Final Project - StyleGAN

Project due: 2025. 08. 17. 11:59 PM (KST)

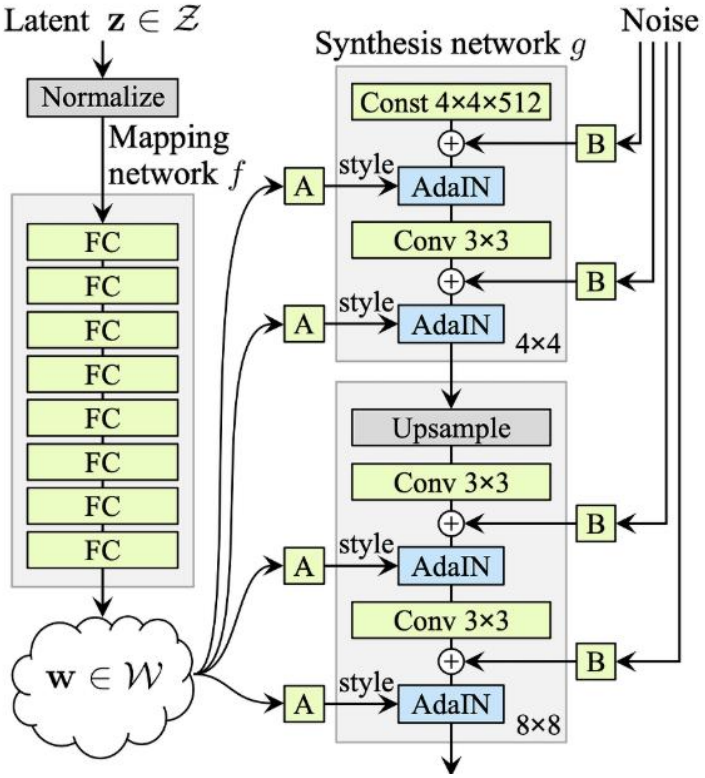
Project Goal

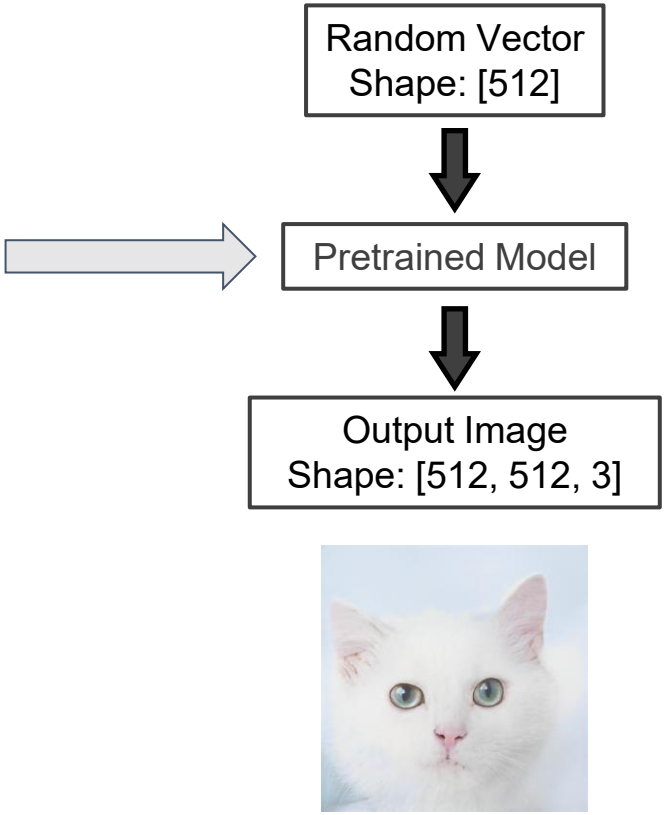
- In this project, we focus on parallelizing and optimizing the inference procedure of the deep learning model
  - Skeleton code: /home/scratch/getp/final-project
  - You can use multiple compute nodes (4 nodes at max) each of which is equipped with 32 AMD EPYC 7452 cores and four NVIDIA RTX 3090 GPUs
  - You should implement 2 different submissions
    - Version 1 (CPU Optimization): Use CPU resources only
    - Version 2 (GPU Optimization): Use all resources including CPU and GPU
  - Due: 2025. 08. 17. 11:59 PM (KST)

- For optimization, you can use Pthread, OpenMP, MPI, AVX and CUDA.
  - Using the external BLAS libraries (like MKL, cuBLAS, cuDNN, etc) is prohibited

Target Model

- StyleGAN model (Generator)
  - Generative Adversarial Network (GAN) that creates realistic images.





Background

Background

- You don’t have to understand all background details
  - You can start optimizing the code by focusing solely on its calculation and memory access patterns

- However, having this knowledge you can try much broader range of optimization techniques.

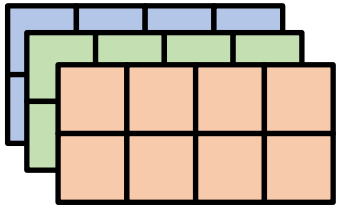
Background - Tensor

- In this project, data is primarily handled in units called tensors.
  - The operations implemented in the provided skeleton code take tensors as both input and output.
  - Definition: `include/tensor.h`, Implementation: `src/tensor.cu`

1D Tensor
(e.g., Vector)

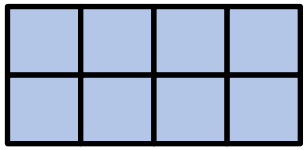
Shape = {4}

3D Tensor
(e.g., RGB image)



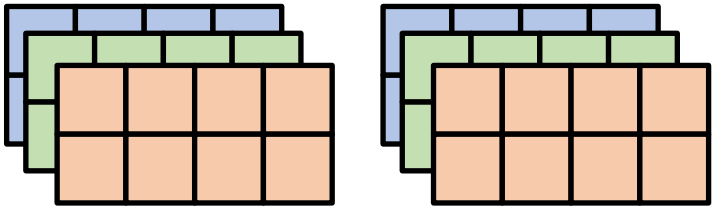
Shape = {3, 2, 4}

2D Tensor
(e.g., Matrix)



Shape = {2, 4}

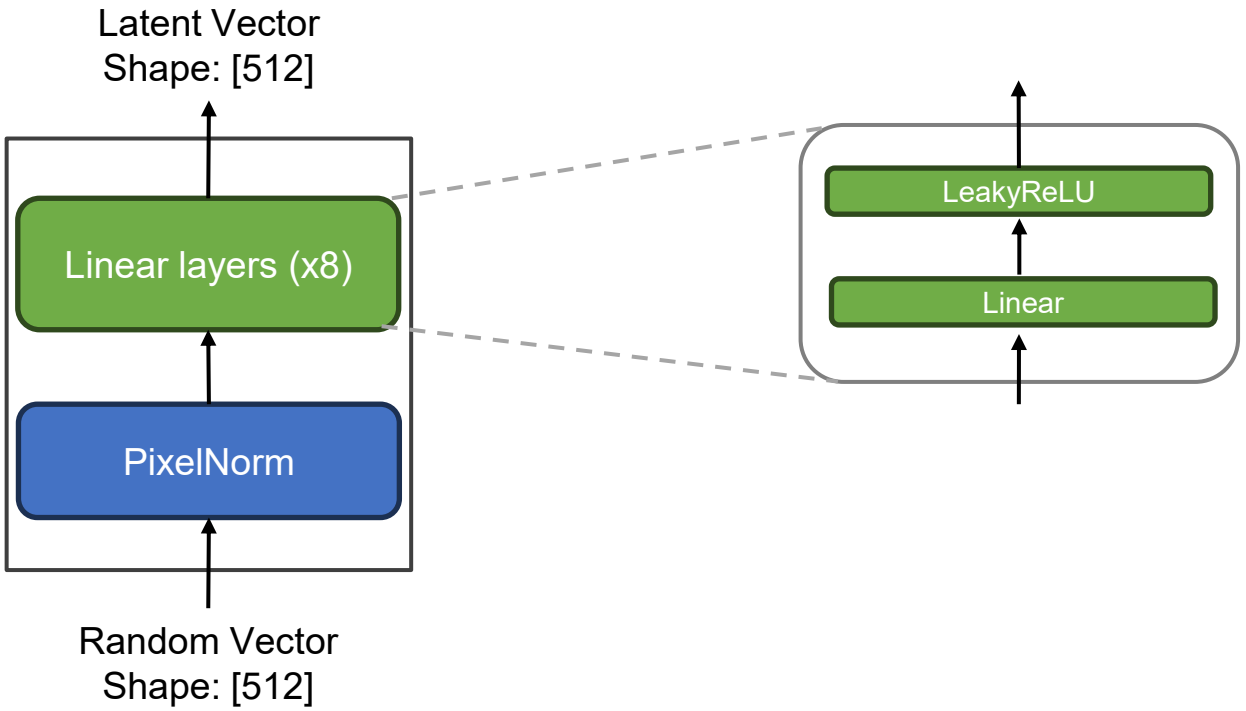
4D Tensor
(e.g., Conv filter)



Shape = {2, 3, 2, 4}

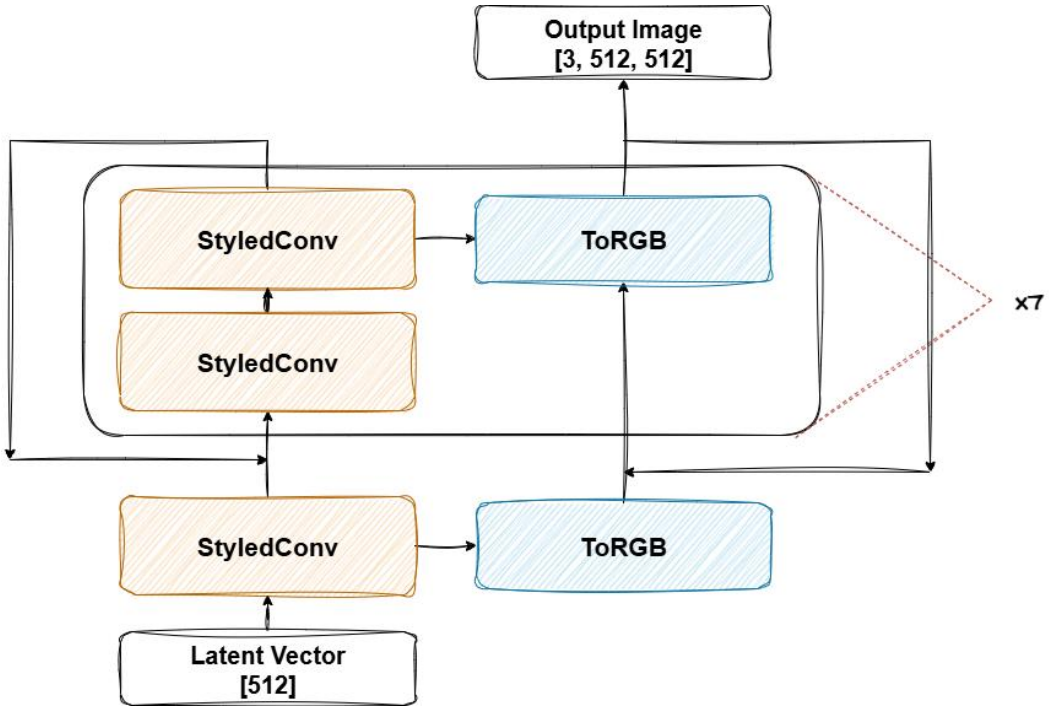
Background - Model

- StyleGAN2 model
  - Definition: `include/model.h`, Implementation: `src/model.cu`



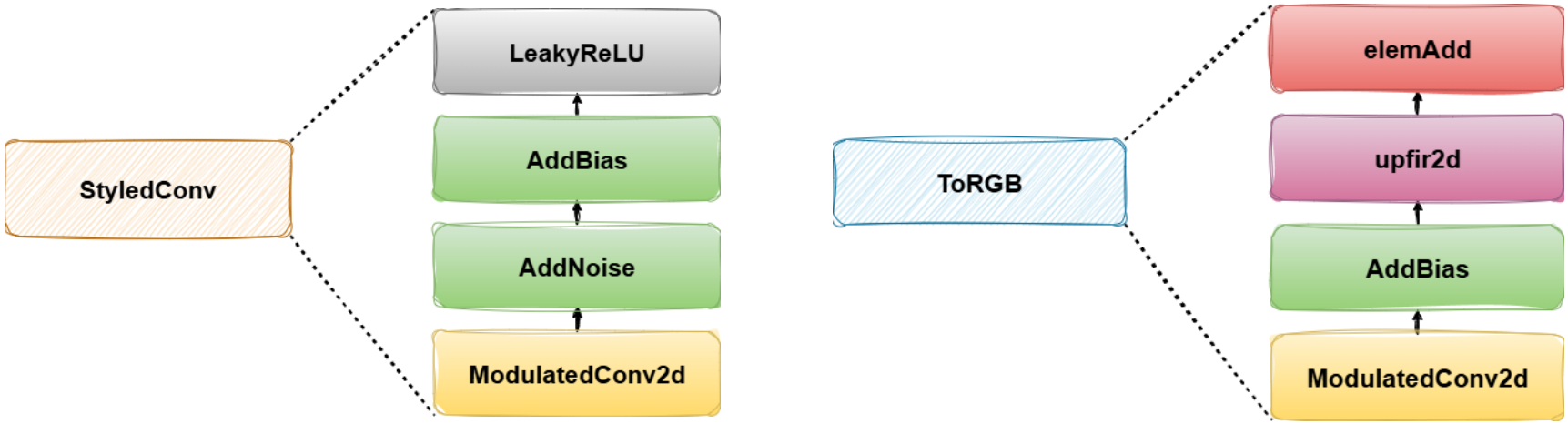
Background - Model (Cont'd)

- StyleGAN2 model
  - Definition: `include/model.h`, Implementation: `src/model.cu`



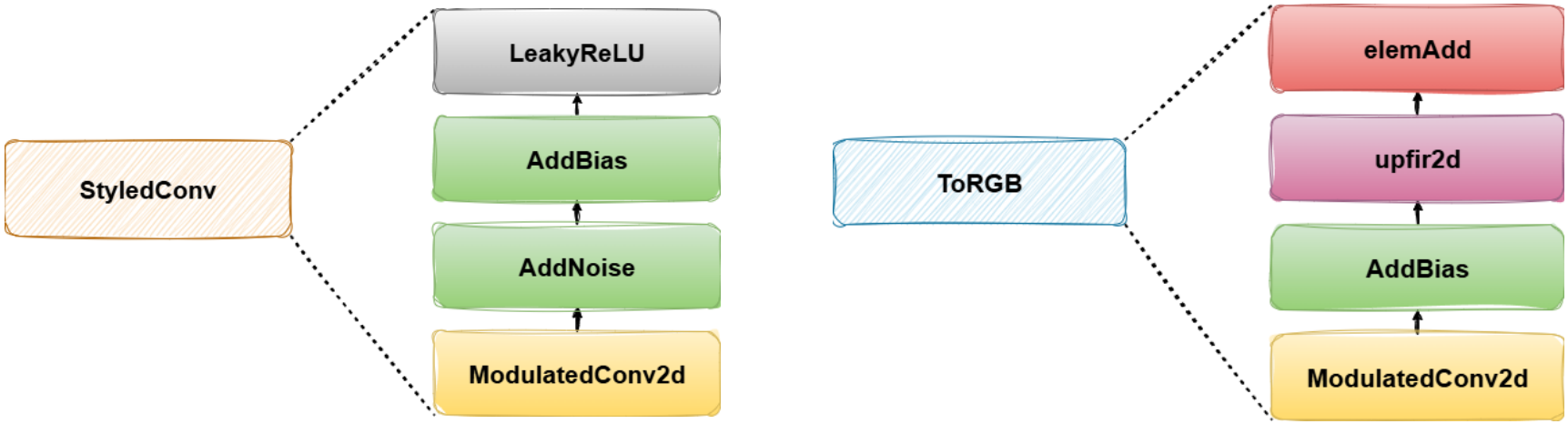
Background - Model (Cont'd)

- StyleGAN2 model
  - Definition: `include/model.h`, Implementation: `src/model.cu`



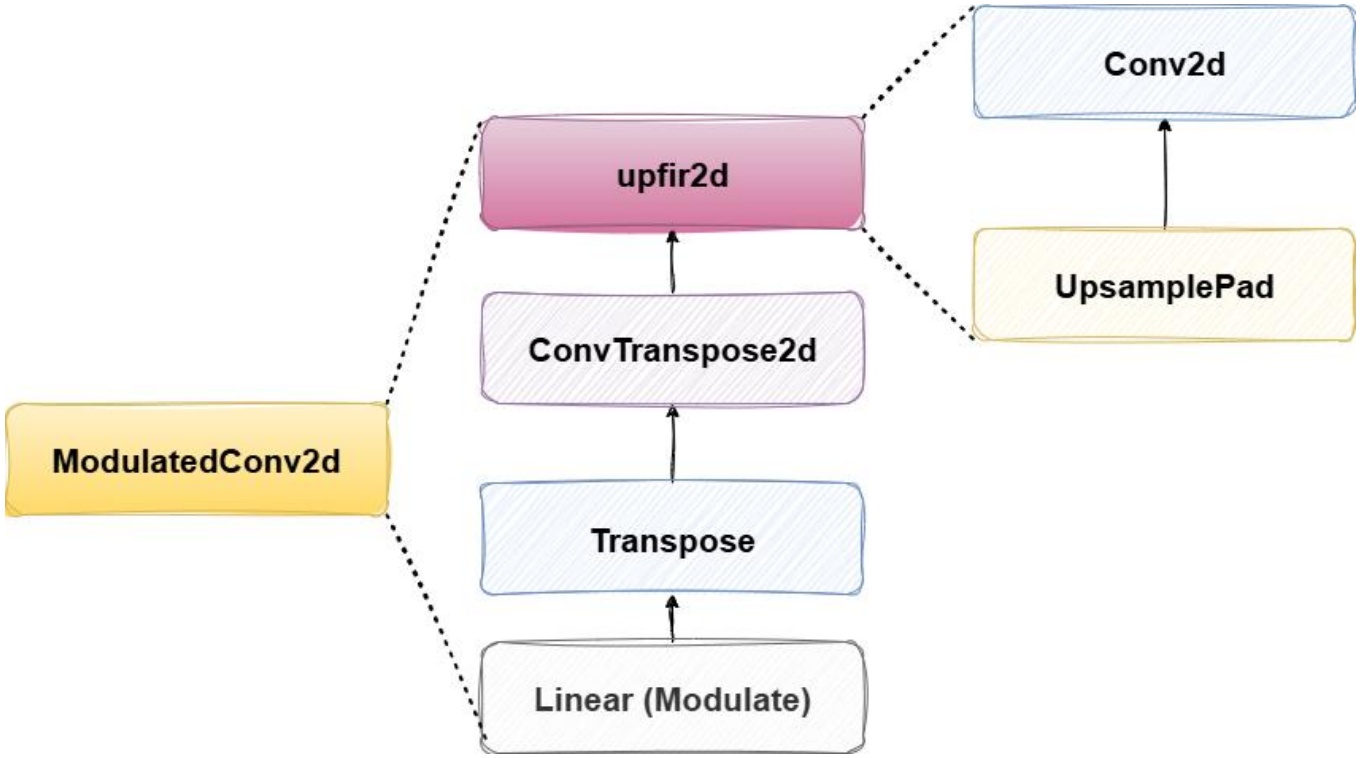
Background - Model (Cont'd)

- StyleGAN2 model
  - Definition: `include/model.h`, Implementation: `src/model.cu`



Background - Model (Cont'd)

- StyleGAN2 model
  - Definition: `include/model.h`, Implementation: `src/model.cu`



Background - Layers

Operations (Layers)

- Definition: `include/layer.h`, Implementation: `src/layer.cu`

1. `PixelNorm`
2. `UpsamplePad`
3. `Conv2d`
4. `ConvTranspose2d`
5. `Linear`
6. `LeakyReLU`
7. `addNoise`
8. `addBias`
9. `elemAdd`

Background - Layers (cont'd)

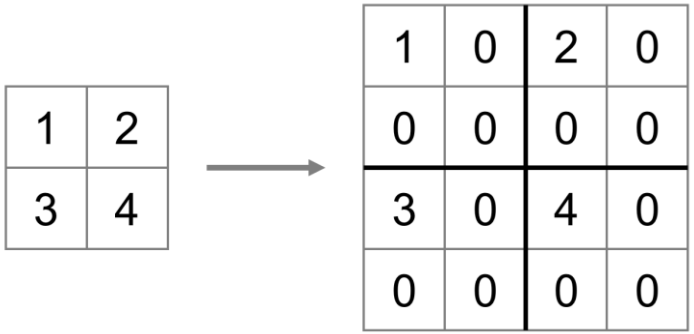
`PixelNorm`

- Normalizes the input tensor along the channel dimension.
- Input & Output
  - `[inout]  in: [N, C]`

Background - Layers (cont'd)

UpsamplePad

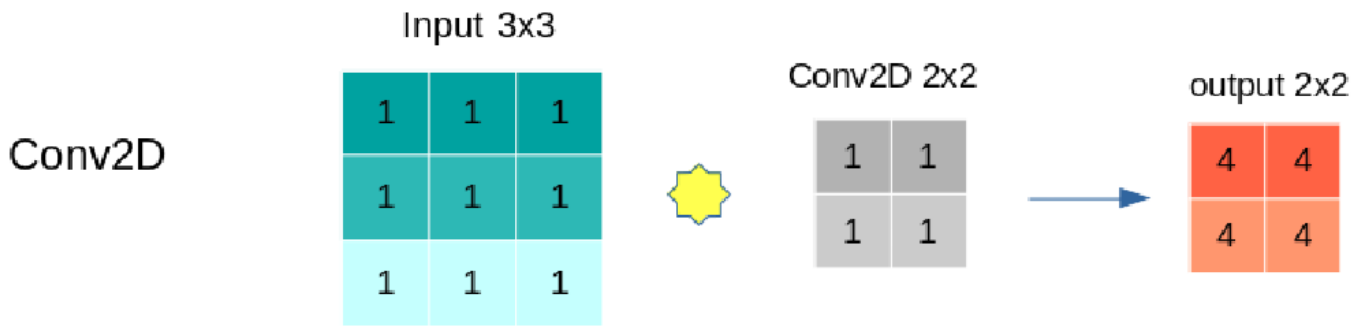
- Upsamples the input tensor and applies padding.
- Input & Output
  - `[in1] in: [N, C, H, W]`
  - `[in2] up: [1]   # scale factor`
  - `[in3] pad0: [1] # padding applied to the top/left sides`
  - `[in4] pad1: [1] # padding applied to the bottom/right sides`
  - `[out] out: [N, C, up · H + pad0 + pad1, up · W + pad0 + pad1]`



Background - Layers (cont'd)

Conv2d

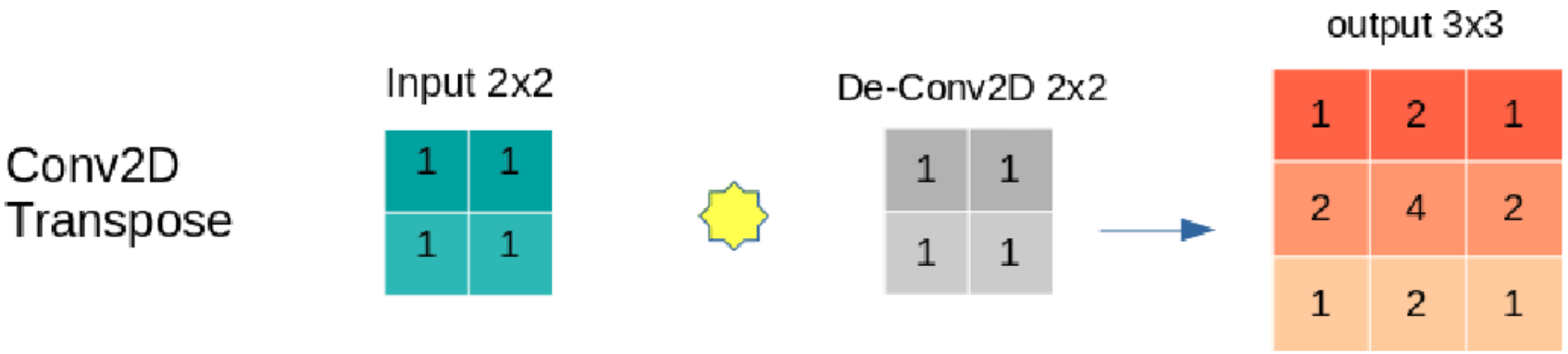
- Applies a 2D convolution over an input tensor
- Input & Output
  - `[in1] in: [N, C, H, W]`
  - `[in2] w: [K, C, R, S]`
  - `[in3] b: [K]`
  - `[out] out: [N, K, OH, OW]`



Background - Layers (cont'd)

ConvTranspose2d

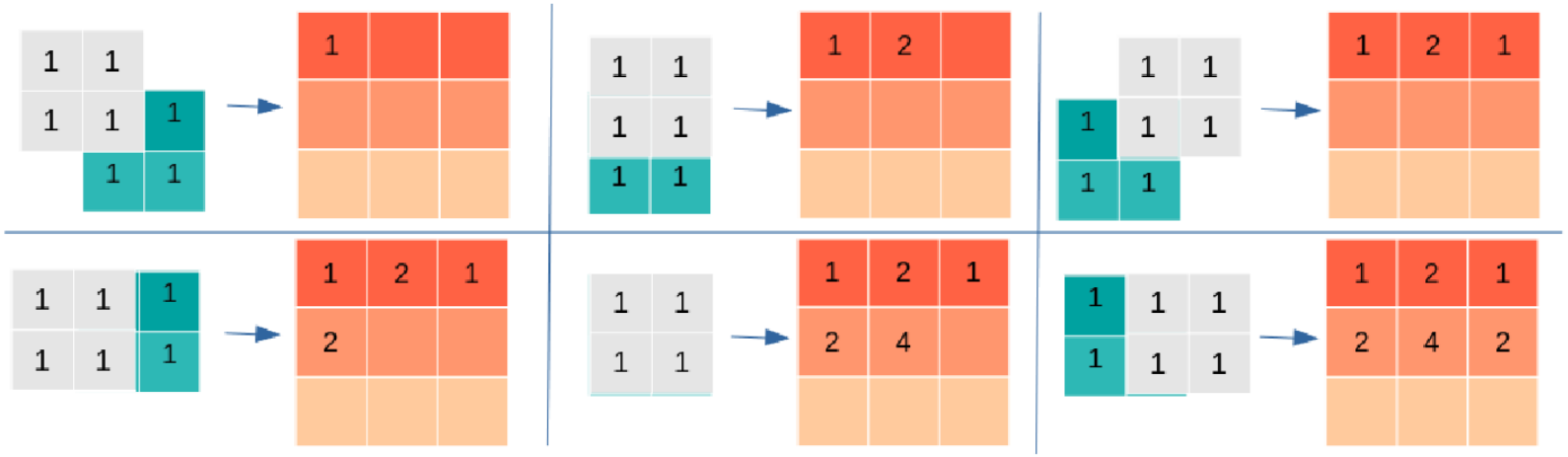
- Performs a 2D transposed convolution (also known as deconvolution)
- Input & Output
  - `[in1] in: [N, C, H, W]`
  - `[in2] w: [C, K, R, S]`
  - `[out] out: [N, K, OH, OW]`



Background - Layers (cont'd)

ConvTranspose2d

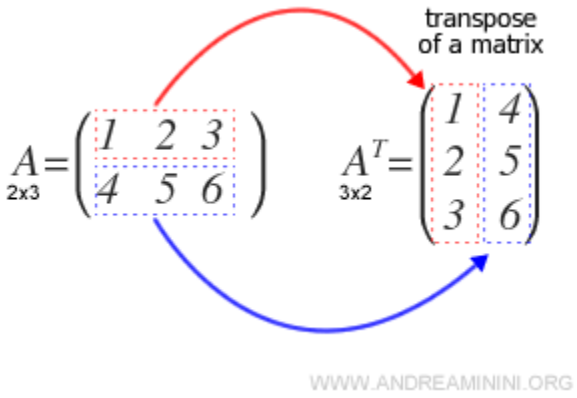
- Performs a 2D transposed convolution (also known as deconvolution)
- Input & Output
  - `[in1] in: [N, C, H, W]`
  - `[in2] w: [C, K, R, S]`
  - `[out] out: [N, K, OH, OW]`



Background - Layers (cont'd)

Transpose

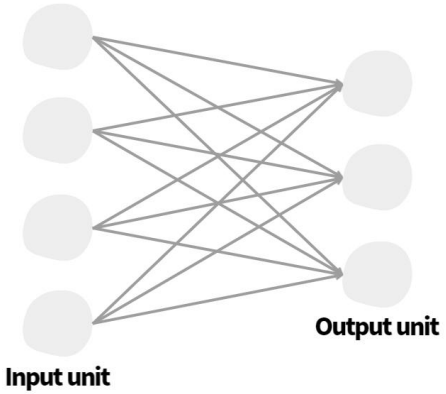
- Swaps the first and second dimensions of the input tensor.
- Input & Output
  - `[in] in: [N, C, H, W]`
  - `[out] out: [C, N, H, W]`



Background - Layers (cont'd)

`Linear`

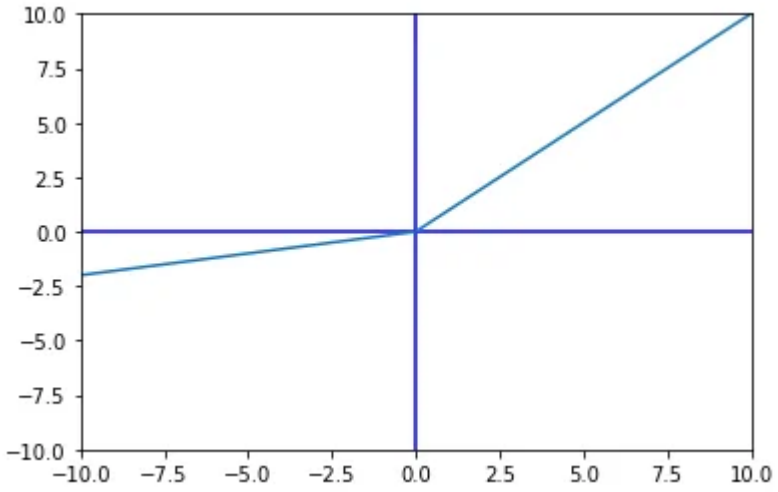
- Applies a linear transformation to the input
- Input & Output
  - `[in1] in: [M, K]`
  - `[in2] w: [N, K]`
  - `[in3] b: [N]`
  - `[out] out: [M, N]`



Background - Layers (cont'd)

LeakyReLU

- An activation function that allows a small, non-zero gradient when the input is negative
- Input & Output
  - `[inout] in: [N]`



Background - Layers (cont'd)

addNoise

- Adds spatial noise to each channel of the input tensor in-place.
- Input & Output
  - `[inout]  inout: [N, C, H, W]`
  - `[in]     noise: [H, W]`

Background - Layers (cont'd)

addBias

- Adds a per-channel bias to the input tensor in-place.
- Input & Output
  - `[inout]  inout: [N, C, H, W]`
  - `[in]     bias: [C]`

Background - Layers (cont'd)

`elemAdd`

- Performs element-wise addition of two tensors in-place.
- Input & Output
  - `[inout]  in: [N, C, H, W]`
  - `[in]     addend: [N, C, H, W]`

Skeleton Code & How to Run

Skeleton Code

- final-project
  - data/
  - include/
    - tensor.h # Tensor struct definition
    - layer.h # Layer declarations
    - model.h* # Model declarations
  - src/
    - tensor.cu # Tensor struct implementation
    - layer.cu # Layer implementation (core computation logic)
    - model.cu # Model implementation
    - main.cpp* # Driver code (entry point)
  - tools/
  - Makefile* # Build script
  - run.sh # Run script

*DO NOT MODIFY THESE FILES.

Example Run

- Build with the make command
  - `$ make`

```
getp35@login:~/final-project$ make
mkdir -p obj
g++ -std=c++14 -O3 -Wall -march=native -mavx2 -mfma -mno-avx512f -fopenmp -I/usr/local/cuda/include -Iincl
ude -c -o obj/main.o src/main.cpp
/usr/local/cuda/bin/nvcc  -Xcompiler=-std=c++14 -arch=sm_75  -Xcompiler=-O3 -arch=sm_75  -Xcompiler=-Wall
-arch=sm_75  -Xcompiler=-march=native -arch=sm_75  -Xcompiler=-mavx2 -arch=sm_75  -Xcompiler=-mfma -arch=s
m_75  -Xcompiler=-mno-avx512f -arch=sm_75  -Xcompiler=-fopenmp -arch=sm_75  -Xcompiler=-I/usr/local/cuda/i
nclude -arch=sm_75  -Xcompiler=-Iinclude -arch=sm_75 -c -o obj/model.o src/model.cu
/usr/local/cuda/bin/nvcc  -Xcompiler=-std=c++14 -arch=sm_75  -Xcompiler=-O3 -arch=sm_75  -Xcompiler=-Wall
-arch=sm_75  -Xcompiler=-march=native -arch=sm_75  -Xcompiler=-mavx2 -arch=sm_75  -Xcompiler=-mfma -arch=s
m_75  -Xcompiler=-mno-avx512f -arch=sm_75  -Xcompiler=-fopenmp -arch=sm_75  -Xcompiler=-I/usr/local/cuda/i
nclude -arch=sm_75  -Xcompiler=-Iinclude -arch=sm_75 -c -o obj/tensor.o src/tensor.cu
/usr/local/cuda/bin/nvcc  -Xcompiler=-std=c++14 -arch=sm_75  -Xcompiler=-O3 -arch=sm_75  -Xcompiler=-Wall
-arch=sm_75  -Xcompiler=-march=native -arch=sm_75  -Xcompiler=-mavx2 -arch=sm_75  -Xcompiler=-mfma -arch=s
m_75  -Xcompiler=-mno-avx512f -arch=sm_75  -Xcompiler=-fopenmp -arch=sm_75  -Xcompiler=-I/usr/local/cuda/i
nclude -arch=sm_75  -Xcompiler=-Iinclude -arch=sm_75 -c -o obj/layer.o src/layer.cu
cc -std=c++14 -O3 -Wall -march=native -mavx2 -mfma -mno-avx512f -fopenmp -I/usr/local/cuda/include -Iinclu
de -o main obj/main.o obj/model.o obj/tensor.o obj/layer.o -pthread -L/usr/local/cuda/lib64 -lstdc++ -lcud
art -lm -lnvToolsExt
```

Example Run

- Run the execution script run.sh
  - `$ ./run.sh -h`

```
getp35@login:~/final-project$ ./run.sh -h
 Usage: ./main [-i 'pth'] [-p 'pth'] [-o 'pth'] [-a 'pth'] [-n 'num_images'] [-v] [-w] [-h]
 Options:
  -i: Input binary path (default: /getp/styles.bin)
  -p: Model parameter path (default: /getp/param.bin)
  -o: Output binary path (default: ./data/outputs.bin)
  -a: Answer binary path (default: /getp/answers.bin)
  -n: Number of inputs (default: 1)
  -v: Enable validation (default: OFF)
  -w: Enable warm-up (default: OFF)
  -h: Print manual and options (default: OFF)
```

Example Run

- Example of running the skeleton code
  - `$ ./run.sh -n 1 -w -v`

```
getp35@login:~/final-project$ ./run.sh -n 1 -v

================================================
 Model
------------------------------------------------
 Validation: ON
 Warm-up: OFF
 Number of images: 1
 Input path: /getp/styles.bin
 Model parameter path: /getp/param.bin
 Answer binary path: /getp/answers.bin
 Output binary path: ./data/outputs.bin
================================================

Initializing inputs and parameters...Done!
Generating images...Done!
Elapsed time: 13.244419 (sec)
Throughput: 0.075504 (images/sec)
Finalizing...Done!
Saving outputs to ./data/outputs.bin...Done!
Validating...PASSED!
```

Example Run (Optional)

- Can check the result images using the provided Python program in the skeleton code.
  - `$ python3 tools/bin2img.py data/outputs.bin --all`

```
getp35@login:~/final-project$ python3 tools/bin2img.py data/outputs.bin --all
Converting 1 samples...
Converting sample 1/1...
File size: 3145728 bytes
Expected size per sample: 3145728 bytes
Number of samples detected: 1
Data range: [-0.961421, 1.154710]
Image saved to: data/outputs_sample_000000.png
```

Project Rules & Restrictions

Skeleton Code (cont'd)

- Files You Must Not Modify
  - styles.bin, answers.bin: Input and ground truth data files
  - param.bin: Pretrained model parameters
  - model.h, main.cpp, Makefile: Core files that must remain unchanged

- Files You Can Modify and Should Submit
  - tensor.h, tensor.cu
  - layer.h, layer.cu
  - model.cu
  - run.sh: You may edit this script to add or adjust program execution options as needed.
    - The project will be evaluated using the command: ./run.sh -v
    - Before submission, make sure to modify run.sh using the best-performing configuration (e.g., with the -n option)

Restrictions

- Modifications to the program logic or model architecture are **Not Allowed**.

- Examples of Allowed Modifications
  - Changing memory layout, Reordering loop structures, Adding padding data or operations, Applying operator fusion, etc

- Examples of Disallowed Modifications:
  - Performing model inference outside the generate function
  - Replacing the model with a different one or using a different algorithm that produces the same output

- If you are unsure whether a modification is allowed, please consult the TA.

Submission & Evaluation

Submission & Evaluation Criteria

- Due
  - 2025. 08. 17. 11:59 PM (KST)
  - Late submissions will not be accepted and will result in disqualification.

- Submission Files (for each version)
  - tensor.h, tensor.cu, layer.h, layer.cu, model.cu, run.sh
  - report.pdf
    - Briefly describe the optimizations you applied.
    - Include the following
      1. Performance (Throughput) measured by yourself
      2. A screenshot of the performance measurement output

Submission & Evaluation Criteria (cont'd)

- Use the getp-submit utility to submit your files.
  - Command: `$ getp-submit submit **final-project-xpu** <filename>`
    - E.g., `$ getp-submit submit **final-project-cpu** run.sh`
    - E.g., `$ getp-submit submit **final-project-gpu** report.pdf`
    - ...
  - Check Submission Status: `$ getp-submit status`

- Evaluation Criteria
  - The score is out of a maximum of 100 points.
    - E.g. 100 - min(80, log2(1st_speed/your_speed) * 40)
  - Your final performance will be based on the Throughput (images/sec) value reproduced by the TA using your submitted files.
  - You may freely choose the number of images to generate (maximum: 1024).
  - If the output is incorrect (i.e., fails validation), you will receive zero points.

Update History

History

- 0729: initial release