

**Can you generate Block Diagram for this**

Here are the comprehensive block diagrams for the CycleGAN architecture from your code:

**1. Overall CycleGAN Architecture**

![CycleGAN architecture block diagram showing dual generators, discriminators, and loss components]



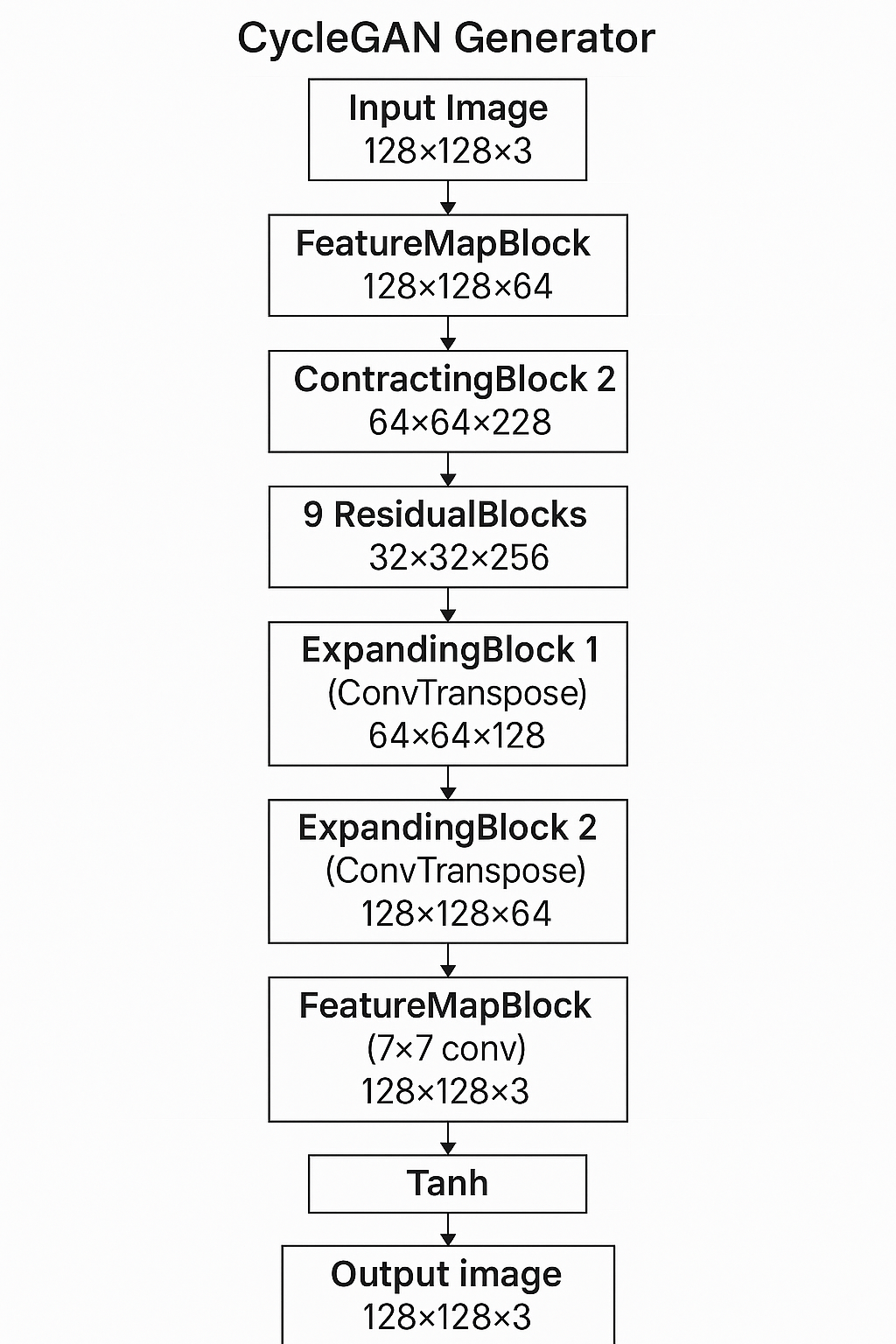
CycleGAN architecture block diagram showing dual generators, discriminators, and loss components

This diagram shows the complete CycleGAN system with:

* **Two Generators**: G\_AB (Domain A → B) and G\_BA (Domain B → A)
* **Two Discriminators**: D\_A and D\_B for evaluating real vs fake images
* **Cycle Consistency**: Images translated A→B→A should match original A
* **Loss Components**: Adversarial loss, cycle consistency loss, and identity loss

**2. Generator Architecture (U-Net with Residual Blocks)**

![CycleGAN Generator detailed architecture with ResidualBlocks and U-Net structure]



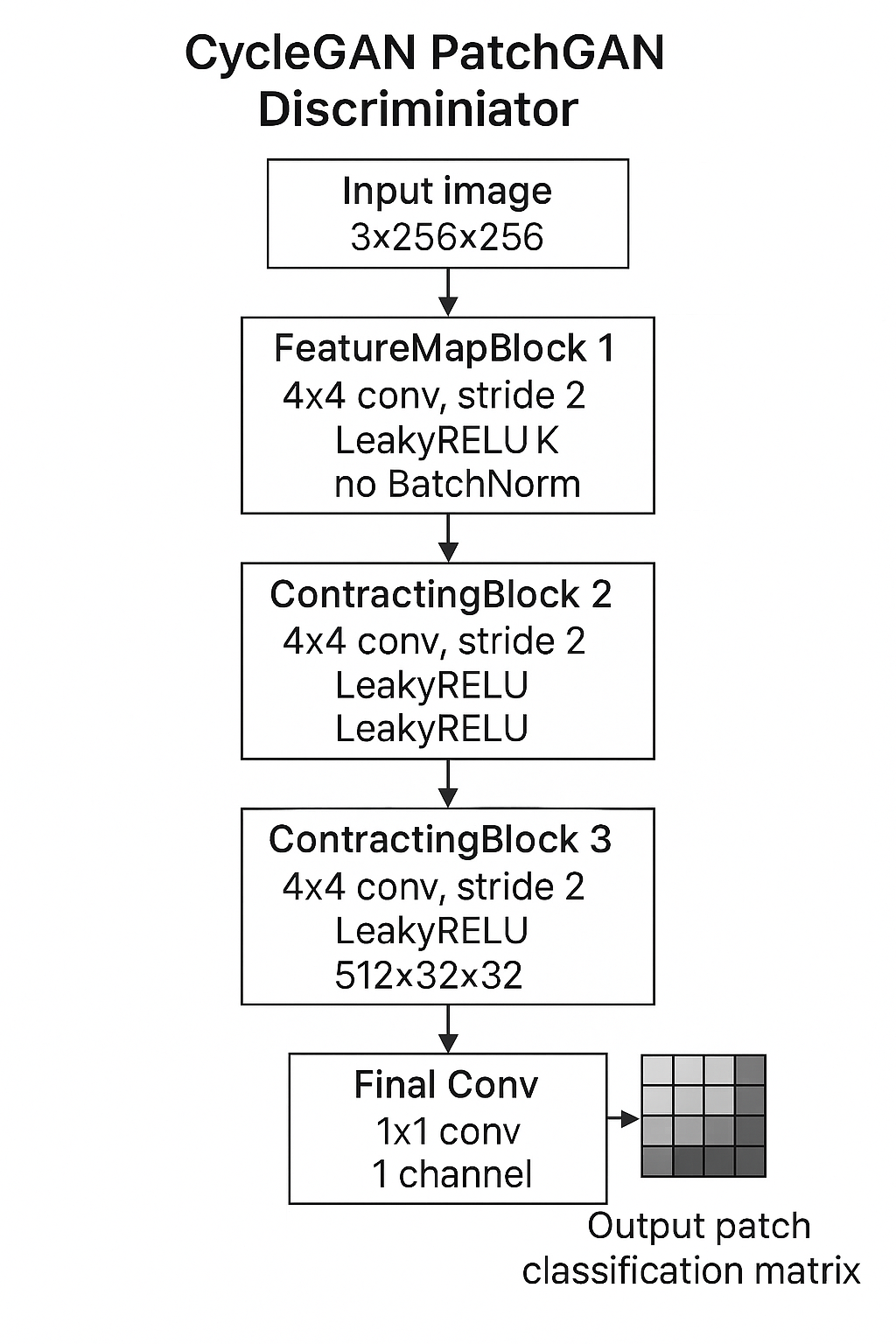
CycleGAN Generator detailed architecture with ResidualBlocks and U-Net structure

The generator uses a U-Net architecture with:

* **Encoder**: FeatureMapBlock + 2 ContractingBlocks (downsampling)
* **Bottleneck**: 9 ResidualBlocks for feature transformation
* **Decoder**: 2 ExpandingBlocks (upsampling) + output FeatureMapBlock
* **Skip connections** within ResidualBlocks for gradient flow

**3. PatchGAN Discriminator Architecture**

![CycleGAN PatchGAN Discriminator architecture showing progressive downsampling]



CycleGAN PatchGAN Discriminator architecture showing progressive downsampling

The discriminator is a PatchGAN that:

* **Processes image patches** rather than whole images
* **Progressive downsampling** through ContractingBlocks
* **Outputs a matrix** where each element classifies a patch as real/fake
* **Uses InstanceNorm** (except first layer) and LeakyReLU activations

**Key Features of Your CycleGAN Implementation:**

* **Instance Normalization** instead of Batch Normalization
* **Reflect padding** to avoid border artifacts
* **MSE loss** for adversarial training (LSGAN)
* **L1 loss** for cycle consistency and identity
* **Lambda weights**: λ\_identity=0.1, λ\_cycle=10 to balance different loss components

This architecture enables unpaired image-to-image translation between your CT scan classes (Normal, Benign, Malign) while preserving important structural information through cycle consistency.

⁂