LUT* to Color Grade

NN Model to make Color Grading Parameters from a Lookup Table*

Final Project

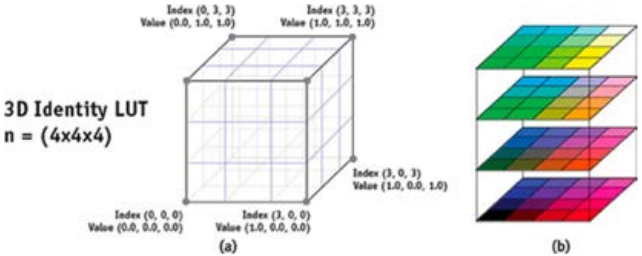
AI for Media, Art, and Design (CS-E7770)
Aalto University

Ossi Luoto

16.2.2024

What problem are we trying to solve?

- Lookup Tables (LUTs) are used in game engines at the post processing stage to implement a “look” to the scene colors after tonemapping
- LUTs come in many shapes, engines often use a 16x16x16 texture (256x16)
- Main principle is to (linear) interpolate RGB colors (#554433 >> #664405)
- Relatively easy to use in limited dynamic range (LDR) but broken for modern HDR pipelines

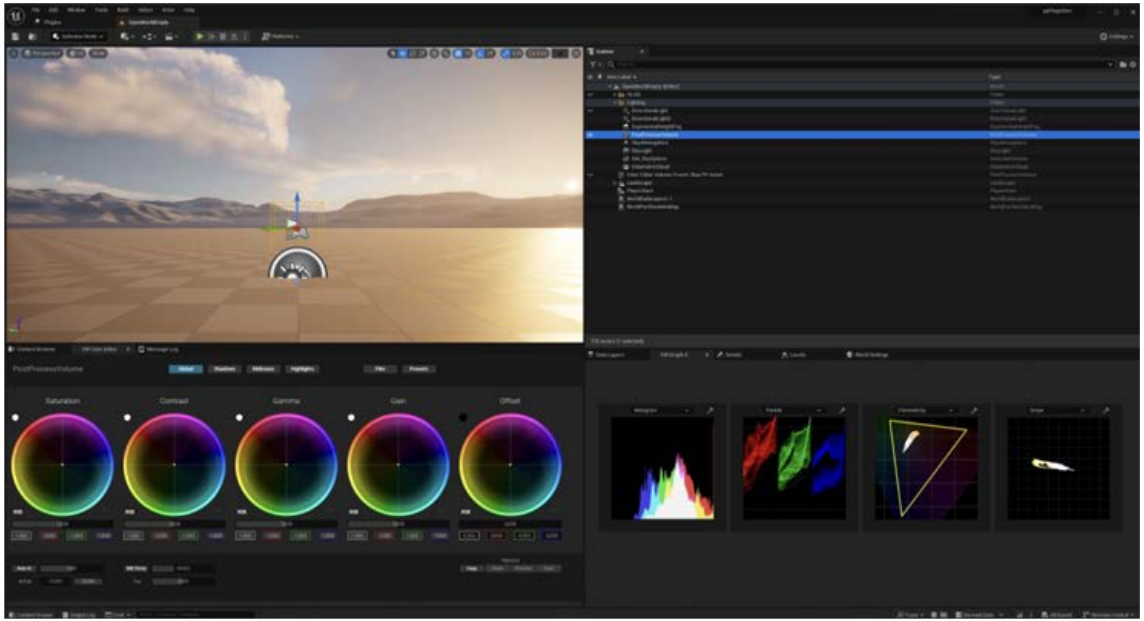


NVidia GPU Gems 2 Chapter 24: Using Lookup Tables to Accelerate Color Transformations
https://developer.nvidia.com/gpugems/gpugems2/part-iii-high-quality-rendering/chapter-24-using-lookup-tables-accelerate-color

Unreal Engine documentation: Using Lookup Tables (LUTs) for Color Grading
https://docs.unrealengine.com/5.3/en-US/using-look-up-tables-for-color-grading-in-unreal-engine/

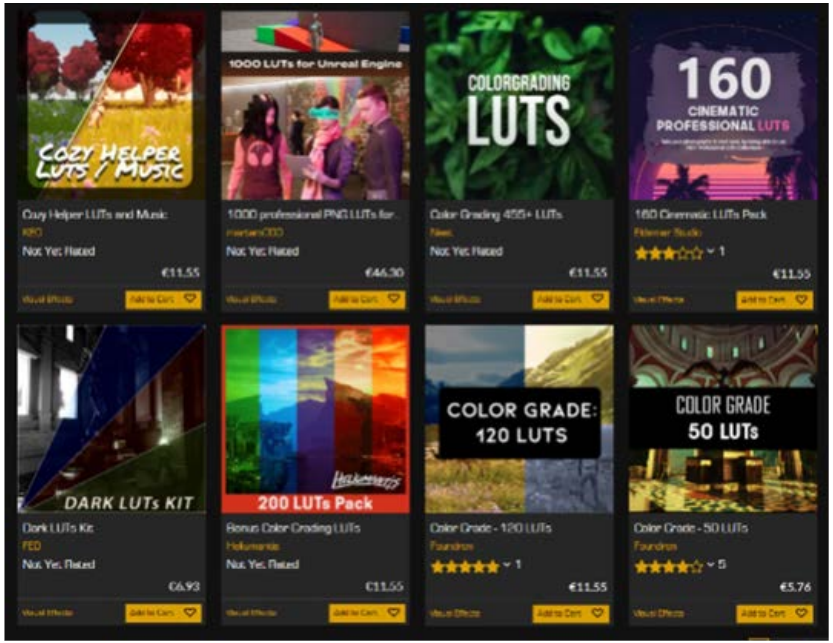
Modern (HDR) Color Grading to the rescue

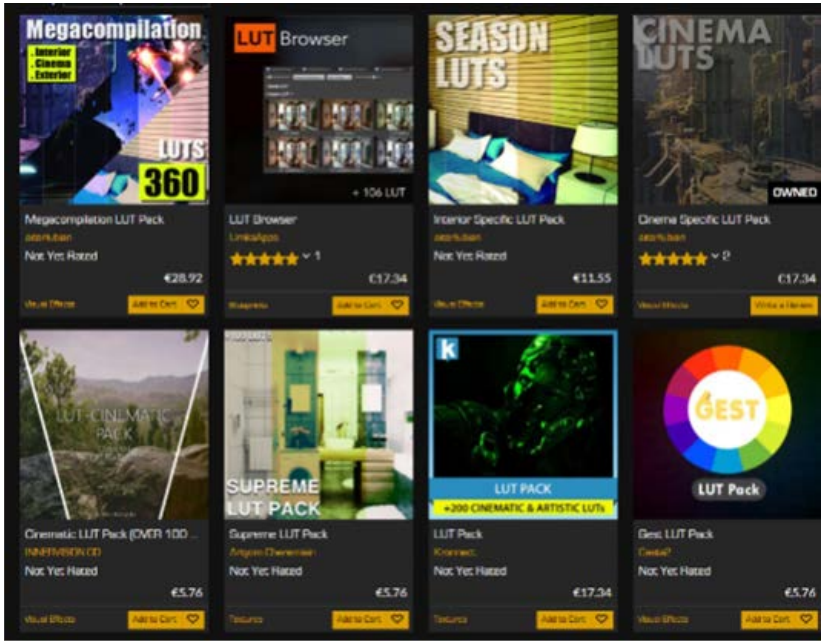
LUTs are not recommended for post processing as game engines (Unity, Unreal, Godot) provide industry standard grading and tonemapping implementations like Academy Color Encoding System (ACES) for more accurate color grading operations.



Color Editor, Plugin for Unreal Engine 5.3

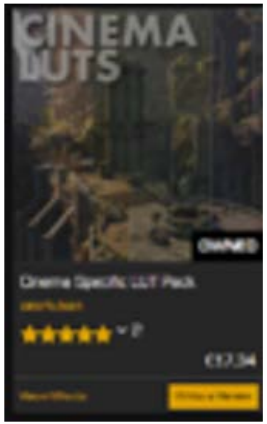
But… LUTs are still being used



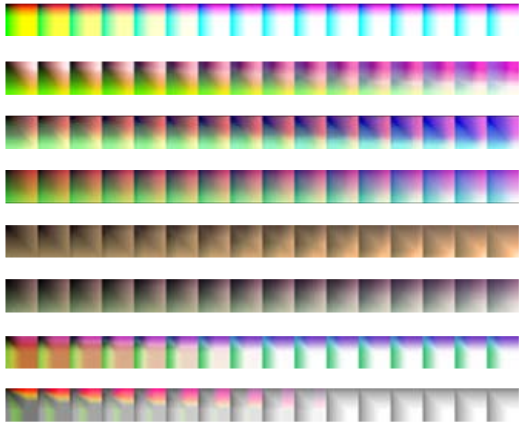


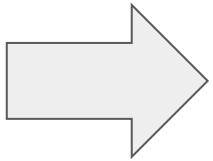
Epic Unreal Engine Marketplace (keyword: LUT)

Project: Neural network model for Color Grading parameters



Input: 256x16 texture



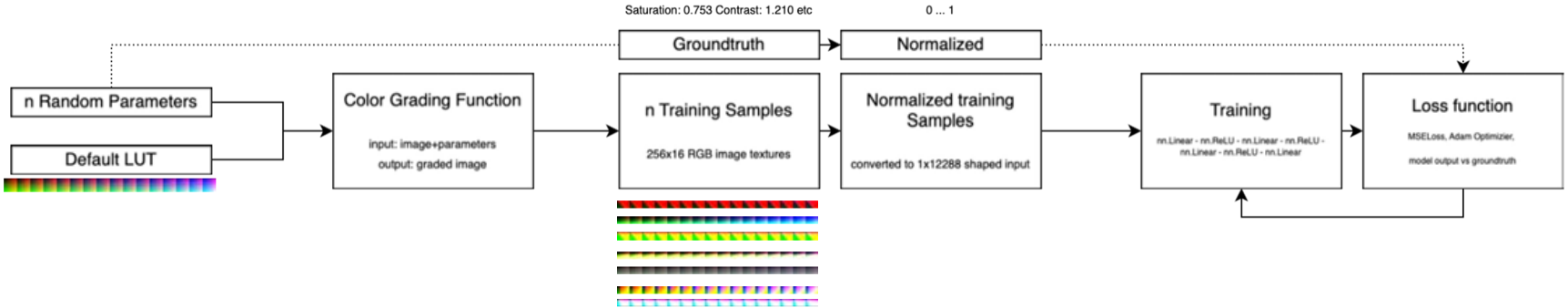


Output: ~10 parameters

- Saturation
- Contrast
- Gamma
- Gain
- Offset

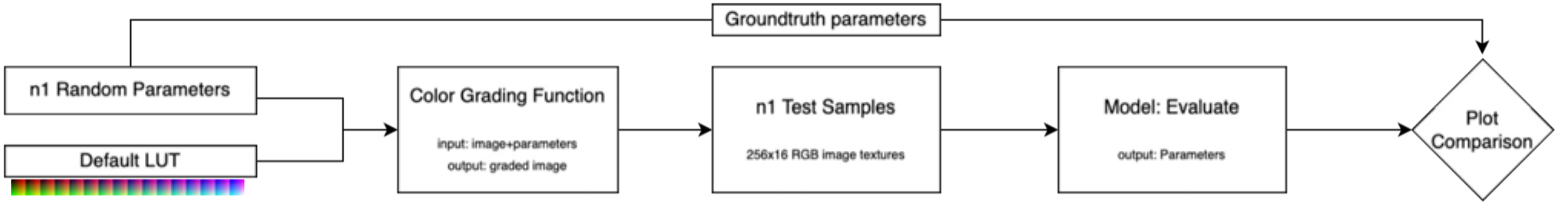
(out of ~100 total)

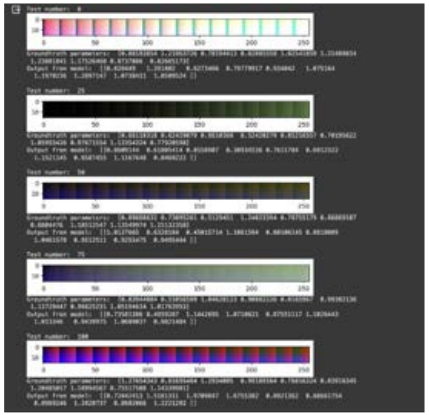
Training the model

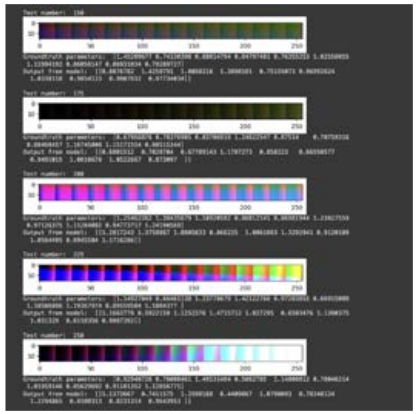


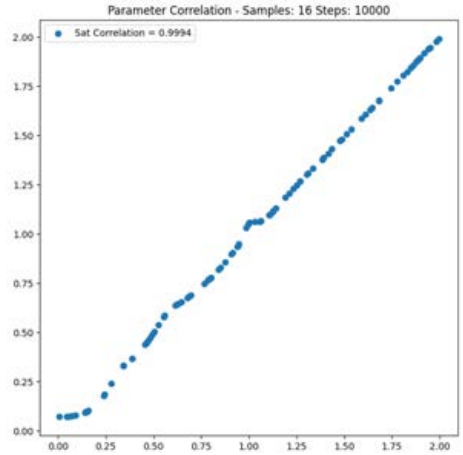
- Proof-of-concept: Saturation (W), Second version (Saturation + Contrast)
- Saturation, Contrast, Gamma, Gain, Offset (W,R,G,B) - Global / Shadows / Midtones / Highlights
- Training sample pool sizes tested from 128 to 65536
- Sobol sequence to create evenly distributed samples in n dimensional space (n being parameters currently in training)
- Training and sample generation combined: from 40 seconds to 14 hours
- GPU (CUDA) optimized version of training - quickly ran out of free Google Colab quota, adjusted for local NVidia 4070Ti training

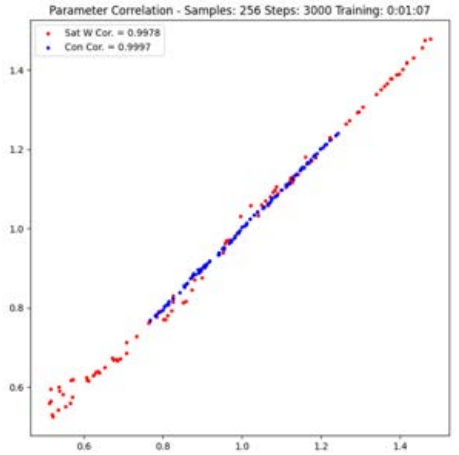
Benchmarking model performance











Benchmark notes

1) Optimal parameters

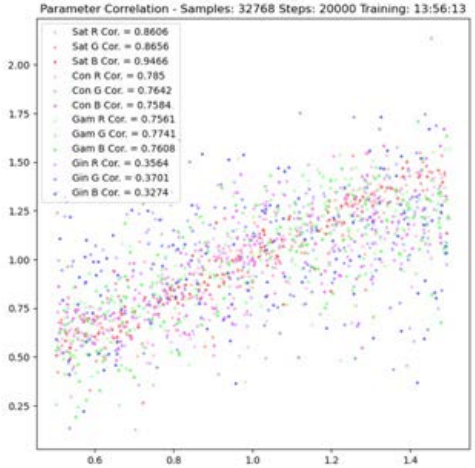
   Tested with 1-12 parameters, current best results with 9 parameters

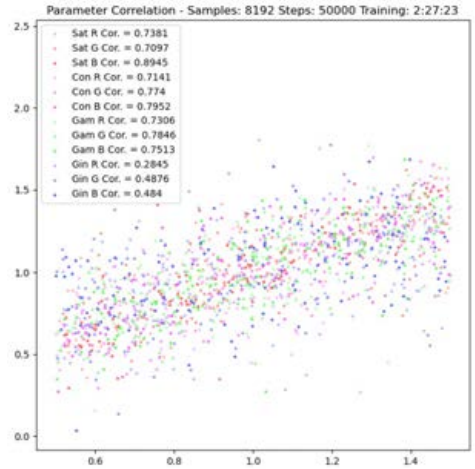
1) Input range limits (LDR)

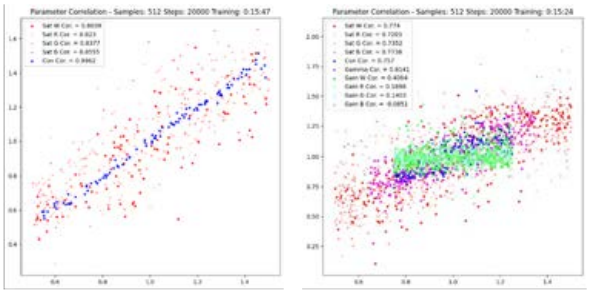
1) Minimize parameter crosstalk

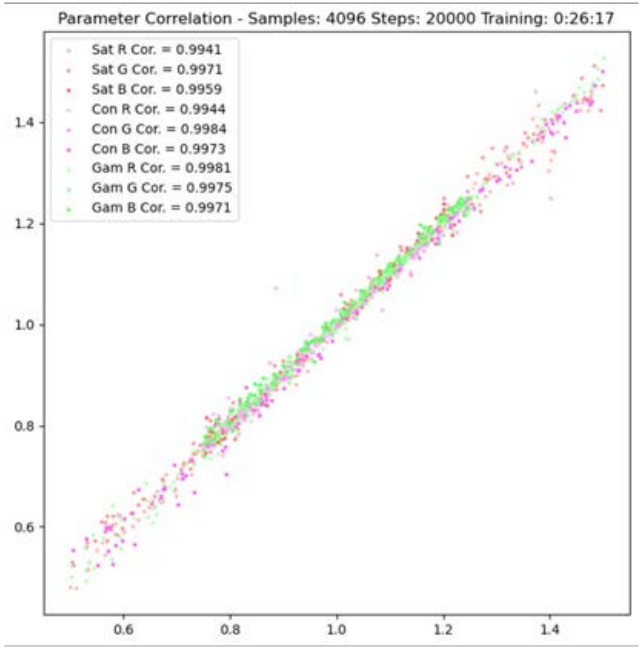
1) With grading, each pixel gets same treatment

   No drastic changes beyond 4k samples, sampleSize*16*16*16 (65536)



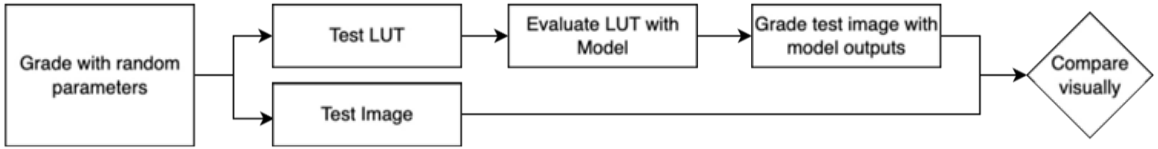


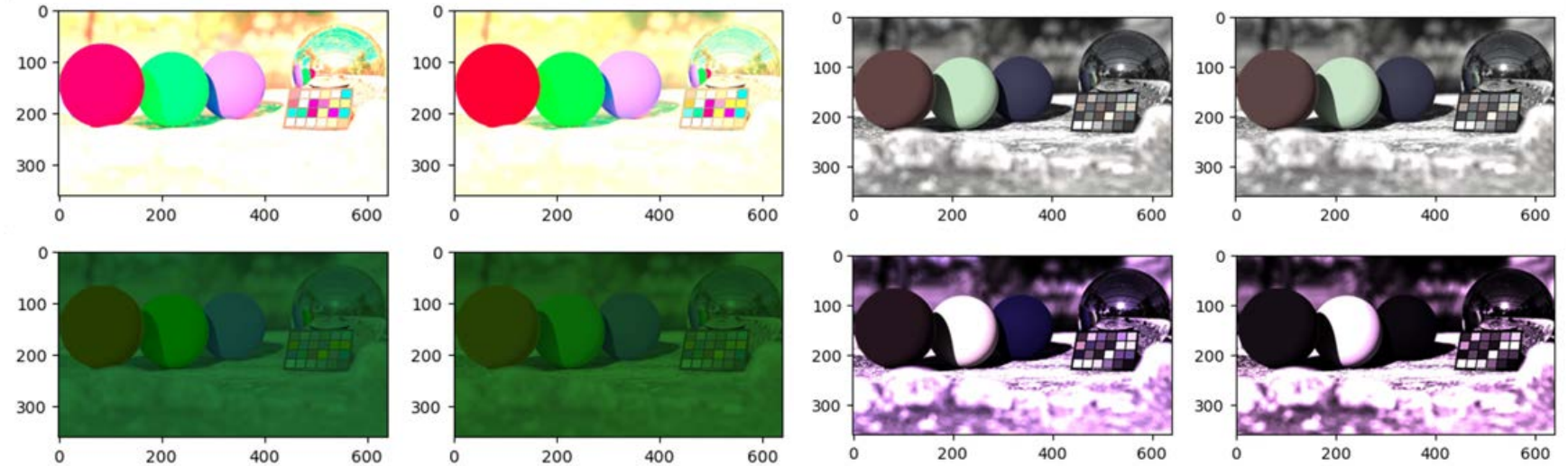




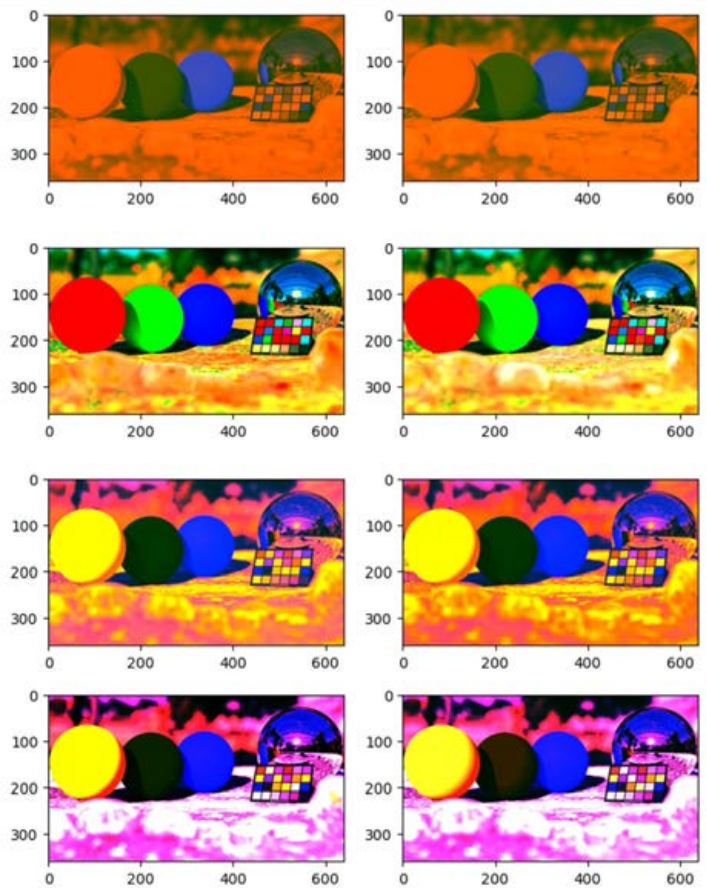
https://github.com/sobotka/Testing_Imagery/blob/main/Sevilla_Park_stone_ring_closeup_linear_sRGB.0001.exr

Visual comparison

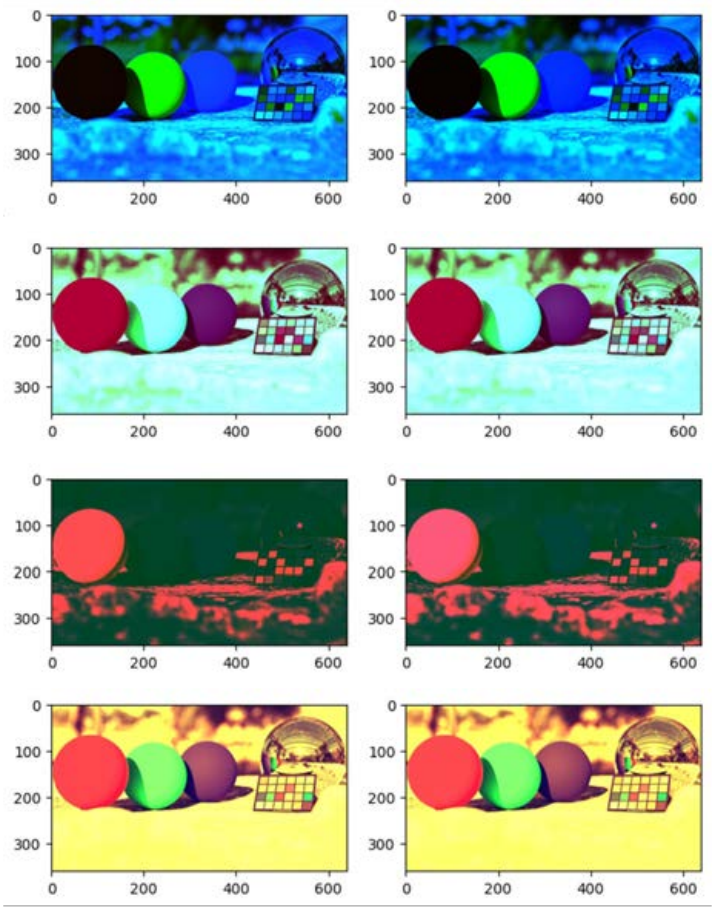


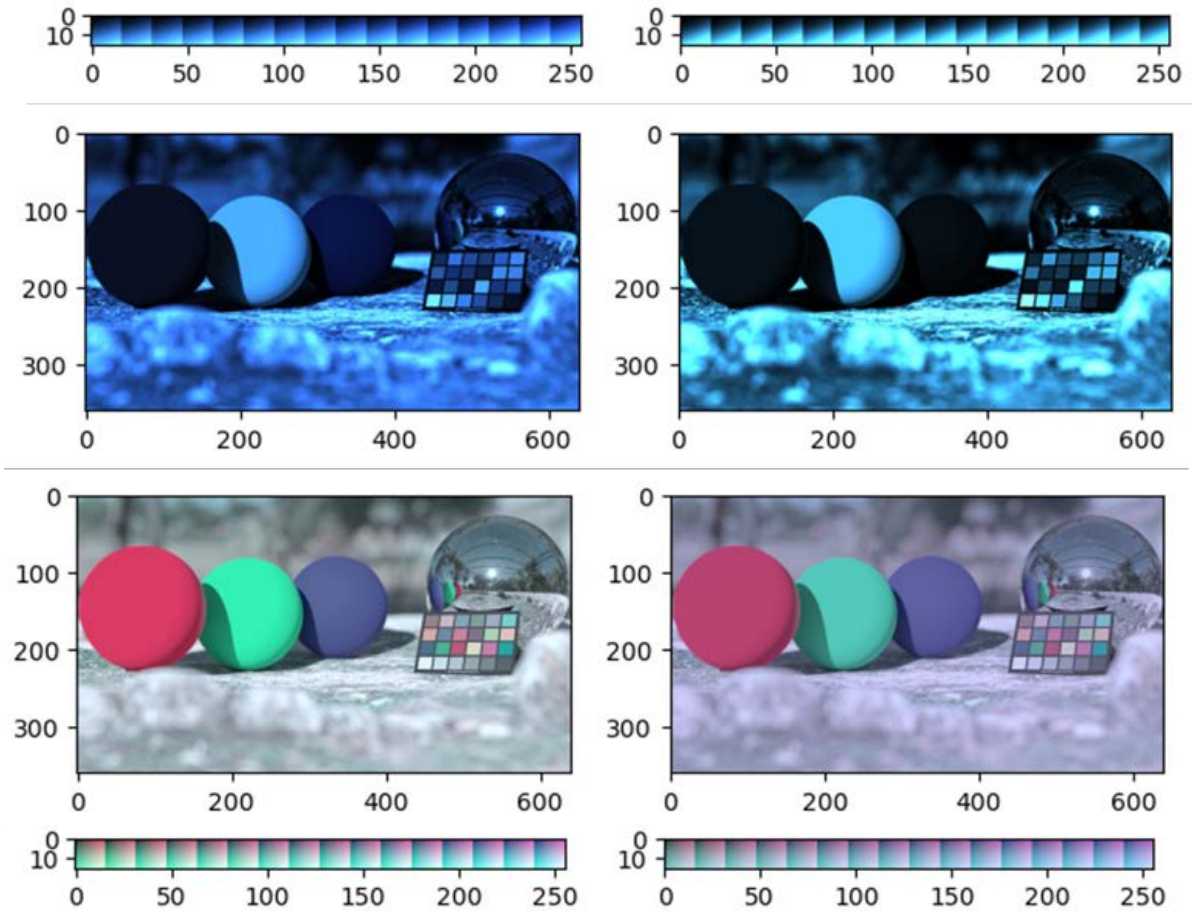


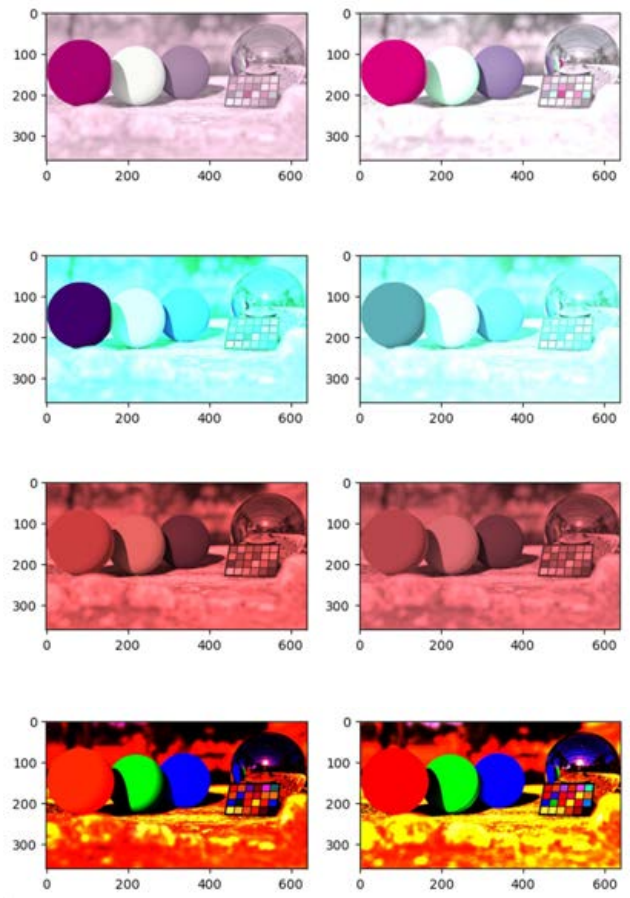
Groundtruth Model



Groundtruth Model

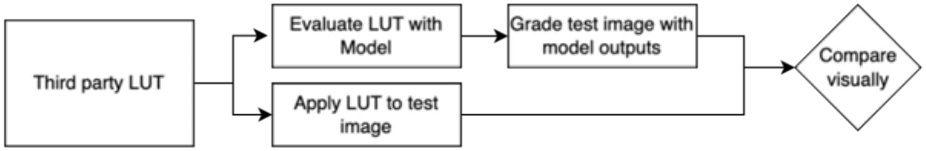






Close but not there yet

Final test



Test image

LUT

Image with LUT (reference)

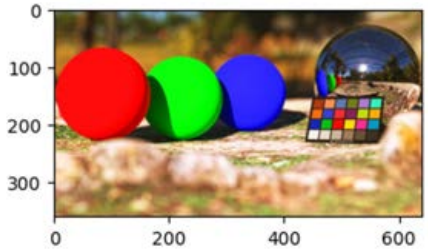
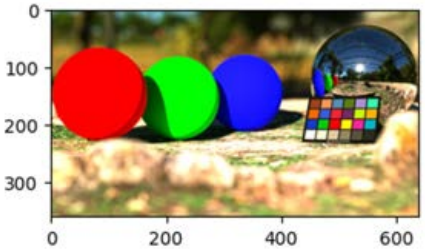
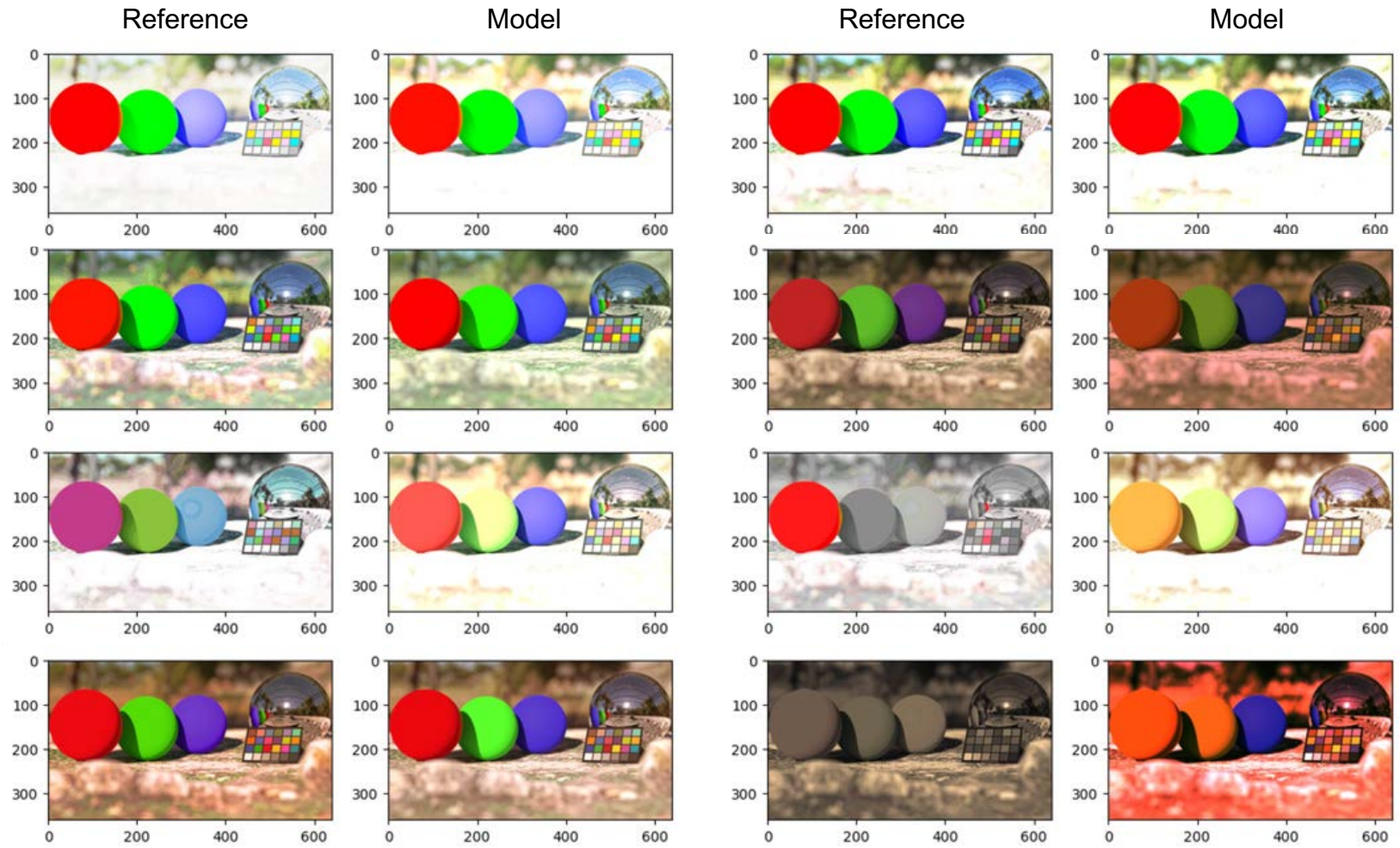


Image with Model grading (output)





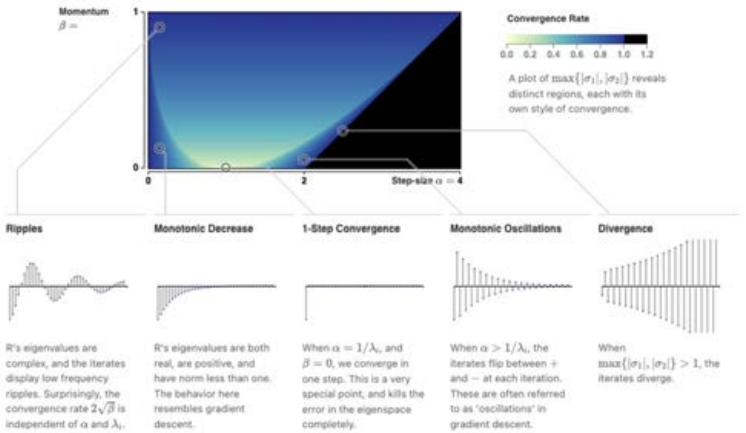
Definitely not there yet

Notes and suggestions for future work

- Find right parameters to create enough variety to cover most of the possible cases. However, too many parameters will create too much noise (all parameters contribute to a pixel component)
- Other color models (f.ex. HSV instead of RGB) are sometimes used when training with images as they might translate better, worth to test out?
- Color grading functions won't be able to replicate all possible input LUTs in a meaningful way, more robust grading operation could still be tested
- For colors in limited range, even slight deviation can sometimes make a big visual impact
    - How universally accurate the results can ever be
    - Benchmarking/loss estimation based on computed visual saliency
- Trained model successfully tested in Unreal (ONNX) - for future work: implement automatic conversion from LUT to PostProcessVolume parameters

Some lessons learned

- With large parameter arrays (LUT = 12 288 floats) and complex initialization functions, verifying correct inputs can be difficult (good visualizations and debug functions help)
- (Free) Google Colab can easily terminate mid training (lose all current training work)
- What convergence should look like



Why Momentum Really Works
https://distill.pub/2017/momentum/

Links

- Jupyter Notebook (Github): https://github.com/A57R4L/LUTModelTraining
  - Test images or third party reference LUTs not included

- Useful test images: https://github.com/sobotka/Testing_Imagery