# **DEEP LEARNING VIDEO/IMAGE COLORIZATION Black-and-White to Color using CNN Regression**

## **📌 OVERVIEW**

This project is a deep learning pipeline that colorizes grayscale video frames using a regression-based CNN model. It extracts frames from the DAVIS dataset, trains a convolutional encoder-decoder model, and reconstructs colorized versions of the input frames to generate video output.

The entire model is trained on the grayscale-to-color transformation using pixel-wise regression (not classification), resulting in fairly high colorization quality (~90% accuracy). We evaluated the output quality using PSNR (Peak Signal-to-Noise Ratio), although alternatives like SSIM were considered.

## **⚙ FEATURES**

* Custom-built CNN for image-to-image translation
* Input: Grayscale images
* Output: RGB colorized images
* Training on 100+ samples per video
* Uses Nadam optimizer and MSE loss
* Output video reconstruction supported
* PSNR-based evaluation
* Optional real-time frontend possible (via Gradio)

## **📁 FILE STRUCTURE**

colorization-project/ │ ├— notebooks/ │ └— colorization.ipynb ← Main training & evaluation code │ ├— model/ │ └— colorization\_model.py ← (Optional) Model structure code │ ├— dataset/ │ └— sample/ ← Sample input-output images │ └— requirements.txt ← Required libraries └— README.txt ← This file

## **👤 CONTRIBUTORS**

Team Members:

* Divyang Mehta — Core Implementation, Model Coding
* Piyush Deshmukh — Architecture, Optimization
* Shashwat Sethi — Dataset Preparation, Validation
* Aditya Kumar — Video Processing, Documentation & Output Generation
* Piyush Kumar Tiwari — Project Structuring, Theory, Architecture & Deployment

## **🧪 METHODOLOGY**

1. Frame Extraction:
   * From the DAVIS dataset using OpenCV
   * Saved as 128x128 RGB and grayscale PNGs
2. Model Architecture:
   * Encoder-decoder CNN built in TensorFlow
   * 3 encoding layers, bottleneck, and 3 decoding layers
   * Final activation: tanh
   * Loss: Mean Squared Error
3. Evaluation:
   * PSNR used for performance measurement instead of SSIM
   * PSNR is easier to compute and interpret during training
   * SSIM, while perceptually better, is more computationally complex
4. Output:
   * Colorized video frames stitched back into video
   * Output saved as .mp4 using OpenCV

## **🔧 HOW TO RUN (IN COLAB)**

1. Open the following notebook in Google Colab: https://colab.research.google.com/drive/1vsVcrwXWzWIKW22luAXJooUrdNYxsP62
2. Make sure to authenticate with Kaggle to access the DAVIS dataset.
3. Run all cells sequentially.

## **📋 REQUIREMENTS**

* TensorFlow >= 2.8.0
* NumPy
* OpenCV
* tqdm

Install using: pip install -r requirements.txt

## **💬 FUTURE IMPROVEMENTS**

* Add user-friendly web frontend (e.g. Gradio)
* Introduce skip connections in the model
* Try U-Net or GAN-based approaches
* Experiment with SSIM as loss or evaluation metric

## **📄 LICENSE**

For educational and non-commercial use only. Attribution required.