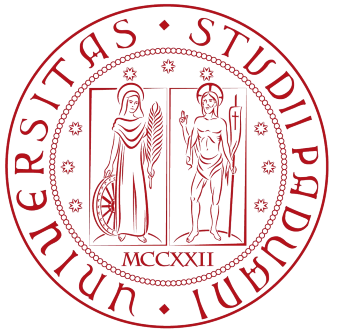


Image Colorization in Urban Landscapes

Authors: José Chacón, Joan Verguizas

15/02/2024

Image Colorization

Task

Process of adding color to a black and white or grayscale image.

Applications: Restore Historical Image, Colorize Old Films, Add color to infrared images, …

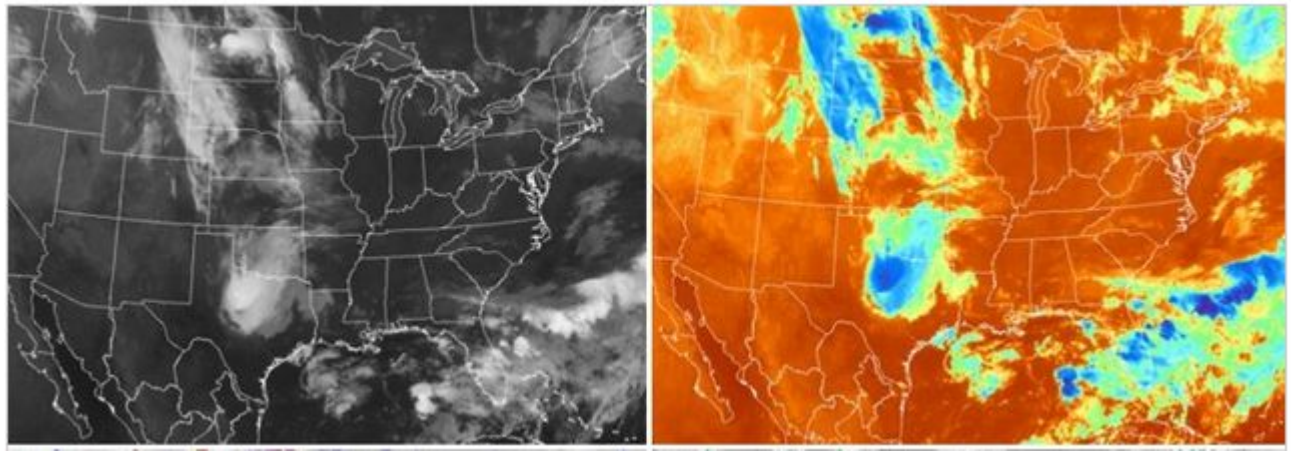


Image Colorization

Goal

Use DNN to perform image colorization on Urban Landscapes. Application: Restoration of historical images on urban sceneries.

The architectures used are different implementations of a encoder-decoder models.

Dataset

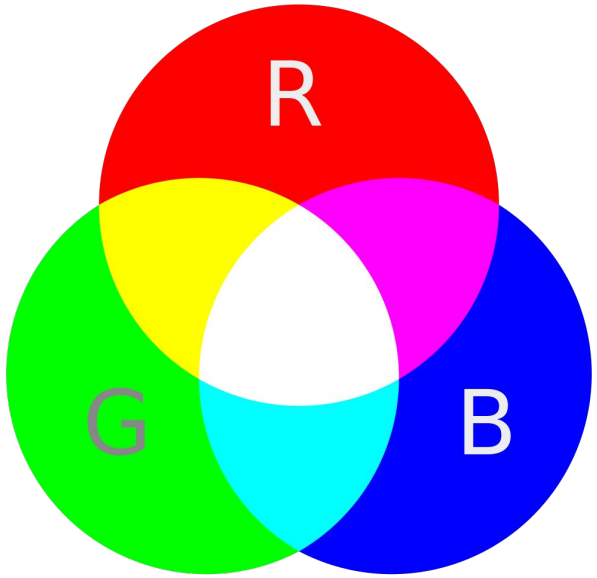
Scene UNderstanding (SUN) dataset.
Created for scene understanding tasks.

We use images of SUN397 that are in urban landscapes: 32 categories that add up to 12950 images.

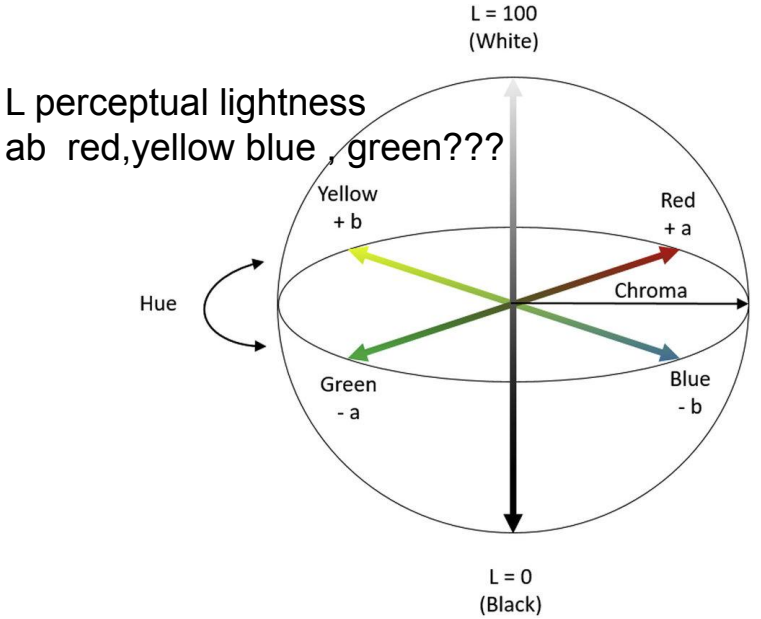
SUN397 is a subset that contains the 397 best well-sampled categories (at least 100 images per class).

Preprocessing

- Resize images to 256 x 256 pixels.
- Process images to have 3 RGB channels (some images are encoded as RGBA at the beginning)



We need to convert from RGB to LAB. Models take L values as input and return AB channels as output. These values are normalized to fall within the [-1, +1] range.



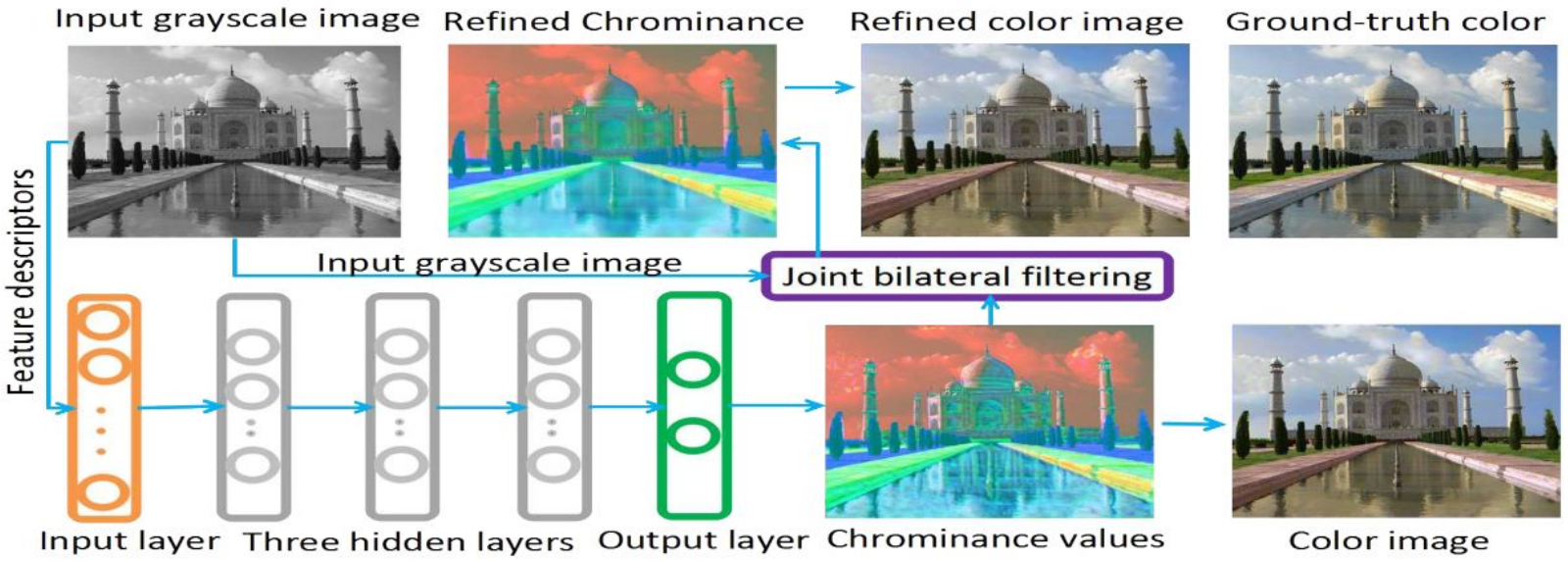
Related Work

Deep Colorization

Input to the model are low, mid and high level features.
- Low features: Pixel Patches.
- Mid features: DAISY descriptors.
- High features: CNN extracted semantic segmentation features.

CNN is used to perform a sub-task, the model does not leverage the power of representation learning.

At the end, refined chrominance and joint bilateral filtering are performed to improve the quality of the resulting image.



Z. Cheng et al. “Deep Colorization” https://arxiv.org/abs/1605.00075

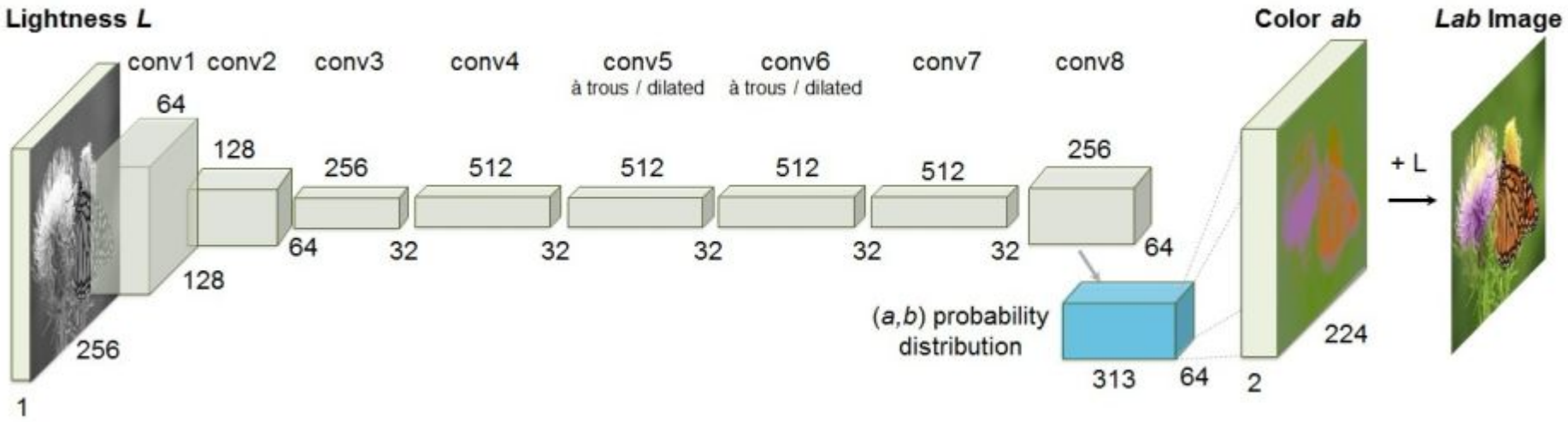
Related Work

Colourful Colorization

The input is the L channel 256 x 256 images. The output will be a 224 x 224 images produced by a probability distribution of the AB channels from which we can sample the image.

8 sets of blocks of layers. Each layer with 2 or 3 convolutional layer followed by a ReLU layer. Each block finishes with a batch normalization layer

Huge GPU RAM required to run the model, > 32M parameters.



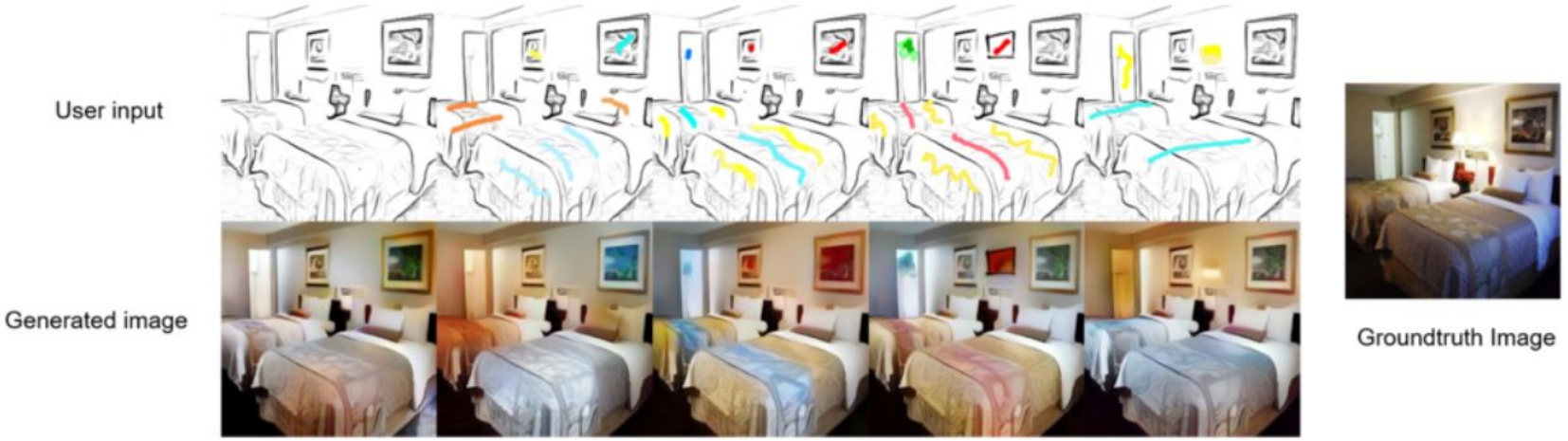
R. Zhang et al. “Colorful Image Colorization” https://arxiv.org/abs/1603.08511

Related Work

Scribblers

User guided deep generative adversarial network. The input to the generator network will be image sketches and color strokes.

Colorization dependent of the user sketching input. System suffers from several limitations. Blurry boundaries can be observed as well as color leaking from nearby objects



P. Sangkloy et al. " Scribbler: Controlling Deep Image Synthesis with Sketch and Color" https://arxiv.org/abs/1612.00835

Baseline Model

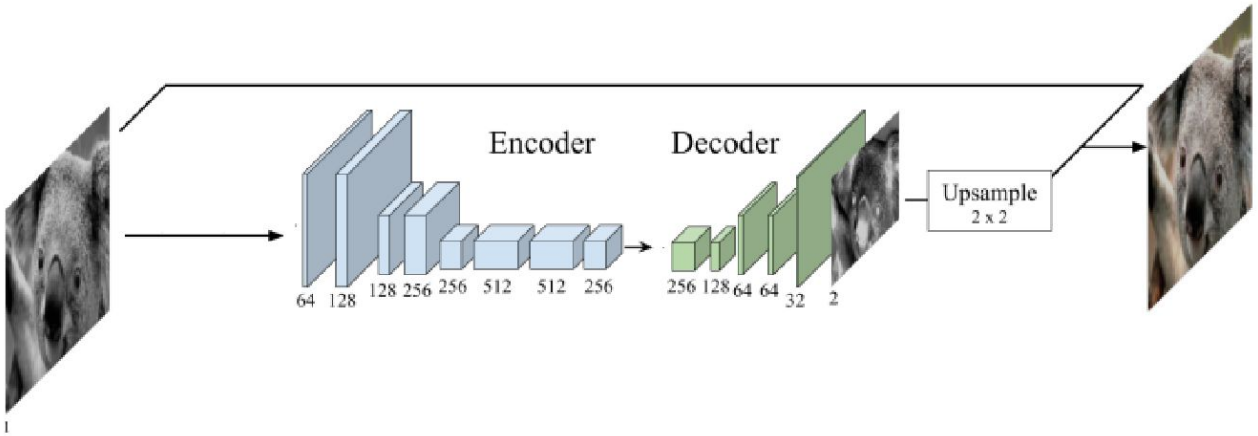
CNN based Encoder-Decoder

The input is the L channel 256 x 256 images. The output will be a 256 x 256 AB channels images. The input merged with the output is converted to RGB model color to obtain the colorized image.

Encoder: Has 8 convolutional layers. A stride of 2 is applied to 1st, 3rd, 5th layers to reduce the input dimension by a factor of 2.

Decoder: Has 5 convolutional layers. After the 1st, 3rd, 5th layers upsampling by a factor of 2 is applied.

The filter for every conv layer is a 3x3 kernel. The activation function is ReLU with the exception of the last one of the decoder that is tanh.



CNN-encoder-decoder-based model

The second architecture builds upon the previous one by adding two modules

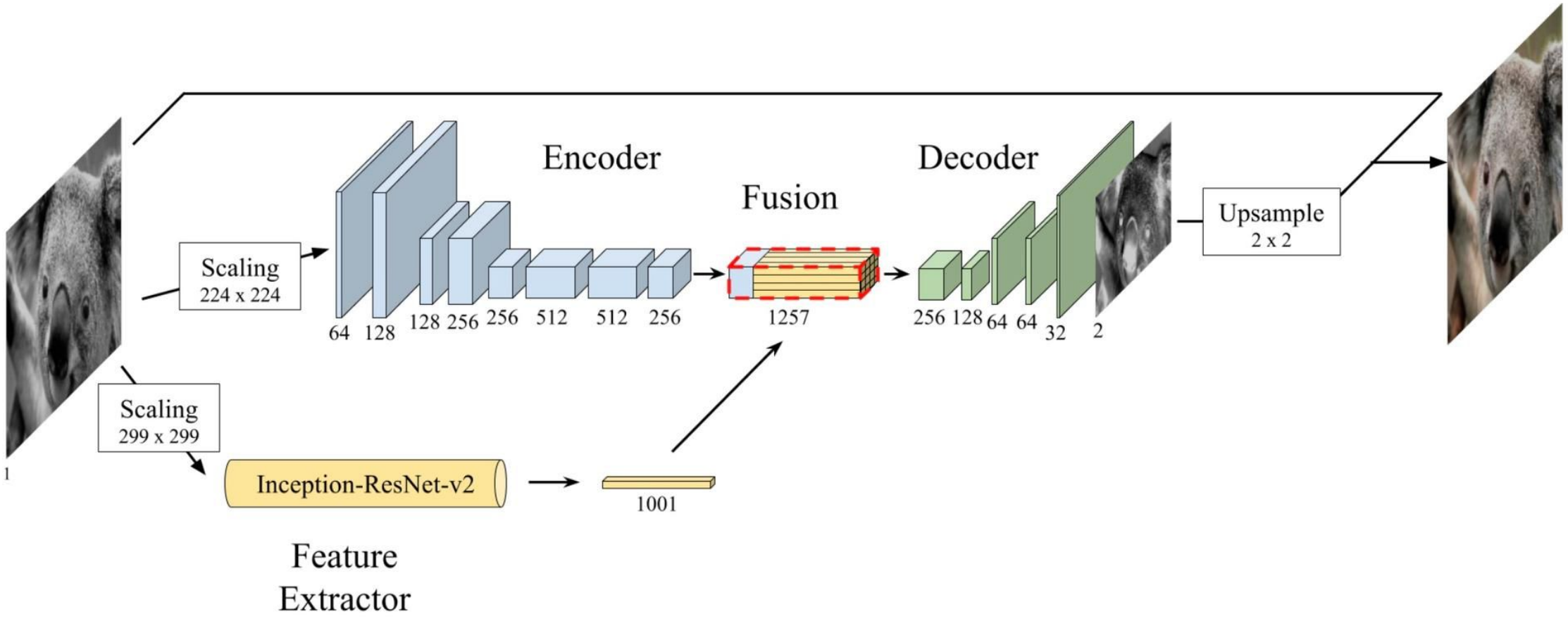
Feature Extractor

Two pre-trained models were used:
Inception V3 and **ViT**

Fusion Module

Responsible for integrating the outputs from the encoder and the feature vector extracted from the pre-trained model

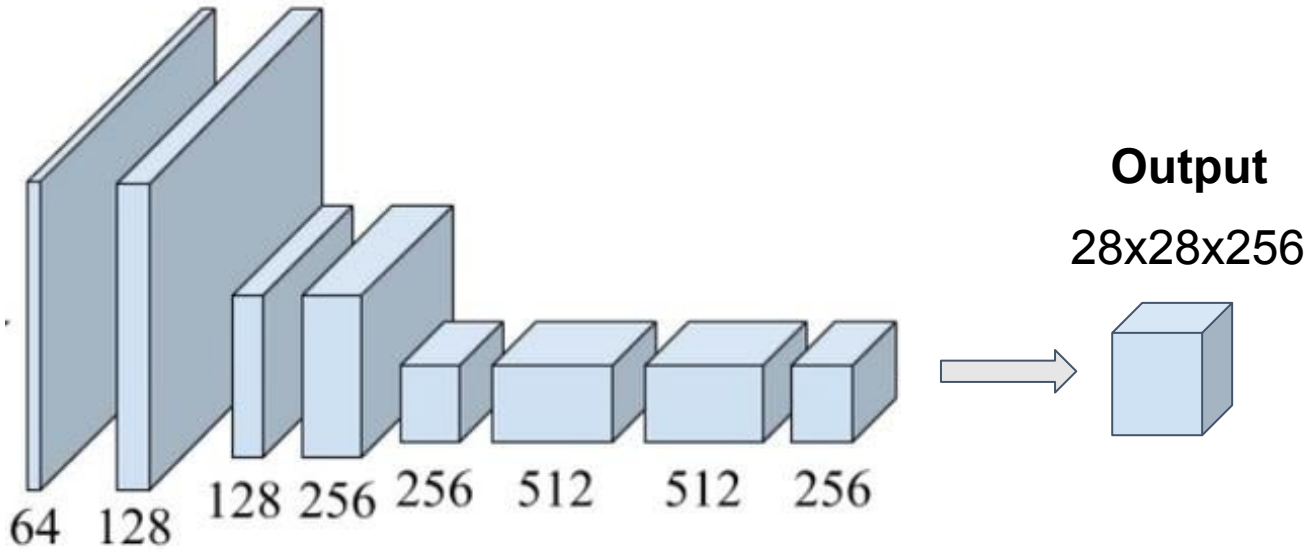
Architecture overview



Architecture from: F. Baldassarre et al. Grayscale Image Colorization using deep CNN and Inception-ResNet-v2 https://arxiv.org/abs/1712.03400

Encoder

- 8 conv layers with 3x3 kernels
- Padding employed to maintain input size of each layer
- Stride of 2 used in first,third and fifth layers to reduce volume dimensions



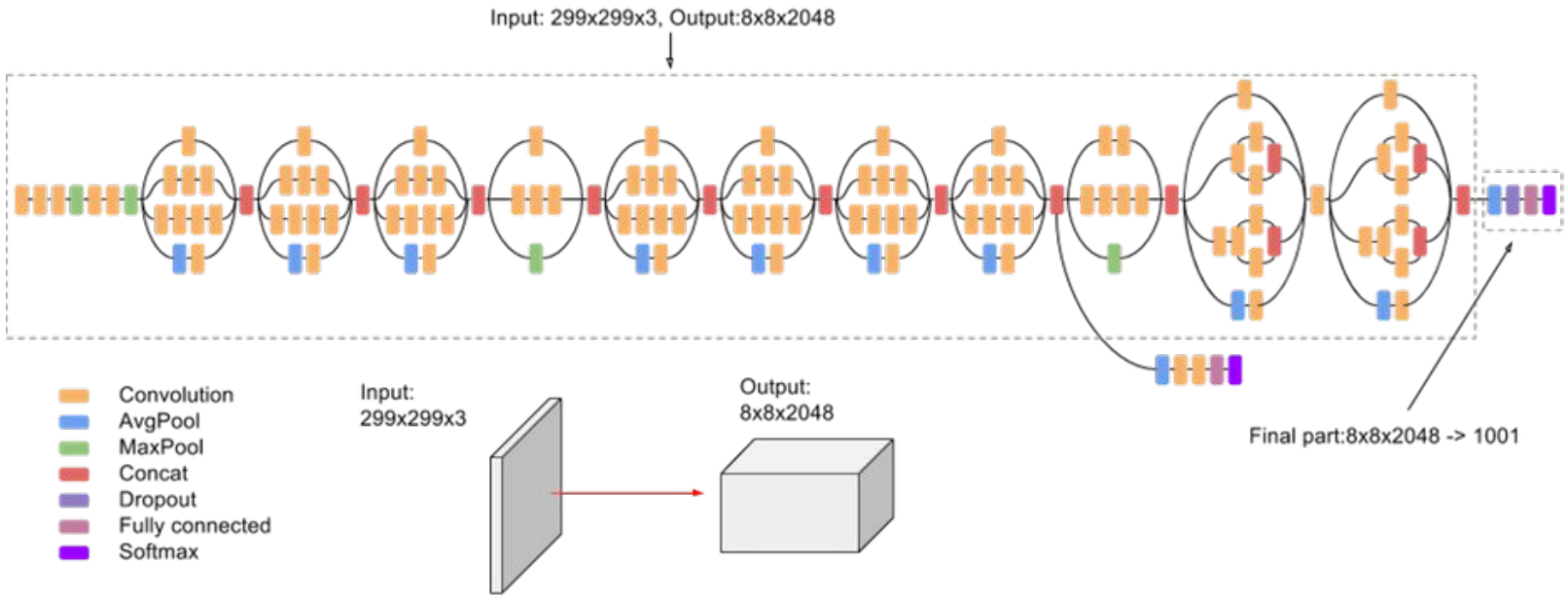
Inception-V3

Architecture consisting of 48 conv layers, trained on ImageNet.

Network within a network topology.

This version expanded the inception modules, improving training speed and reducing # of parameters.

Input images need to be 299x299x3



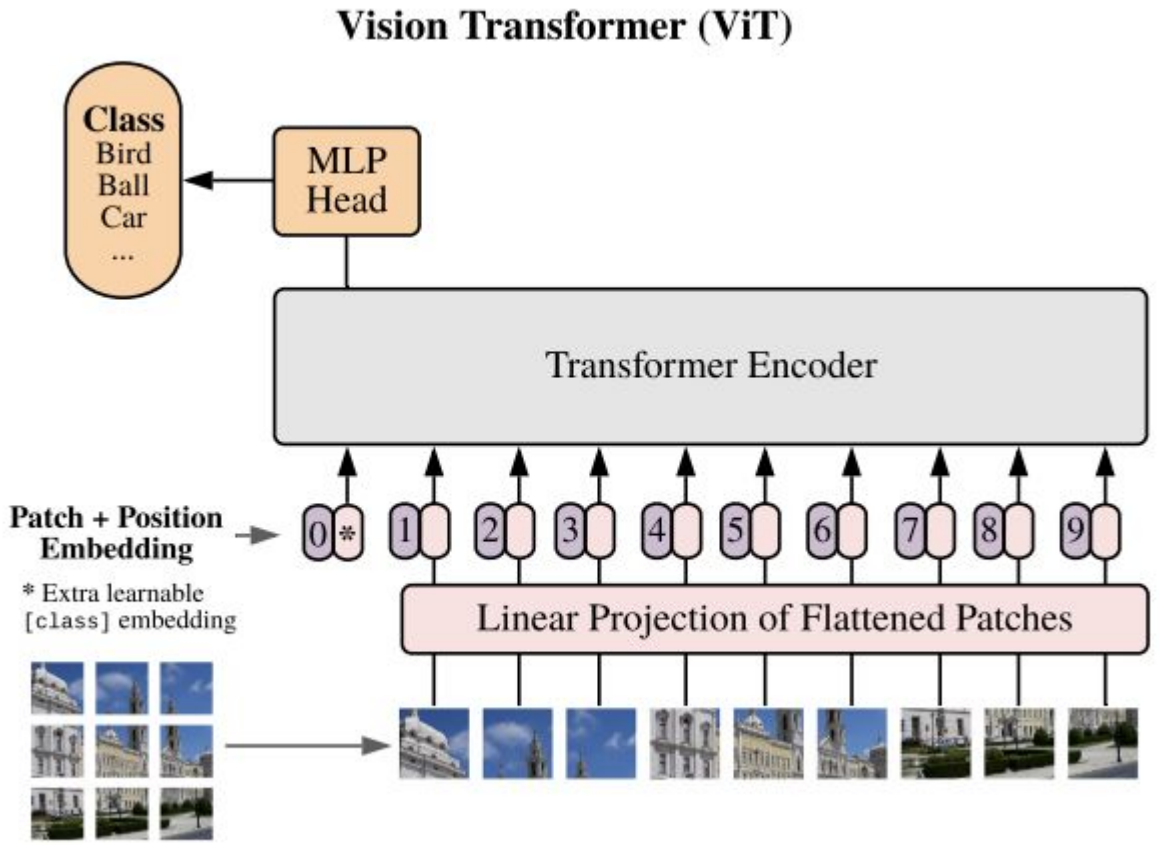
ViT

It divides an image into patches and processes them as sequences.

Configuration used : “google/vit-base-patch-16-224”

Contextual relationships across the entire image are captured due to self-attention mechanism

Input images need to be 224x224x3

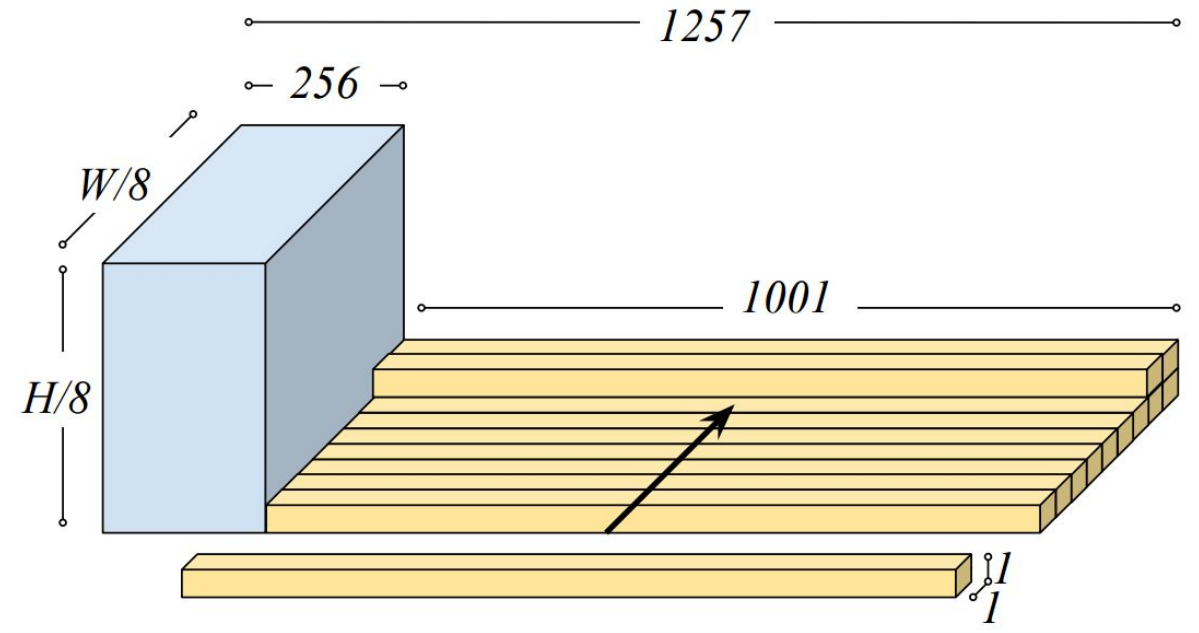


Fusion Module

Feature vector is concatenated along the depth axis.

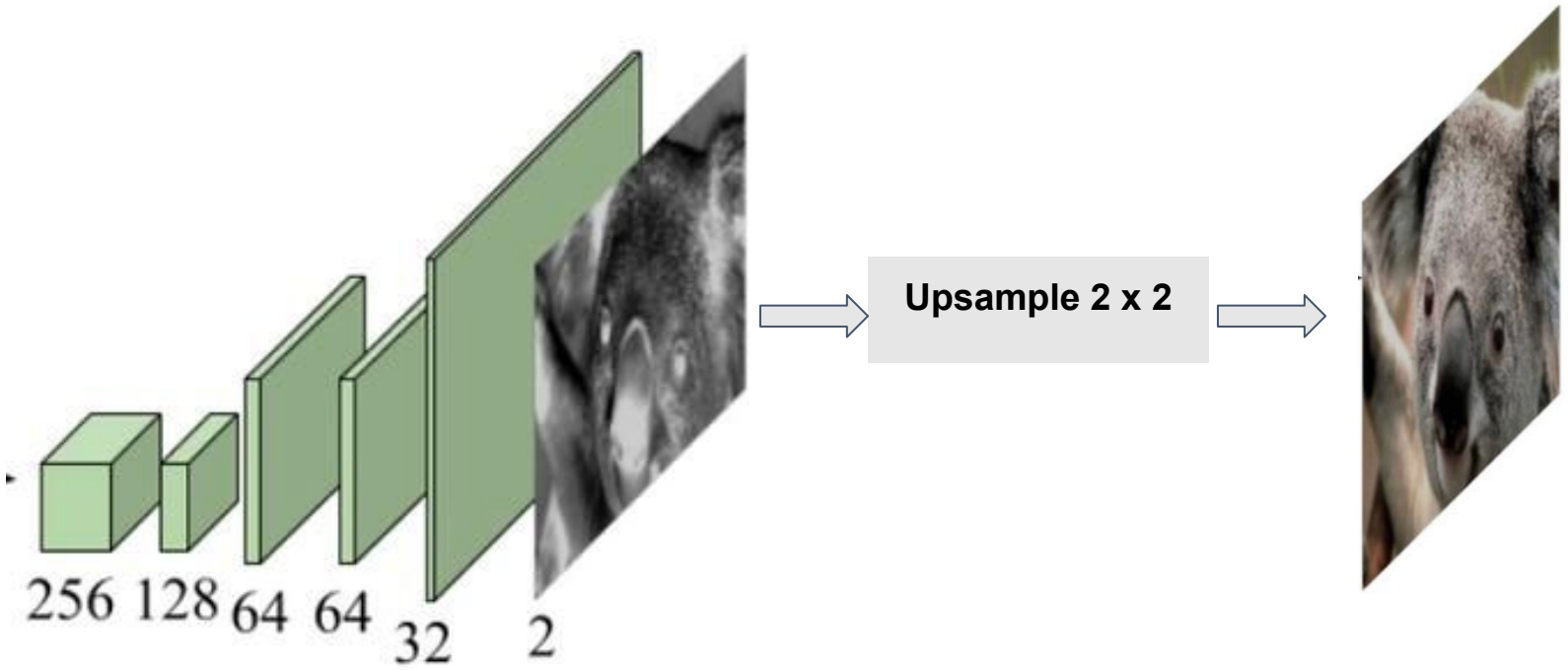
Semantic information caught by the pre-trained model is propagated through all regions of the image.

256 1x1 “bottleneck” layers are applied to reduce channel dimension



Decoder

- 5 conv layers of 3x3 kernels
- Upsample of scale 2 is performed after the first, third and fifth conv layer

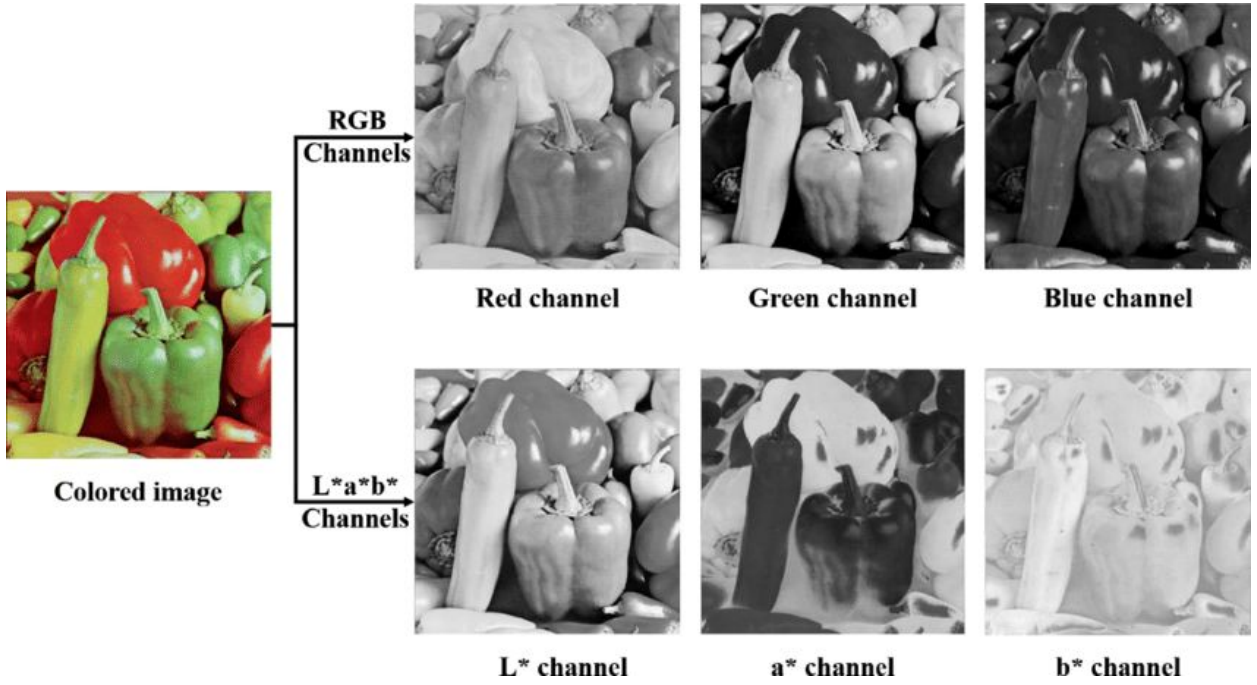


Preprocessing

Consisted of resizing each image so it could fit the resolution requirement for encoder and feature extractor modules.

After that, images are converted to LAB channel and had their L and AB channel extracted.

To reduce computational workload before training, images were preprocessed in advance and its representations were stored in the cloud



Training setup

All experiments were conducted on 1 NVIDIA T4 GPU.

Adam Optimizer was used with learning rate η = 0.0001 and weight decay 1e-6.

Hyperparameters

Hyperparameter	Value
Epochs	50
Learning Rate	1e-4
Batch size	64
Weight decay	1e-6

Results

Model	PSNR	SSIM	LPIPS	MSE
CNN Baseline	22.0465	0.8712	0.1725	0.0091
CNN + Inception	25.0957	0.9390	0.1158	0.0073
CNN + ViT	24.9997	0.9375	0.1106	0.0076

Metrics

Peak Signal-to-Noise Ratio (PSNR)

Measures the quality of a reconstructed or predicted image compared to the original. Focuses on pixel-level differences.

Structural Similarity Index Measure (SSIM)

Unlike PSNR, considers changes in structural information, luminance, and contrast.Intends to reflect how human vision perceives image changes.

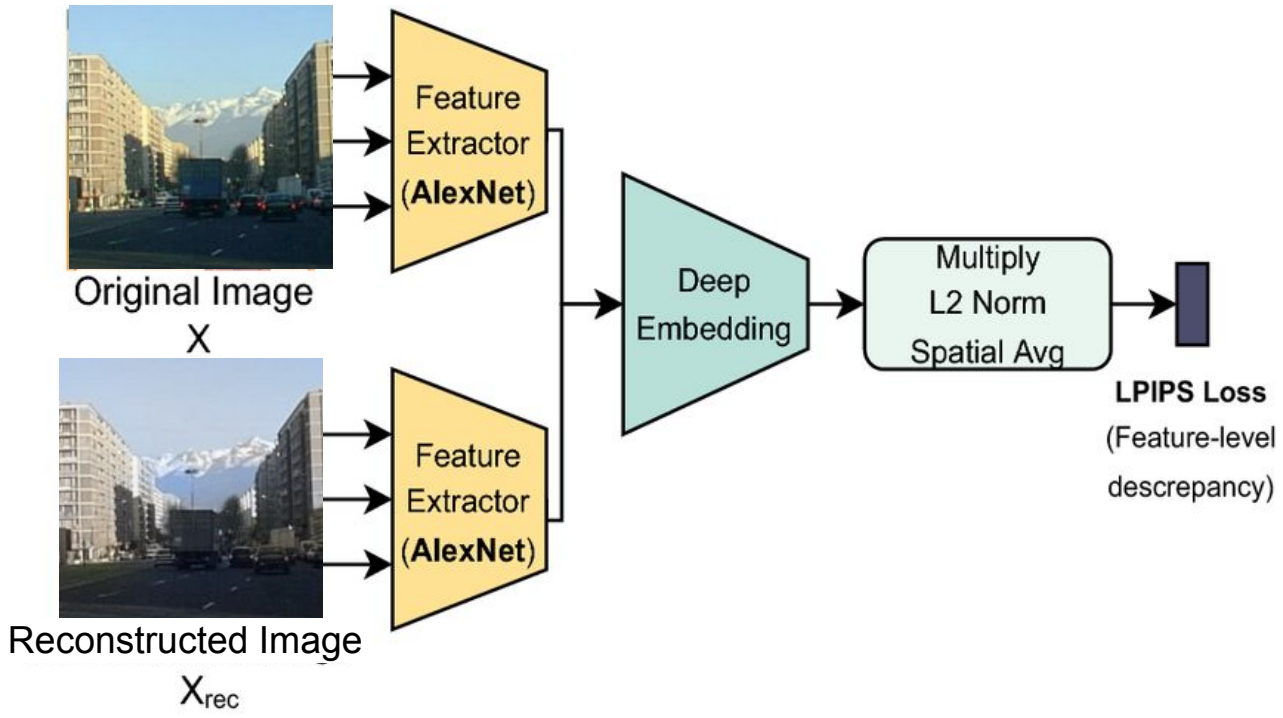
Mean Squared Error (MSE)

Calculates the average of the squares of the errors between corresponding elements in the original and reconstructed or predicted image.

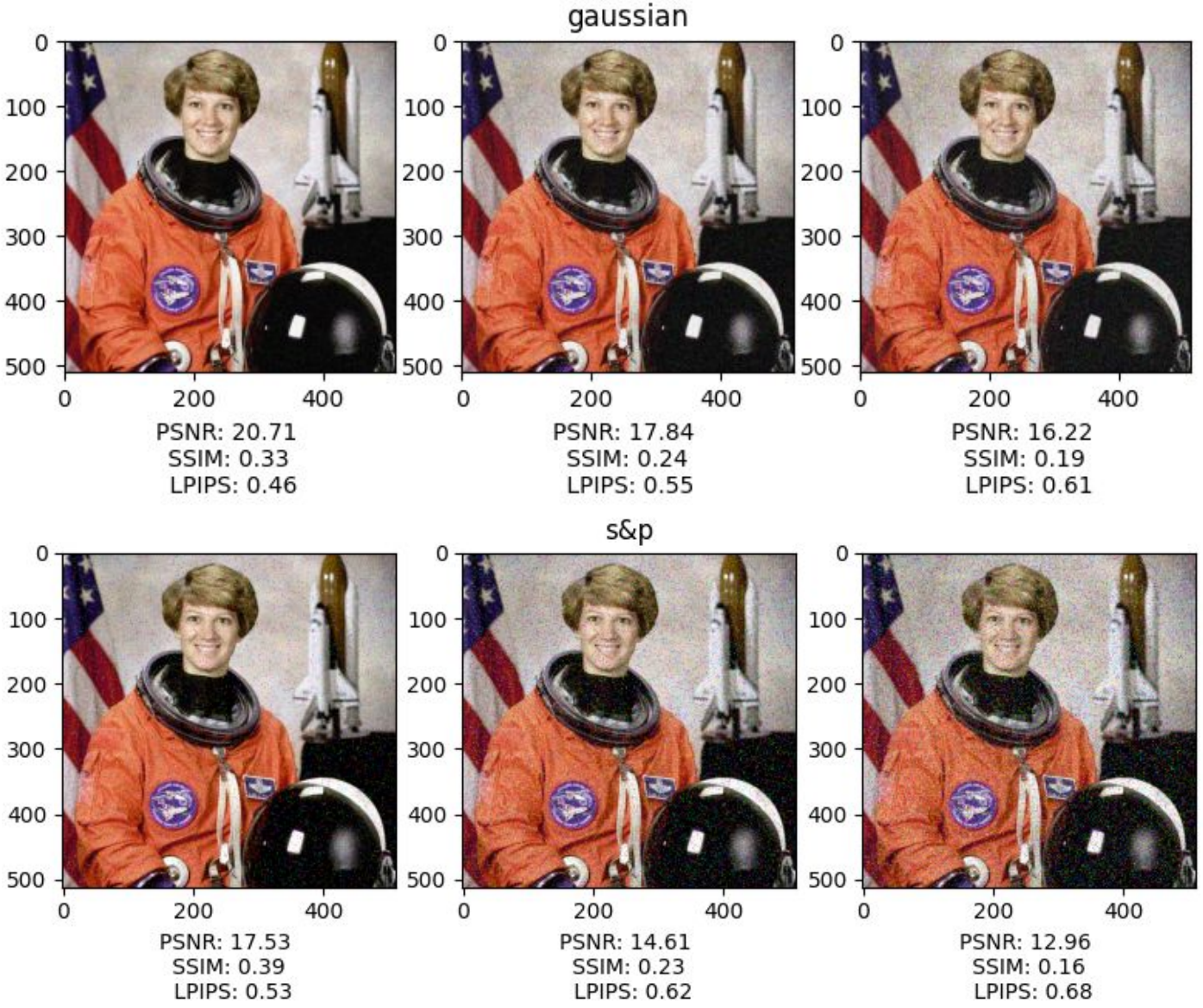
Metrics

Learned Perceptual Image Patch Similarity (LPIPS)

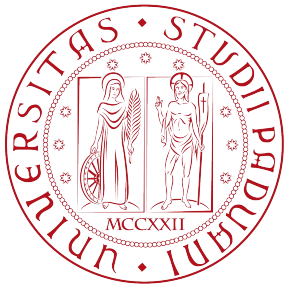
Measures the difference between features extracted from images by pre-trained neural networks, providing a score that reflects how similarly two images are perceived by human observers. **In our implementation, AlexNet was used**



Metrics


gaussian
PSNR: 20.71
SSIM: 0.33
LPIPS: 0.46
PSNR: 17.84
SSIM: 0.24
LPIPS: 0.55
PSNR: 16.22
SSIM: 0.19
LPIPS: 0.61
s&p
PSNR: 17.53
SSIM: 0.39
LPIPS: 0.53
PSNR: 14.61
SSIM: 0.23
LPIPS: 0.62
PSNR: 12.96
SSIM: 0.16
LPIPS: 0.68

Results



Performance metrics on test set

Model	PSNR	SSIM	LPIPS	MSE
CNN Baseline	22.0465	0.8712	0.1725	0.0091
CNN + Inception	25.0957	0.9390	0.1158	0.0073
CNN + ViT	24.9997	0.9375	0.1106	0.0076

Results

Classes with best colorization performance (according to MSE)

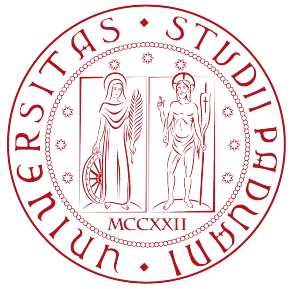
Class	PSNR	SSIM	LPIPS	MSE
Basilica	27.4201	0.9555	0.0708	0.0033
Skyscraper	25.6794	0.9468	0.0772	0.0050
Cathedral	26.0213	0.9444	0.0861	0.0053
Street	27.2297	0.9635	0.0889	0.0043

Qualitative Results

Some scene semantics were captured by the models like the fact that grass tends to be green and that the sky is typically blue

In others, model's predictions show a tendency for muted colors. This conservative values could be attributed to the loss function (MSE)

Failure cases



Classes with worst colorization performance (according to MSE)

Class	PSNR	SSIM	LPIPS	MSE
Phone booth	22.5894	0.8602	0.1418	0.0117
Market	22.1302	0.9117	0.1997	0.0133
Playground	22.1466	0.9111	0.1791	0.0134
Balcony	23.3941	0.9144	0.1128	0.0135

Failure cases

Predictions made by CNN + ViT model

Playground

PSNR: 21.837
SSIM: 0.918
LPIPS: 0.202

Market

PSNR: 19.939
SSIM: 0.903
LPIPS: 0.207

Phonebooth

PSNR: 18.382
SSIM: 0.868
LPIPS: 0.199

Applying our models on Padova historic urban landscapes

Porta Portello

Ground Truth

Inception CNN colorization

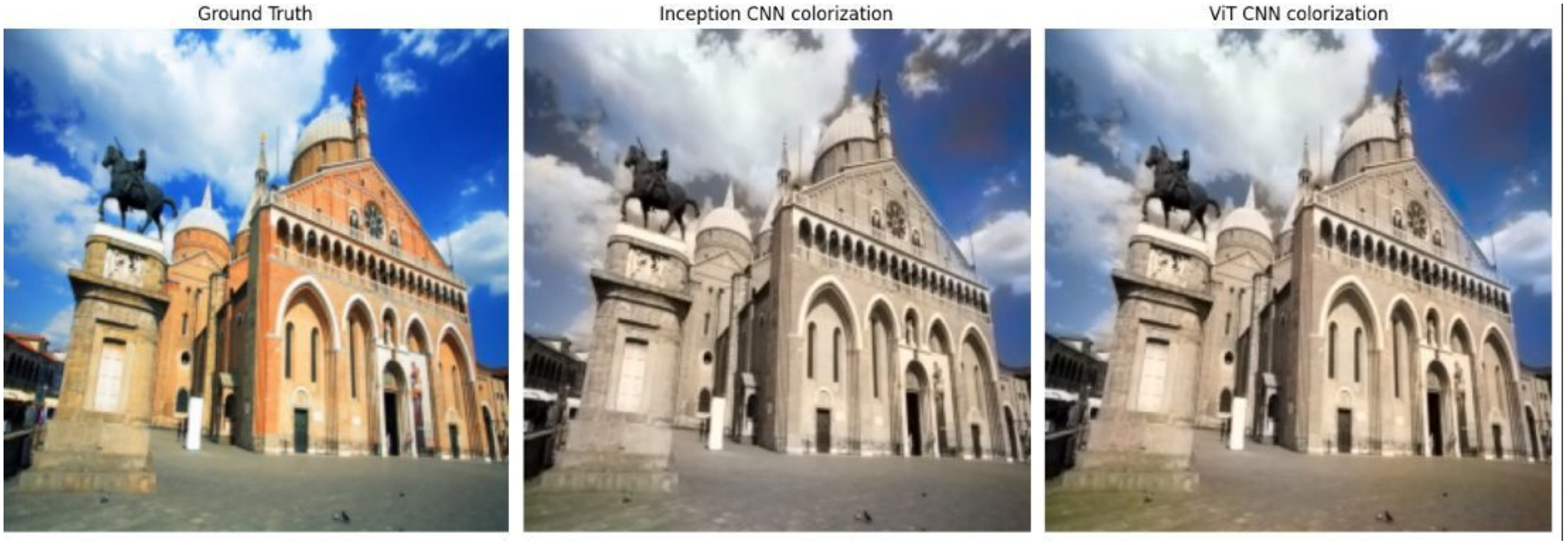
PSNR: 24.969
SSIM: 0.964
LPIPS: 0.073

ViT CNN colorization

PSNR: 24.761
SSIM: 0.963
LPIPS: 0.068

Applying our models on Padova historic urban landscapes

Basilica di Sant'Antonio di Padova



PSNR: 19.189
SSIM: 0.891
LPIPS: 0.153

PSNR: 18.940
SSIM: 0.885
LPIPS: 0.150

Applying our models on Padova historic urban landscapes

Pratto della Valle

Ground Truth

Inception CNN colorization

PSNR: 22.312
SSIM: 0.945
LPIPS: 0.173

ViT CNN colorization

PSNR: 21.777
SSIM: 0.937
LPIPS: 0.169

Conclusions

The overall performance of the image colorization task is satisfactory specially in high-level image components such as the sky, the sea or forests. The performance on coloring small details needs to be improved.

The colorization tends to be conservative with models tending towards low AB values. The MSE loss function may not penalize small AB values enough to encourage more diverse colorizations.

The inception model architecture is the one that performs better in MSE, PSNR, SSIM metrics. While performance of the visual transformer is better in the LPIPS metric while very close in the others . This can be explained both to architecture enhancement and pretraining.

Going Forward

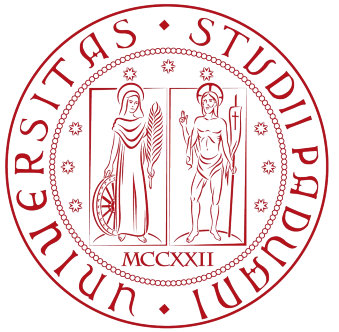
Training with a different loss function than MSE may help. Some of the loss functions that may be considered are pixel loss (L1 distance between the colorized image and the ground true image) or colourfulness loss (which is extracted making use of the mean and std of the colorized pixels). [1]

Other architecture features may be tested in order to try to improve the performance such as attention mechanism layers or skipping connections. [2]

Finally, image colorization task can be extended to be used for video colorization using video sequences instead of images. This, would require adapting the architecture to accommodate temporal coherence between subsequent frames.

[1] One recent paper that shows a modern approach to the task is: "DDColor: Towards Photo-Realistic Image Colorization via Dual Decoders" https://arxiv.org/abs/2212.11613

[2] H. Carrillo et. al. used a self-attention mechanism for the task of image colorization in "Super-attention for exemplar-based image colorization" https://link.springer.com/chapter/10.1007/978-3-031-26284-5_39



Thank You for Your Attention