dB
Displacement		38.53 dB	44.20 dB	53.06 dB

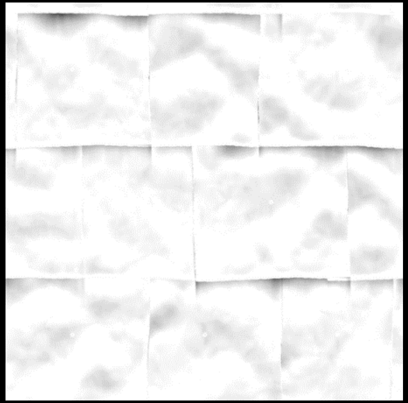
Performance

- Evaluate training + inference performance with a single AMD Radeon^{TM} RX 7900 XT GPU
- Use “MetalPlates013” material
  - 2 RGB textures
  - 4 single-channel textures

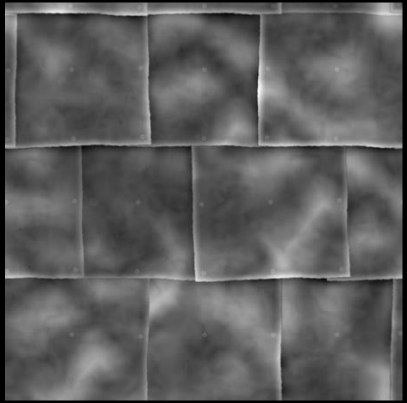
Diffuse



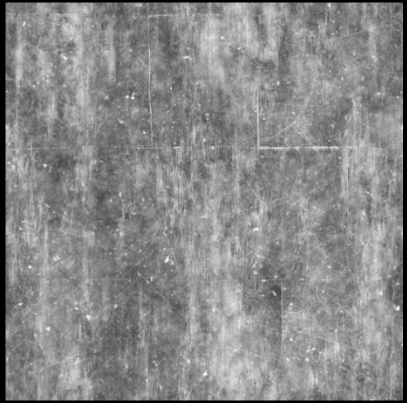
Normal



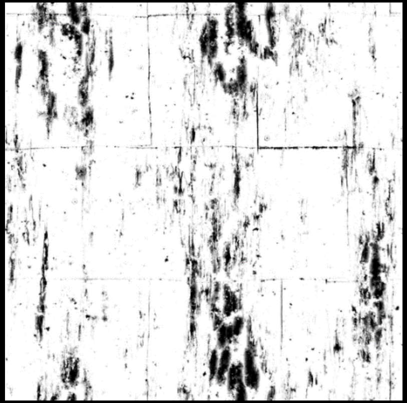
AO



Displacement



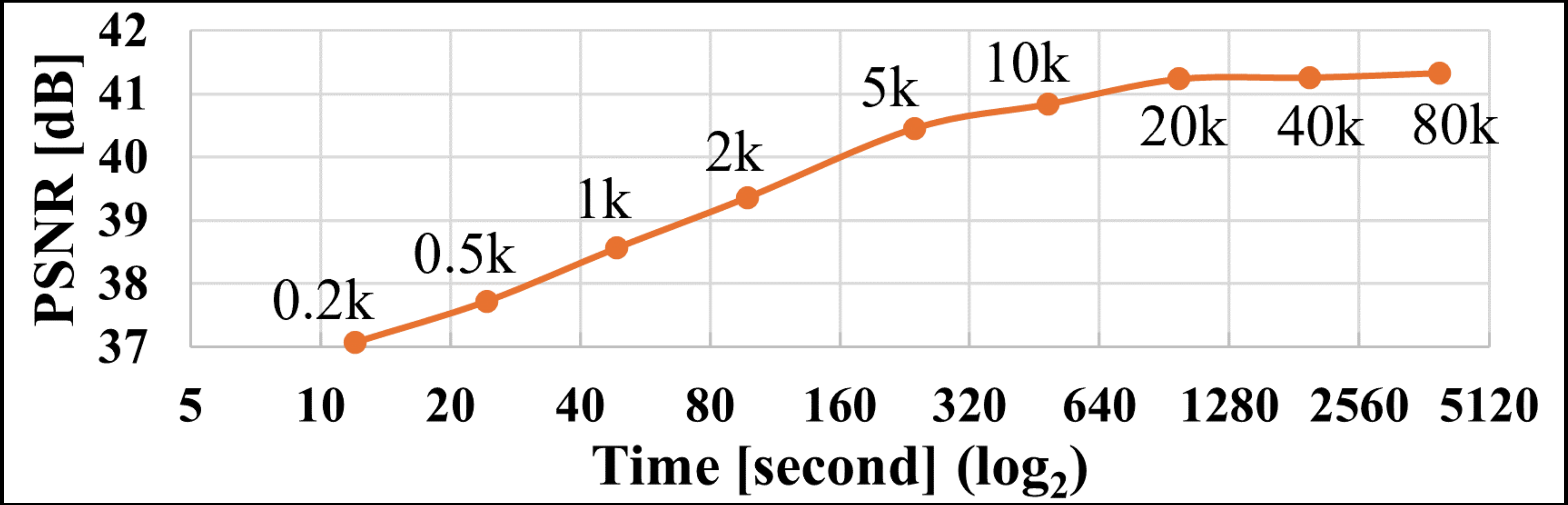
Roughness



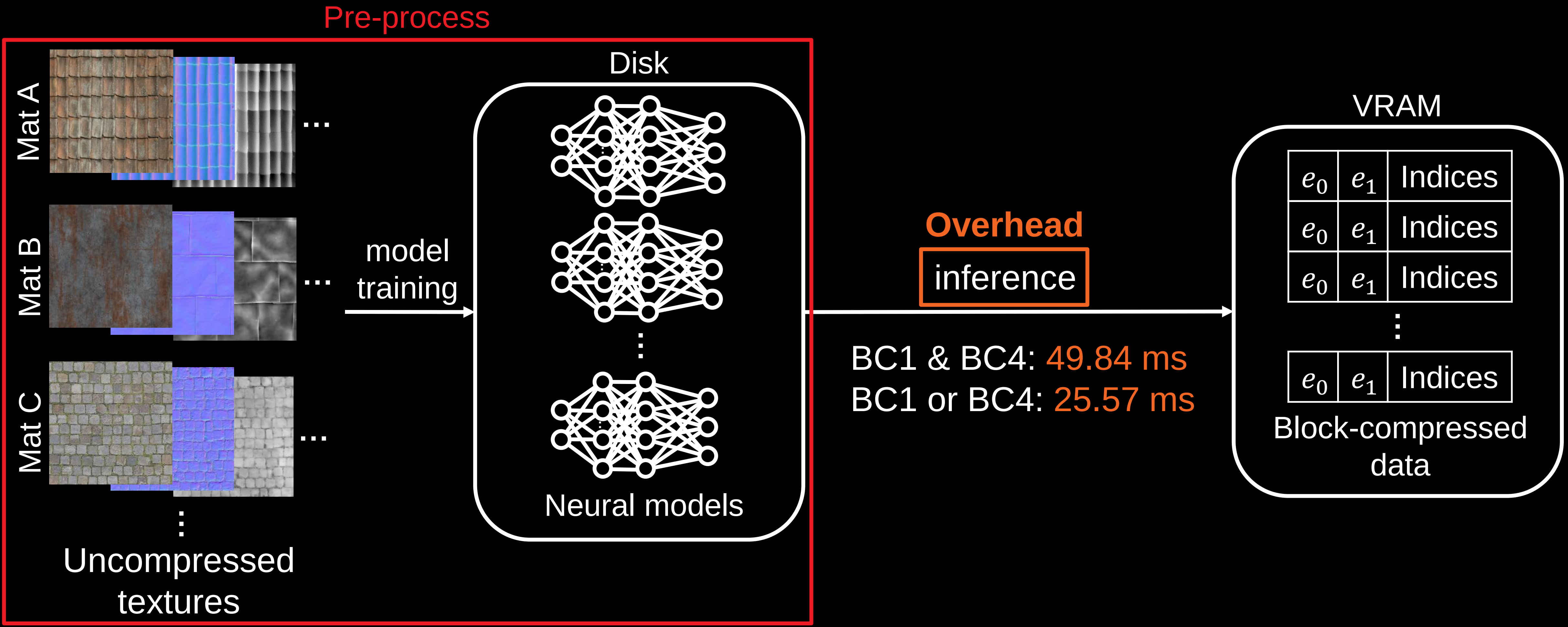
Metalness

Performance - Training

- NTBC training time over PSNR averaged over all the textures in the material
- 20k iterations gives saturated results in 16 minutes
- 2k iterations provides reasonable quality just in 100 seconds



Performance - Inference



Pre-process

Mat A

Mat B

Mat C

⋮

Uncompressed textures

model training

Disk

Neural models

Overhead

inference

BC1 & BC4: 49.84 ms

BC1 or BC4: 25.57 ms

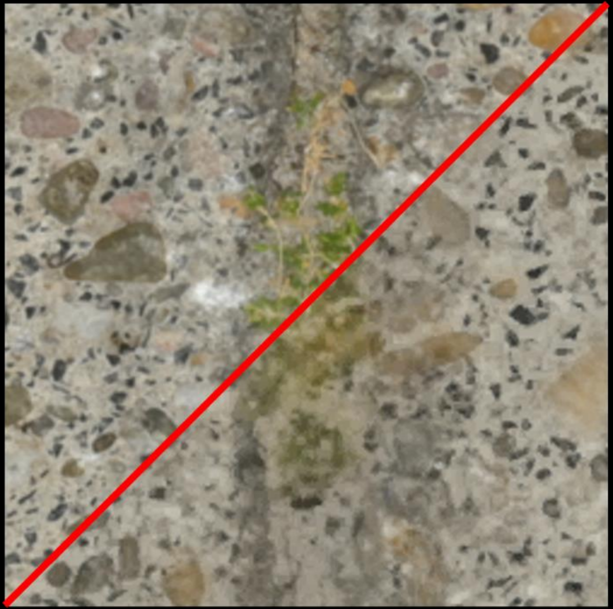
VRAM

e_0	e_1	Indices
e_0	e_1	Indices
e_0	e_1	Indices
⋮		
e_0	e_1	Indices

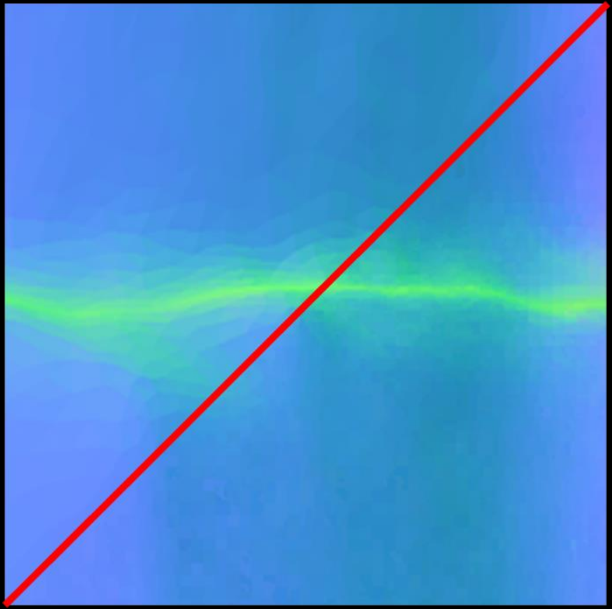
Block-compressed data

Limitations

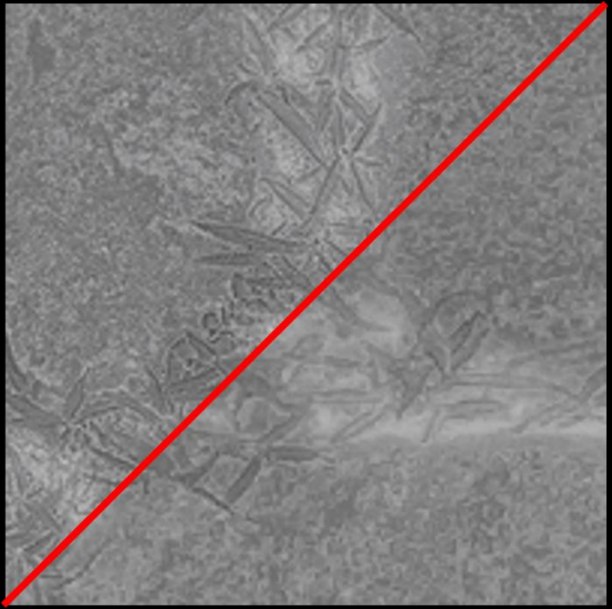
- Color degradation
  - Happens if the texture contains high-frequency details both in luminance and colors
- Loss of detailed content
  - Block artifacts and blurred details
  - NTBC uses lower-resolution grids than the original texture, which could cause the errors



Color degradation



Block artifacts



Blurred detail

Future Work

- Input encoding specialized for textures
  - Use frequency information to handle high-frequency details efficiently
- Extension to more complex formats such as BC6H and BC7
  - BC1 and BC4 are simple but only show limited-quality results
- Mipmap support
- Evaluation in practical real-time applications

ARR

Advanced Rendering Research Group