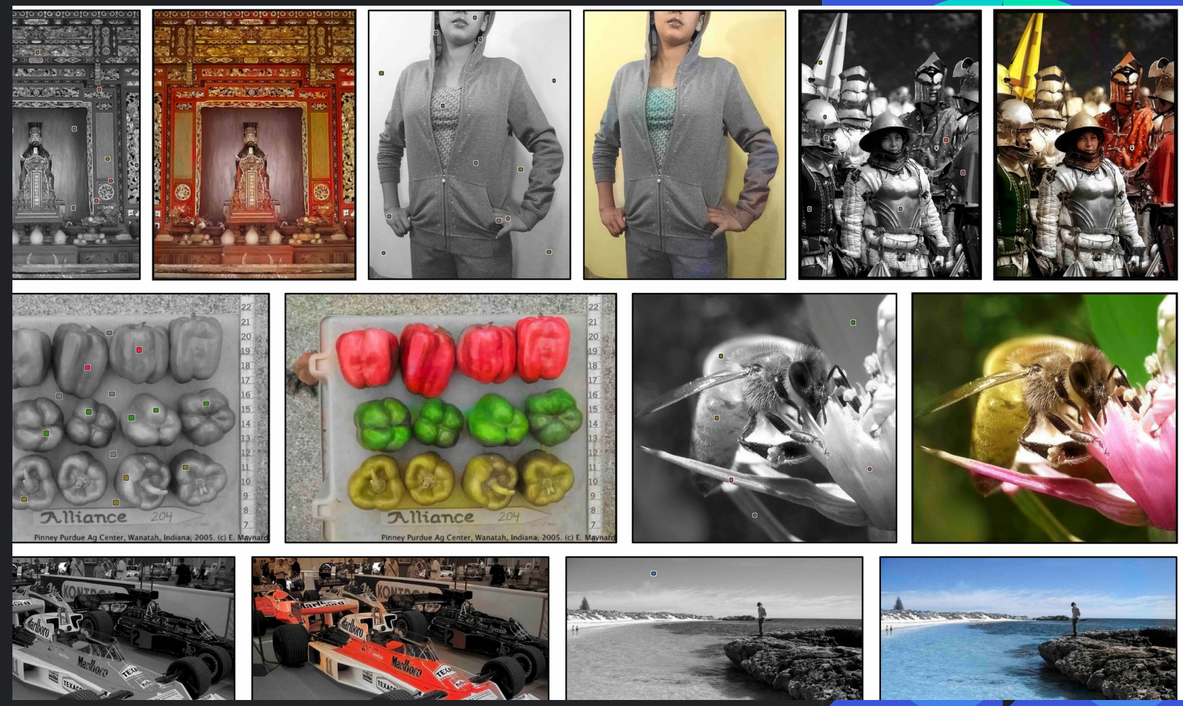
IMAGE COLORIZATION PROJECT

Created by
1953223,
1966150,
2005008

IMAGE COLORIZATION WITH DEEP LEARNING

- Our project is centered around image colorization using deep learning methods.

- **Goal**: The primary objective of this project is to create an effective model for automatically adding color to grayscale images, harnessing the capabilities of PyTorch, a renowned deep learning framework.



Project Overview

Step By Step Outline

- Data loading
- Model architecture
- Training process
- Testing and evaluation
- Results

Dataset

kaggle

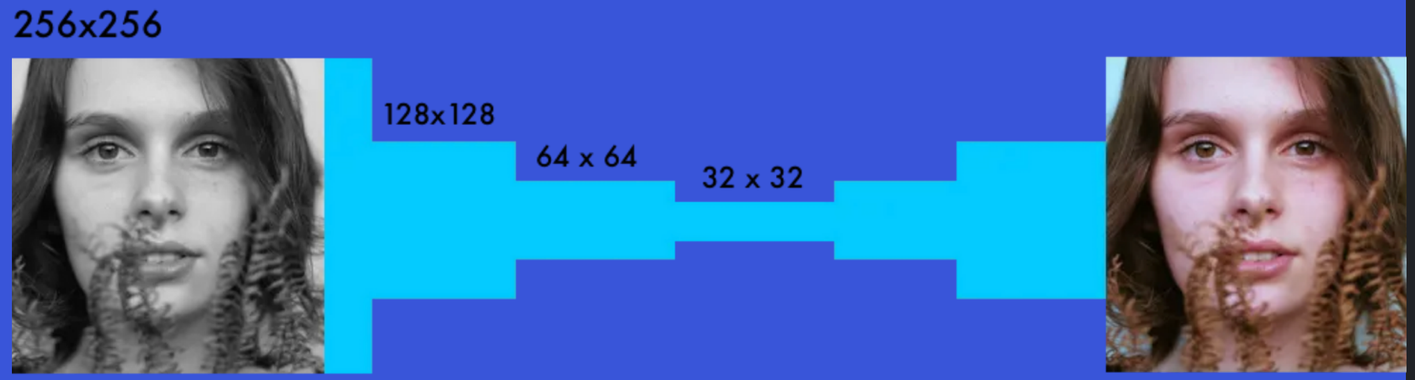
Custom Data Loader

```python
import torch
import os
from torchvision.io import read_image, ImageReadMode
from torch.utils.data import Dataset

class MyDataLoader(Dataset):
    def __init__(self, img_x_path, img_y_path, transform=None):
        self.img_x = img_x_path
        self.img_y = img_y_path
        self.transform = transform
        self.file_list = os.listdir(self.img_x)  # List of filenames in the directory

    def __len__(self):
        return len(self.file_list)

    def __getitem__(self, idx):
        single = self.file_list[idx]
        image_x = read_image(os.path.join(self.img_x, single), ImageReadMode.RGB)
        image_y = read_image(os.path.join(self.img_y, single), ImageReadMode.RGB)

        if self.transform:
            image_x = self.transform(image_x)
            image_y = self.transform(image_y)
        return image_x, image_y
```

DATA SPLITTING

- Data Division: Split data into training and testing sets which already done manually during data preparation pace.
- Training Data: Loaded using data_loading(path_train_gray, path_train_rgb, data_transform).
- Testing Data: Loaded using data_loading(path_test_gray, path_test_rgb, data_transform).
- Batching: Batch size of 50 for efficient training and we were able to manage it with colab GPU.

MODEL ARCHITECTURE



Model: Utilized a custom model named MyModel for image colorization.

Architecture: Employed a Convolutional Neural Network (CNN) in an encoder-decoder architecture.

Encoder: The encoder component extracts essential features from grayscale images.

- Gradually halved picture dimensions using a stride of 2.

:

Decoder: The decoder component generates colorized images.

- Implemented upscaling by a factor of 2 using transpose convolutional layers.

Activations

both activations are used in the model

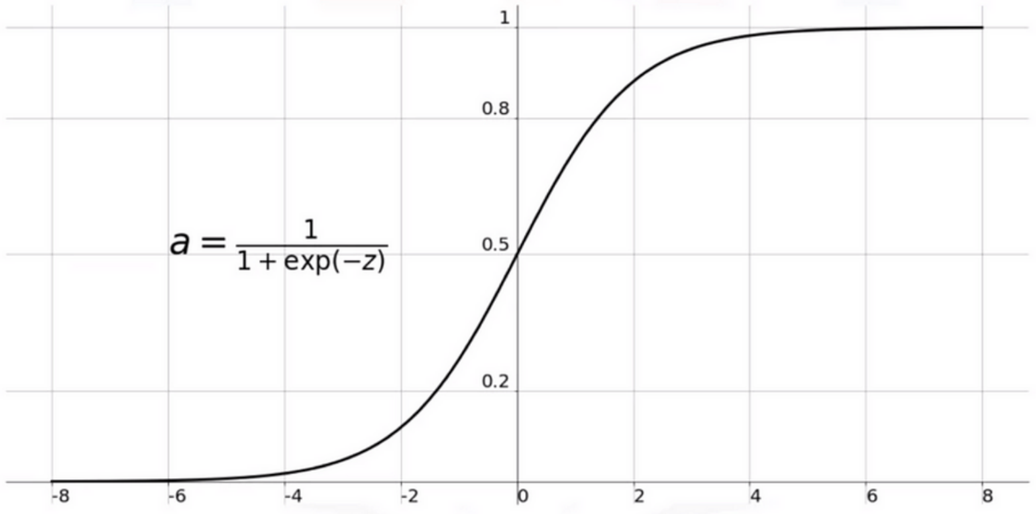
The ReLU Activation Function...

$$f(x) = \max(0,x)$$

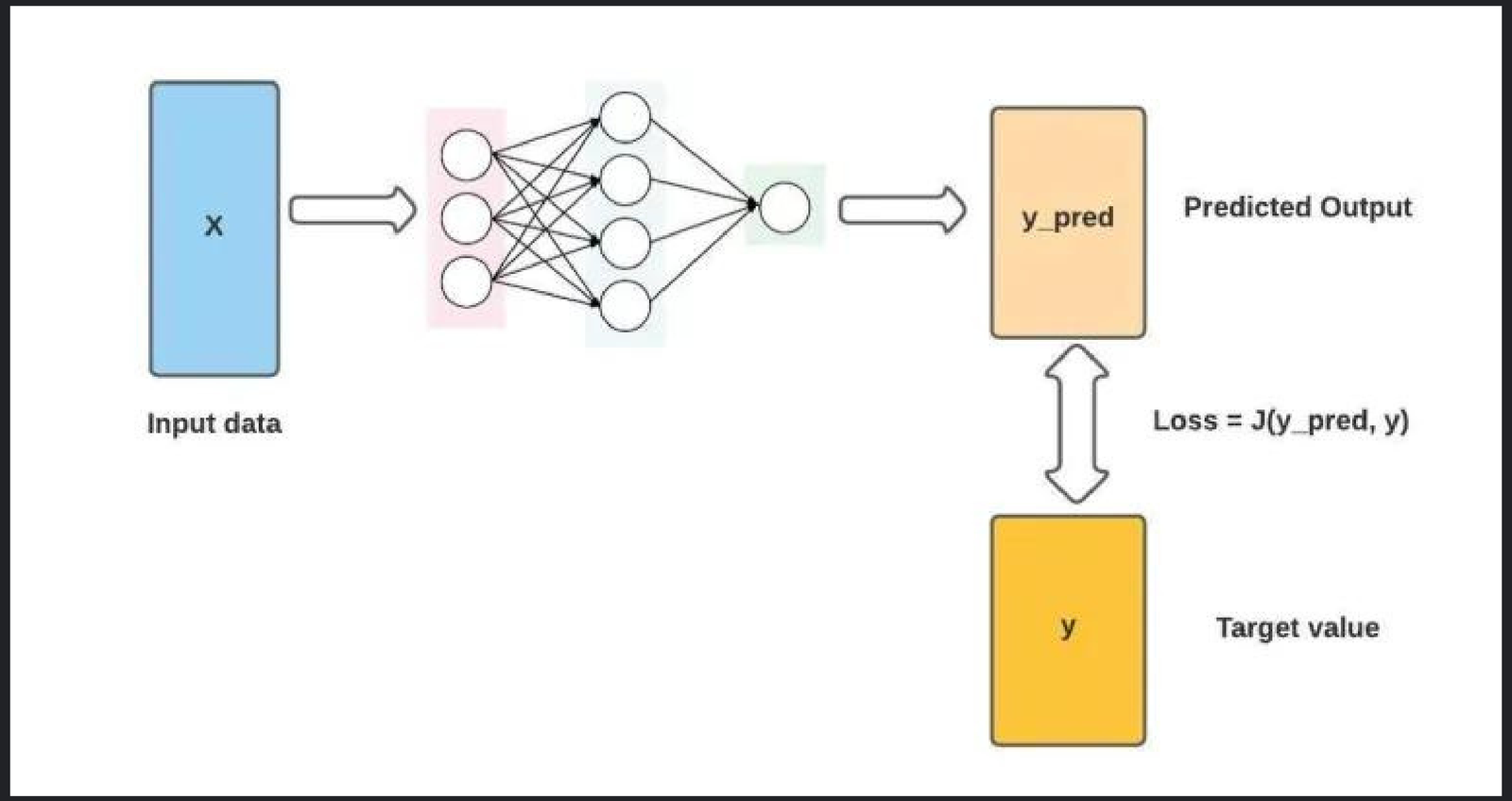
...Clearly Explained!!!

Sigmoid Function

$$a = \frac{1}{1 + \exp(-z)}$$



Loss function



$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

TRAINING PROCESS

TRAINING

Trained the model using a training loop

LOOP

Iterated through 2000/50 batches for 300 epochs.

COMPONENTS

Forward passes, loss computation (MSE), and AdamW optimization

MONITORING

Batch-wise loss printed during training

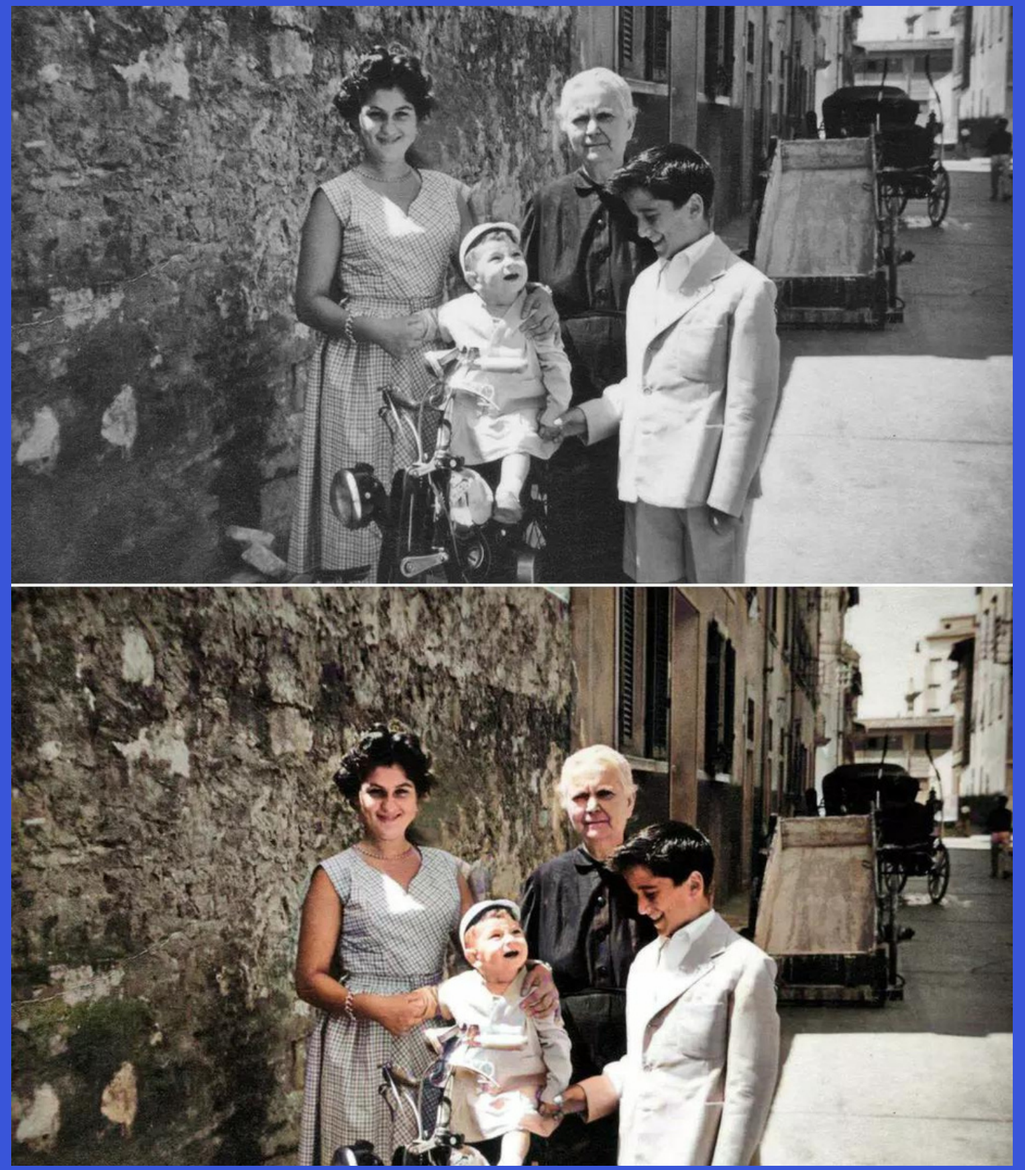
TESTING AND EVALUATION

Testing:
Model evaluated using a testing loop.

Output:
Saved colorized outputs and ground truth images. (we just picked1 random picture to test)

Visual Evaluation: We evaluate how gradually increases the quality of the same test image.

COLORIZED IMAGE RESULTS



With just 2000 training images we managed to colorize the BW image successfully.

Hyperparams:

L-rate= 0.005

Batch=50 (used Colab GPU)

Final_loss = 0.00021

- BW and Original image are present. The images initially 128x128 images 3 channels each.

THANK YOU!