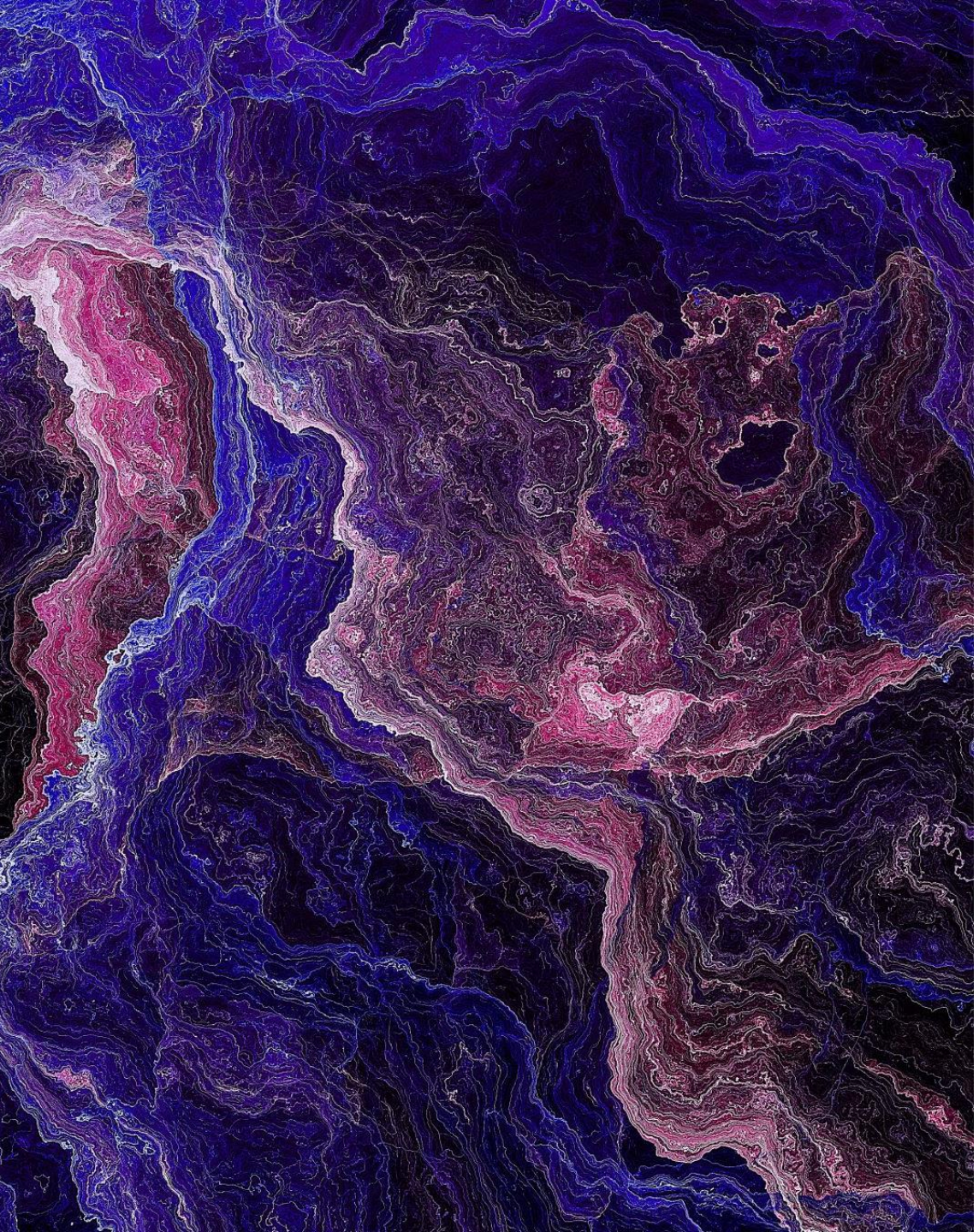
Resolution Revolution: Unleashing AI For Superior Image Quality

Authors: Amer Zuher Alriyahy & Hisham Maher Sunjaq

Supervisor: Dr. Mohmmad Al-Musaideen



Introduction

- Deep learning has revolutionized fields like computer vision and AI.
- Significant advances in image enhancement and restoration.
- Focus: utilizing deep learning to improve image quality by reducing noise, enhancing resolution, and improving aesthetics.

Motivation And Background

- **Demand for High-Quality Images**:
  - Increasing need for high-quality images in digital media.
  - Issues with images captured in poor conditions or over constrained networks.

- **Limitations of Traditional Methods**:
  - Traditional techniques like interpolation and deblurring have limitations.
  - Deep learning, especially CNNs (Convolutional Neural Network) and GANs (Generative Adversarial Network), offers superior performance by learning complex patterns.

Applications

NVIDIA's DLSS 3.5.

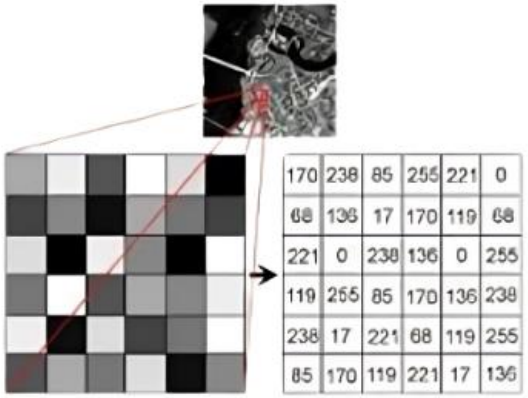
enhance the resolution of satellite images.

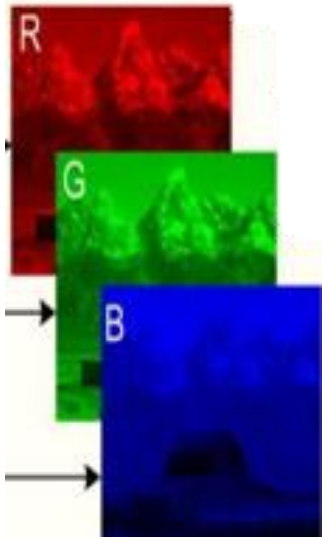
- **Medical imaging**:
  - Enhances clarity of MRI and CT scans for better diagnostics.
- **Satellite imagery**:
  - Companies like Maxar technologies use deep learning for higher resolution images.
- **Consumer electronics**:
  - Ai-enhanced cameras in smartphones and AI tools in photo editing software.
- **Gaming industry**:
  - Nvidia's DLSS (Deep Learning Super Sampling): real-time upscaling for improved visuals and frame rates.

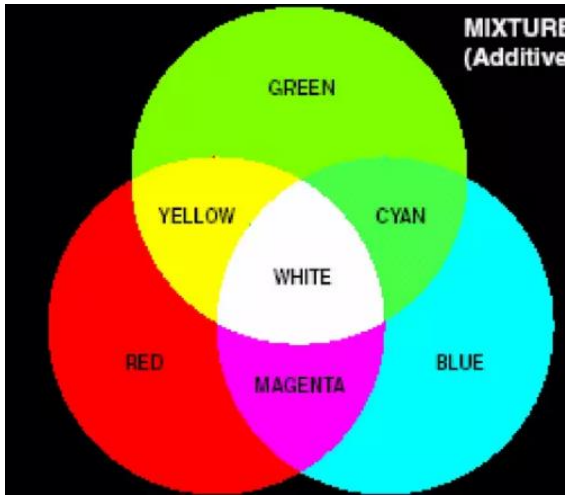
RGB Images

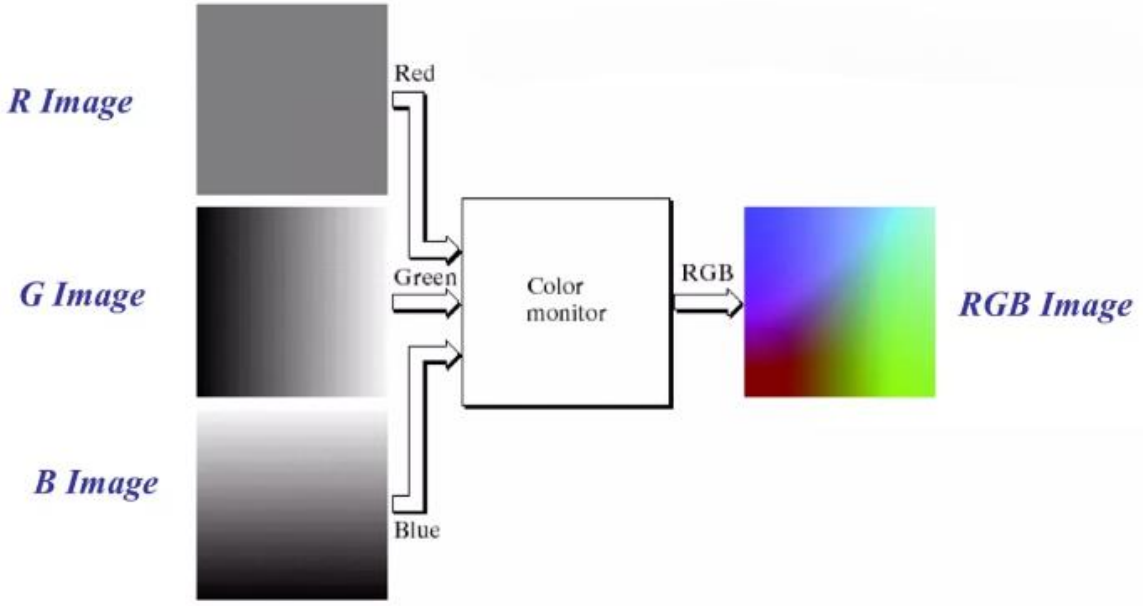
- Colors: Red(R), Green(G) and Blue(B) are referred as the primary colors and when mixed with various intensity proportions, can produce all visible colors.

(0-255)









Dataset Used: DIV2K

SAMPLE OF DATASET

•**Details**:

- **Name**: DIV2K (DIVerse 2K)
- **Resolution**: 2K (2048 x 1024 pixels)
- **Total Images**: 900
- **Split**: 720 training images (80%), 180 validation images (20%)
- **Content**: Natural landscapes, urban environments, everyday objects

Preprocessing

Preprocessing Steps

- **Importance of preprocessing**:
  - Ensures data quality and consistency.
  - Maximizes model performance.
- **Steps** :
  1. Image selection
  2. Image resizing
  3. Train-test split
  4. Image scaling
  5. Convert RGB to YUV and extract luminance
  6. Downscale training images

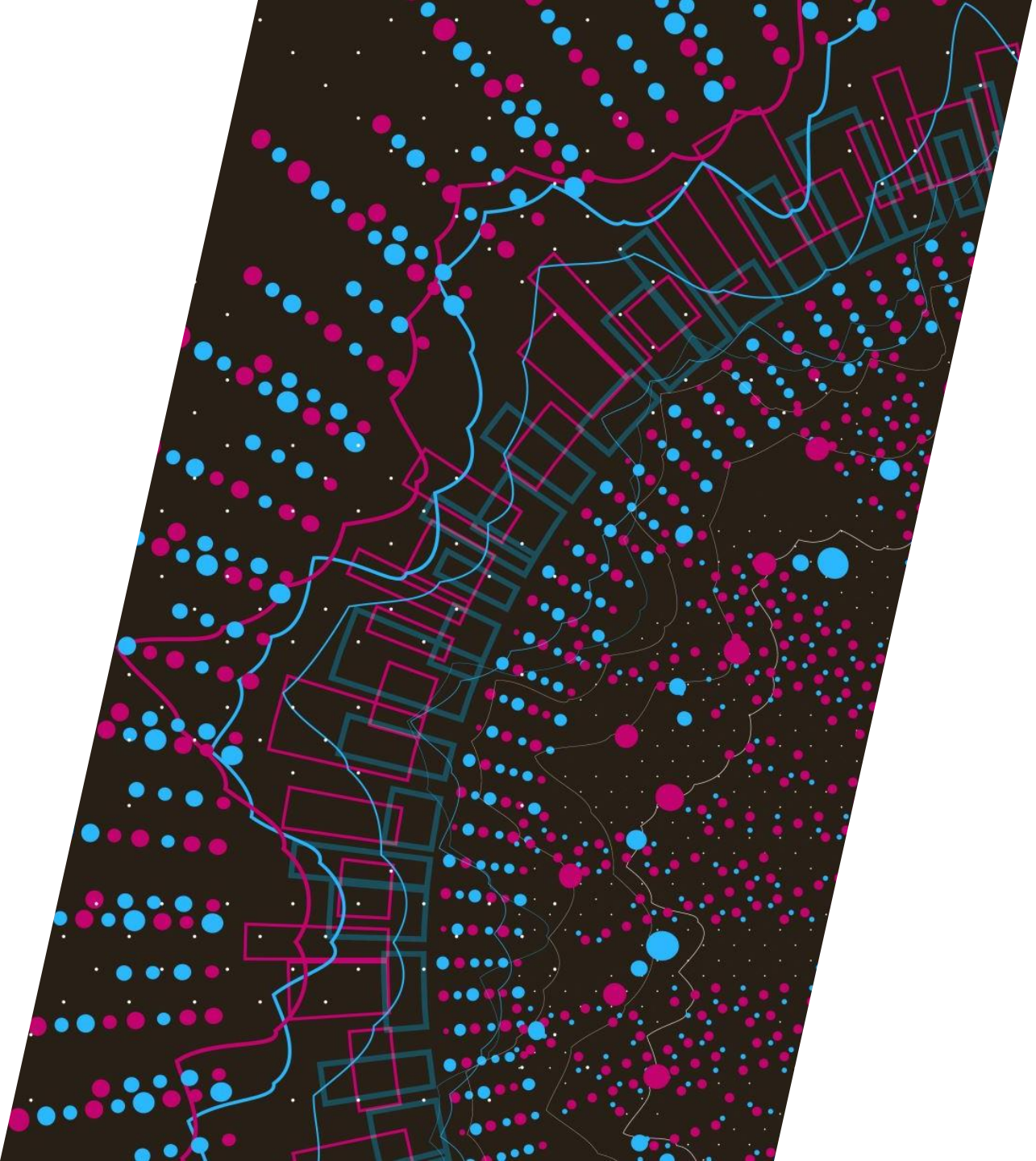


Image Selection

- **Remove 1-channel images:** delete images that do not have 3 color channels (i.E., They are not RGB).
- **Remove low-resolution images:** delete images that are below a specified resolution.
- Ensures sufficient resolution and correct color format.

Image Resizing And Train-test Split

- **Image resizing**:

  Resize all images to **498x300** pixels.

  Ensures uniform size for efficient training and comparison.

- **Train-test split**:
  - 80% for training, 20% for validation.
  - Critical for evaluating model performance.

Image Scaling

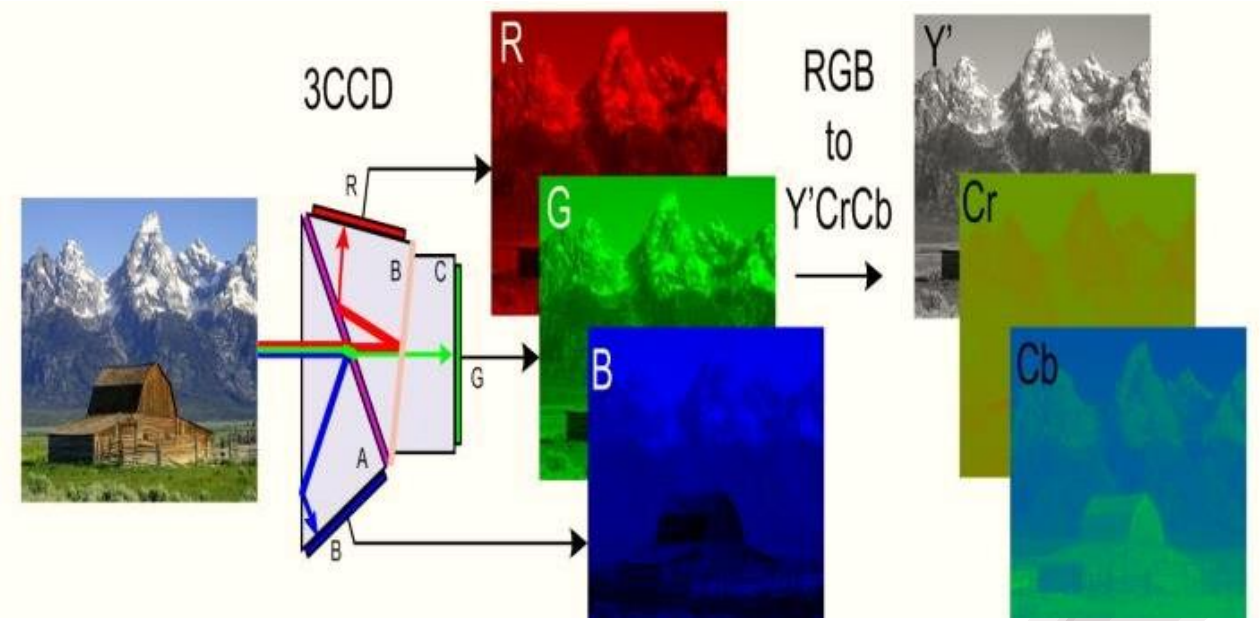
- **Image scaling**:
  - Normalizes pixel values to [0, 1].
  - Benefits: improved convergence, reduced numerical instability, consistency.
  - Scaling before conversion helps in maintaining consistency and accuracy in the transformation process.

$$\text{normalized\_pixel} = \frac{\text{pixel\_value}}{255.0}$$

Convert RGB To YUV And Downscale Training Images

- **Convert RGB to YUV**:
  - Separates luminance (Y) from chrominance (U, V).
  - Focus on brightness details for better image processing.
  - Y (Luminance): Range: 0 to 255 Represents the brightness or luminance of the color
  - U (Chrominance Blue) and V (Chrominance Red):

    Range: -128 to 127 Represents the color information
  - With Scaling: Y: [0, 1], U and V close to [-0.5, 0.5]
- **Downscale Training Images**:
  - Resize from 498x300 to 166x100 pixels.
  - Simulates low-resolution input for model training.



Y (Luma):

$$Y = 0.299R + 0.587G + 0.114B$$

MODEL

Model Architecture Overview

- **Model type**: convolutional neural network (CNN)
- **Purpose**: designed for image super-resolution.
- **Key components**:
  - Convolutional layers to enhance image resolution.
  - Residual dense blocks (RDB) for improved feature extraction and image reconstruction.

Layer details and configuration

in general:

- activation function: relu
- kernel initializer: orthogonal: weights and biases: default orthogonal w= based on the random seed ,b=0

conv2d layers:

- Filters per layer: 64, 64, RDB(64) , 32, RDB(32), 16 , RDB(16) ,9.
- Kernel sizes per layer: 5, 3, 3, 3, 3.
- Num of all hidden layers (**convolutional),** include RDBs: 17

residual dense blocks (RDB**)**:

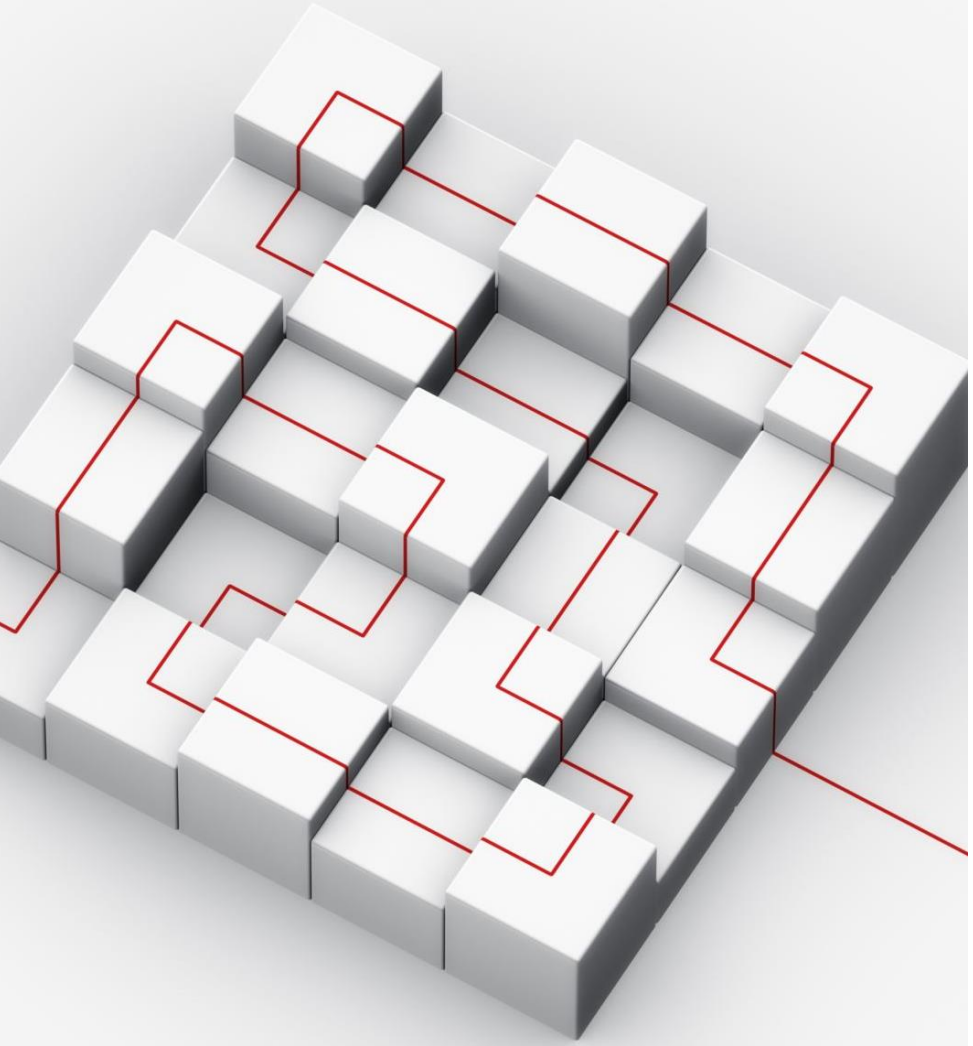
- filters per RDB call: 64, 32, 16
- kernel sizes per rdb: 3, 3, 3 ,1
- num of hedin layers per **rdb**: 4

depth-to-space:

- used for upscaling image resolution

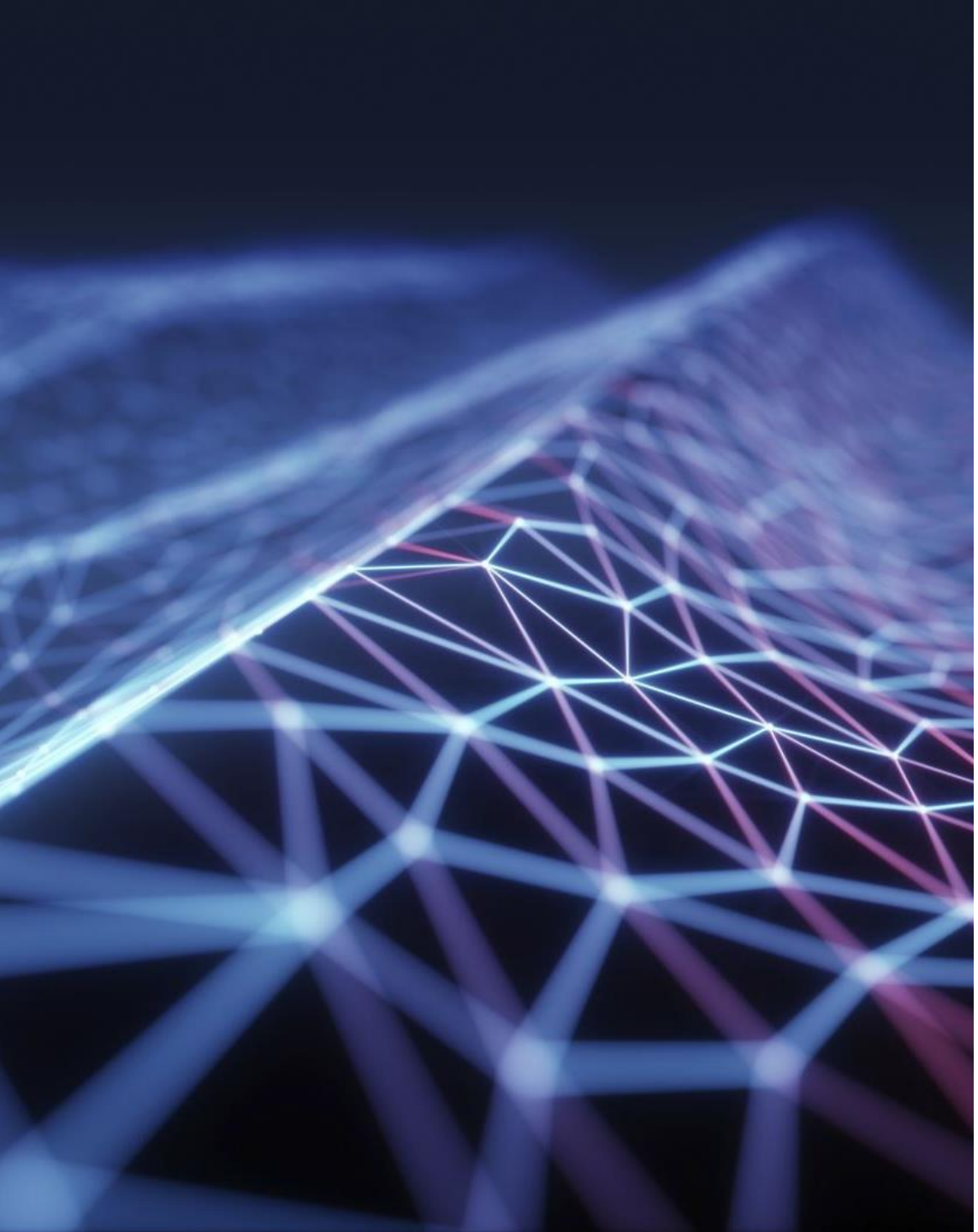
Layer Details And Configuration

- **Hyperparameters**:
  - **Batch Size**: 32
  - **Number of Epochs**: 100.
  - **Optimizer**: Adam Optimizer
  - **Learning Rate**: Adam's Default L.R (0.001)
  - **Loss Function**: MSE, (mean squared error).
  - **Regularization Parameters**: Early stopping:
    - Patience of 10, min_delta of 0.0001.



Residual Dense Block (RDB)

- **Function**:
  - **Combines** outputs from multiple layers and passes through another layer.
  - **Optimize** :RDBs improve feature reuse, gradient flow, and parameter efficiency, enhancing learning and performance across various architectures.
- **Benefits**:
  - **Efficient feature reuse**: RDBs facilitate the reuse of features across multiple layers within the block.
  - **Improved gradient flow**: the skip connections in RDBs aid in the flow of gradients during training, alleviating the vanishing gradient problem.
  - **Parameter efficiency**: RDBs achieve high expressiveness with fewer parameters compared to traditional architectures, making them suitable for resource-constrained environments.
  - **Non-linear mapping**: RDBs enable the learning of complex non-linear mappings between input and output data.
  - **Residual learning**: leveraging residual connections, RDBs learn to predict residuals, leading to faster convergence and improved generalization.
  - **Versatility**: RDBs can be easily integrated into various architectures and adapted to different tasks, offering flexibility in model design.



Upscaling: Super-resolution

- **Method**:

  Depth to space transformation.

  - **Rearranges** the elements of an input tensor to change its spatial dimensions while preserving the information content.
  - **Height** and **width** are the **spatial** dimensions of the input.
  - **Depth** represents the number of channels or feature maps in the input.

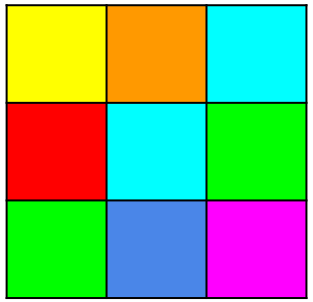
- **Purpose**:

  - **Depth-to-space transformation** is a useful tool for increasing the spatial resolution of feature maps while reducing the depth dimension, making it beneficial for tasks such as image super-resolution and dense prediction tasks in computer vision
  - Increase image resolution by rearranging depth dimension into spatial data.

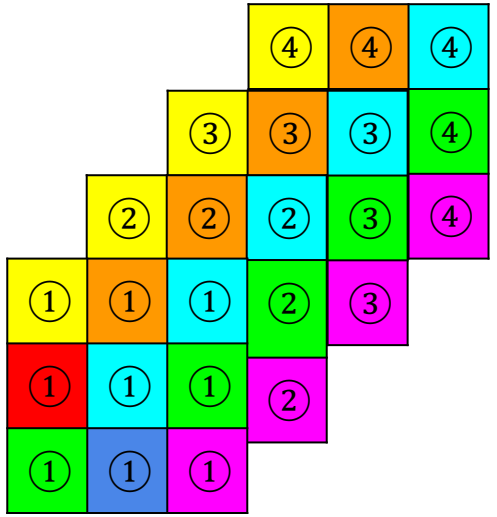
Depth to Space

Increase image resolution by rearranging depth dimension into spatial data.

Input

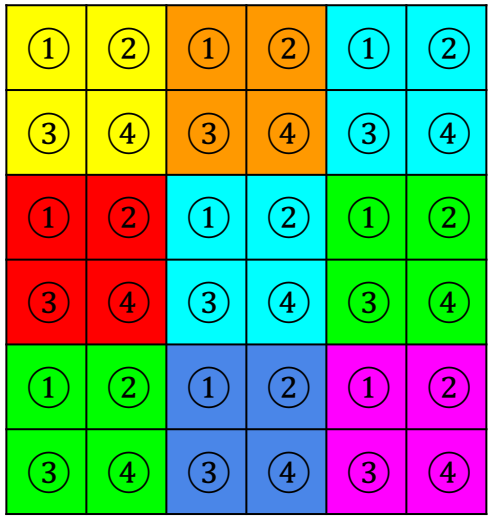


Conv

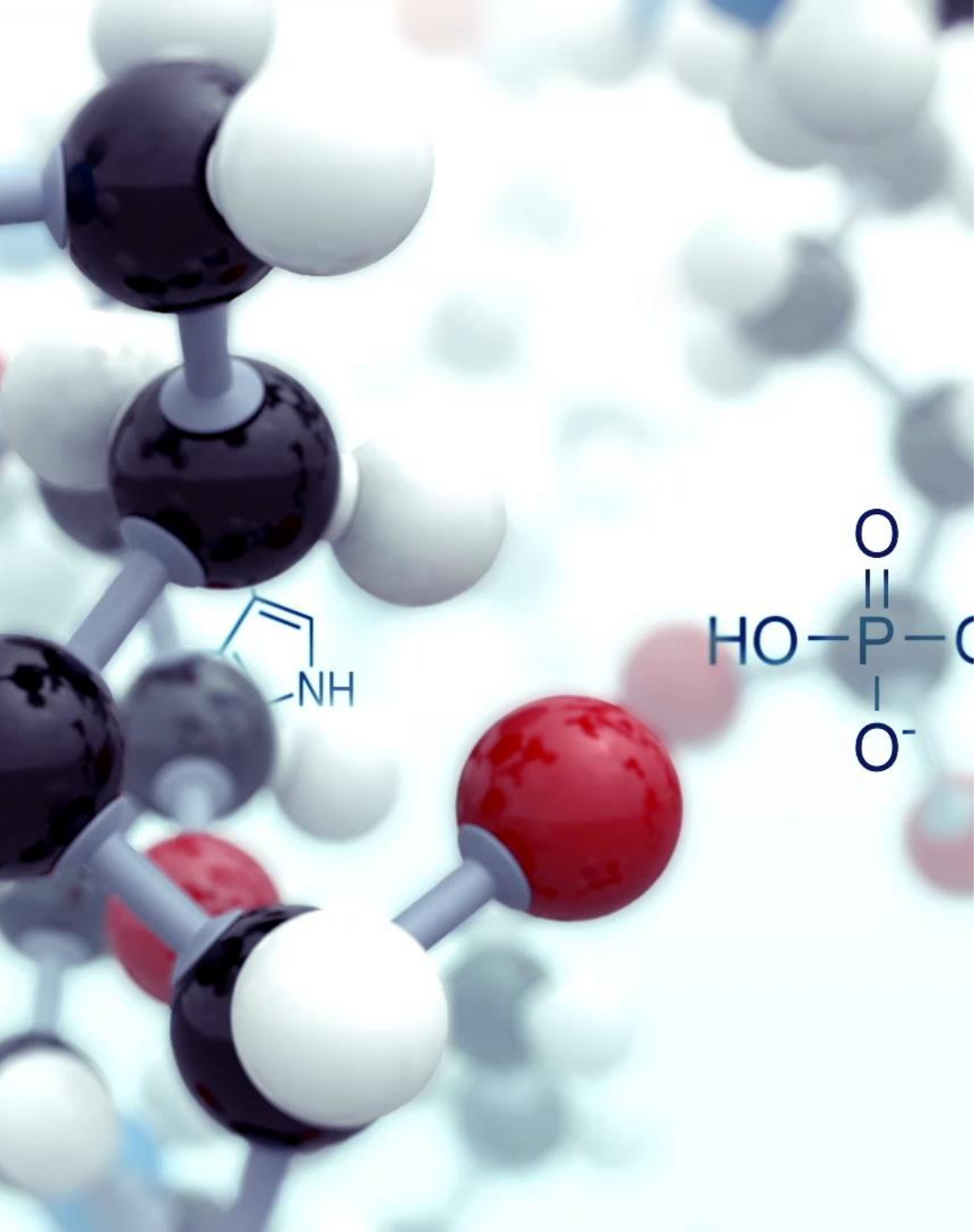


DtS

Output



Model Architecture Review



Model Explanation

- **Structure**: CNN with RDBs.
- **Advantages**:
  - Captures spatial hierarchies in images.
  - Dense connections for better feature reuse.

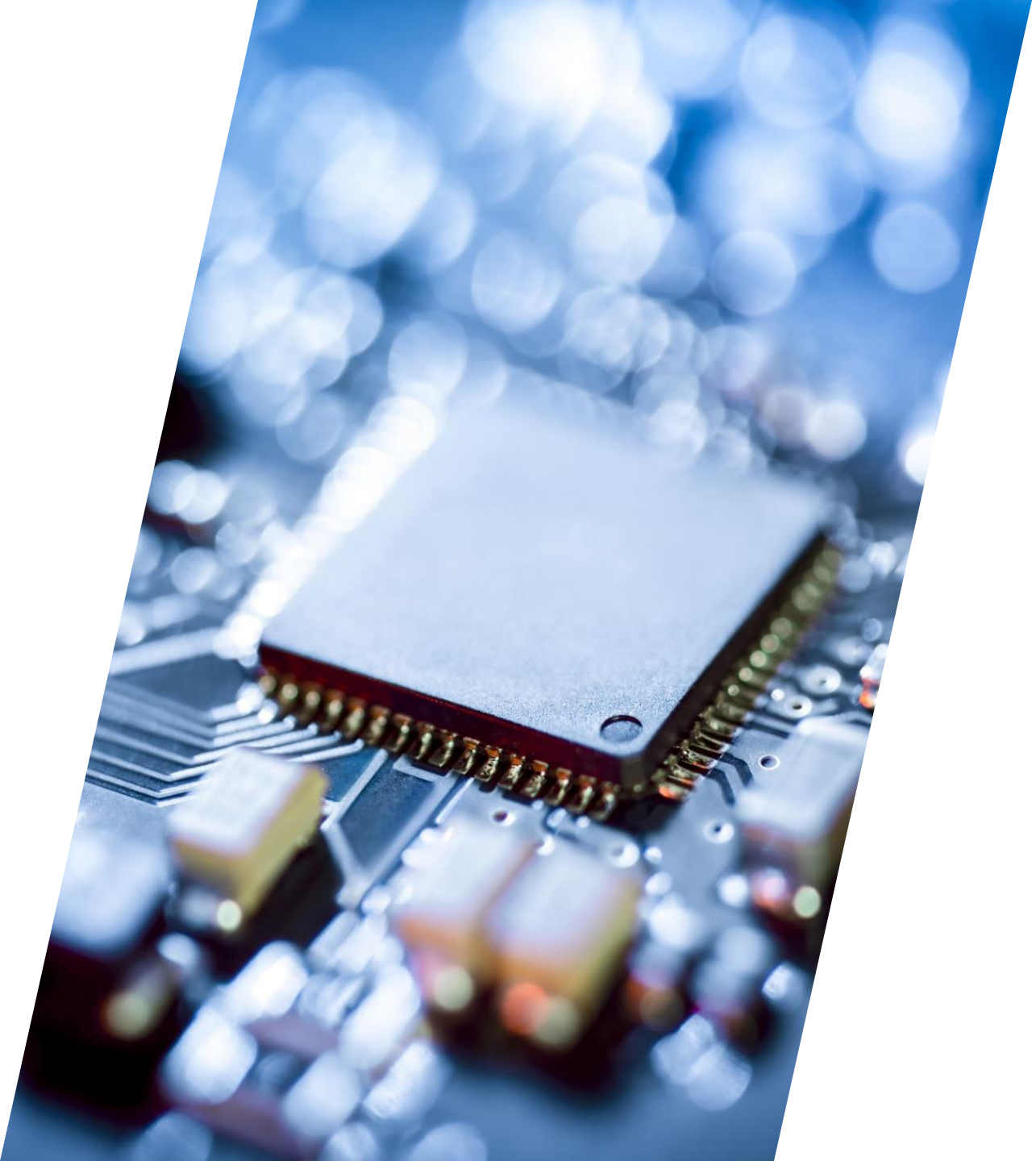
EARLY STOPPING AND MODEL FITTING

Early Stopping:

- Monitors training loss.
- Stops training if no improvement.
- Patience of 10, min_delta of 0.0001.

Model Fitting:

- Uses **model.fit** to train on the dataset.
- Adjusts weights and biases based on gradients.



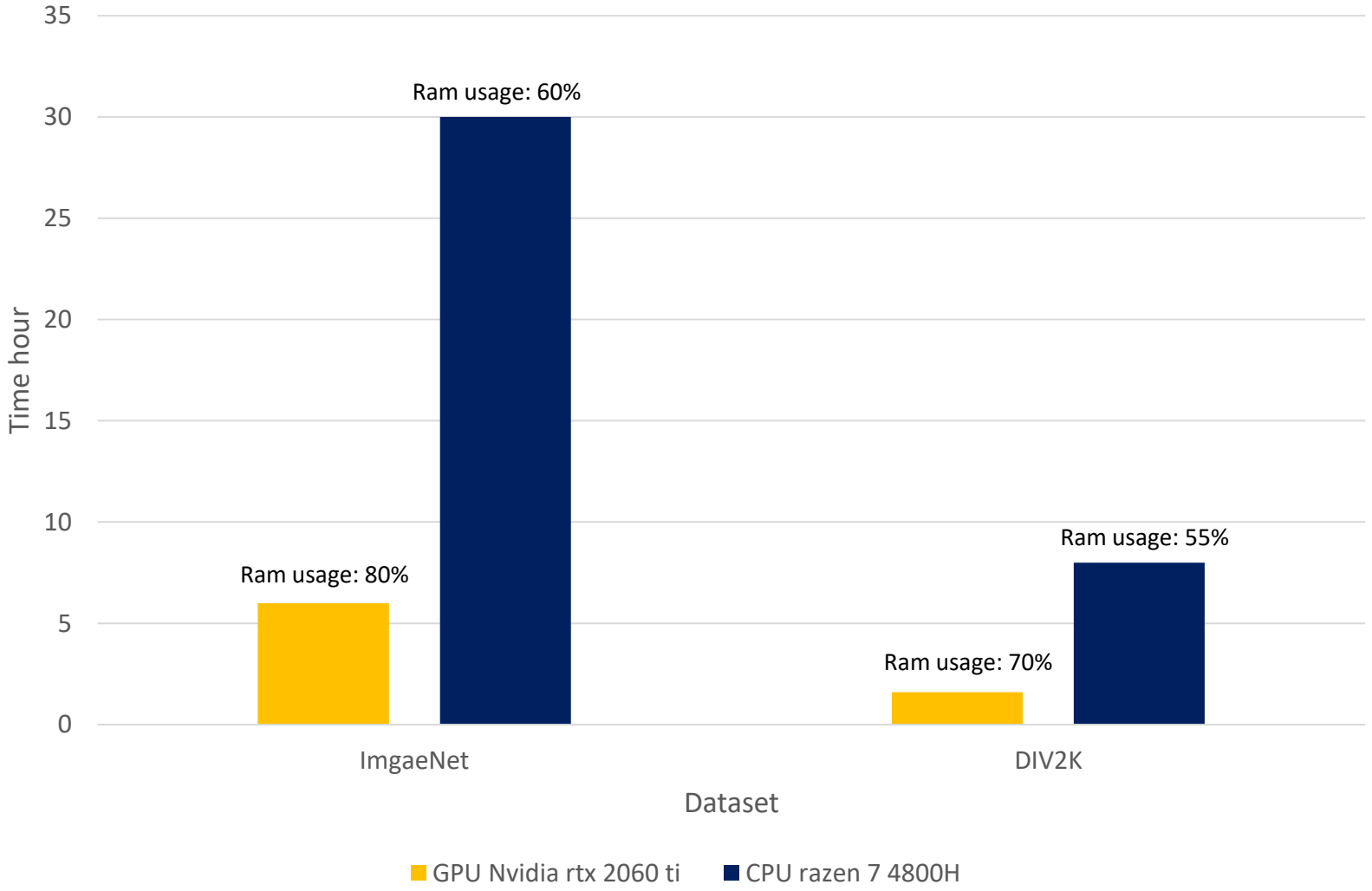
Model Saving And Hardware Used

- **Saving the Model**:
  - Format: keras (file).
  - Stores model for future use.
- **Hardware**:
  - **GPU**: Nvidia RTX 2060 TI
  - **CPU**: AMD Ryzen 7 4800H
  - **RAM**: 48GB, DDR4, 3200MHz

Training Time And Resource Usage

- **Comparison**:
  - **GPU (Nvidia RTX 2060 TI)**:
    - RAM Usage: 100%
    - Training Time: Faster
  - **CPU (AMD Ryzen 7 4800H)**:
    - RAM Usage: 55-90%
    - Training Time: Slower

Resources and Training Time Comparison



Postprocess

Preprocessing Model's Input

YUV Conversion: Convert to YUV color space and extract the Y (Luminance) channel.

Array Conversion: Convert the Y channel of the YUV image to an array for further processing and model input.

Normalization: Convert pixel values from [0, 255] to [0, 1].

Reshaping:Reshape the normalized Y channel to fit the model's expected input dimensions.Input shape: (1, height, width, 1) - 1 image at a time, with height, width, and 1 channel (luminance).

Postprocessing Model's Output

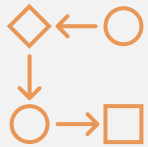
Reshaping: Reshape the output to remove additional dimensions added during preprocessing.

Denormalization: Multiply pixel values by 255.0 to convert back to the original range [0, 255].

Clipping: Clip pixel values to ensure they fall within the valid range for display.

Array Conversion: Convert the array back into an image format.

Postprocessing Model's Output



Resizing Chrominance Channels (U, V): Resize Cb and Cr channels to match the dimensions of the enhanced luminance channel using Bicubic Interpolation.

Merging Channels: The resized Cb and Cr channels are merged with the

enhanced Y channel to recreate the full YCbCr image

Convert YUV to RGB: The YCbCr image is converted back to the RGB color space. This final conversion restores the image to its original color format

Challenges Faced

1. lack of Computational Resources:

1. Struggled to find sufficient computational resources for effective training.

2. University Support:

1. University's inadequate support and neglect hindered progress.

3. Data Volume:

1. Forced to use a small dataset due to resource limitations.

4. Training Time:

1. Excessively long training time due to limited resources.

Thanks For Your Attention

- **Authors:**

  Amer Zuher Alriyahy

  Hisham Maher Sunjaq

- **Supervisor:**

  Dr. Mohmmad Al-Musaideen