Colorization of Black and White Images using DNNs

Project Report

Aymaan Shahzad - 22B1520

Introduction:

This project explored the fascinating application of Deep Neural Networks (DNNs) in colorising black and white images. By utilizing PyTorch as the primary library, we embarked on a journey through the fundamentals of Python and machine learning, starting with simple linear/logistic regressions and gradually ascending to the powerful realm of Convolutional Neural Networks (CNNs).

Technical Journey:

1. Building the Foundation: We laid the groundwork by establishing a comprehensive understanding of Python, including its essential libraries like NumPy, Pandas, and Matplotlib. This provided the basic tools for data manipulation, analysis, and visualisation.

2. Linear/Logistic Regressions: We implemented linear and logistic regressions as the initial stepping stones. This provided a firm grasp of prediction, error functions, and gradient descent optimisation.

3. Intro to CNNs: We transitioned to the intricate world of CNNs, dissecting their architecture layer by layer. We studied the roles of various components like convolutional layers, pooling layers, activation functions, and backpropagation in extracting and interpreting spatial features from data.

4. Understanding the Network: Our focus shifted to understanding how different aspects of the network contribute to its performance. We analyzed the impact of error functions like Mean Squared Error (MSE) and Cross-Entropy on learning and prediction accuracy. We explored the influence of learning rate, epochs, and batch sizes on optimisation and convergence.

5. Challenges faced: Throughout the project, we encountered common hurdles such as data format inconsistencies, mismatched dimensions, and difficulties implementing specific layers efficiently. We learned to troubleshoot and refine our model for optimal performance.

Project Highlights:

- Successful Implementations: We successfully built and trained a CNN model capable of colorizing black and white images, demonstrating the power of DNNs in image processing tasks.
- Deepened Understanding: The project offered valuable insights into the workings of CNNs, equipping us with a strong foundation for further exploration in deep learning.
- Problem-solving Skills: By overcoming technical challenges, we honed our problem-solving skills and learned to adapt to the complexities of machine learning projects.

Future Directions:

- Enhancing Accuracy: Exploring advanced CNN architectures and hyperparameter tuning techniques to improve the colorization accuracy and preserve natural image details.
- Style Transfer: Implementing style transfer algorithms to enable users to customize the colorization process and evoke specific artistic styles.
- Real-world Applications: Investigating the integration of colorization models into video processing or historical photograph restoration projects.

Conclusion:

This project was a rewarding journey into the fascinating realm of colorizing black and white images using DNNs. We gained valuable experience in Python programming, built a strong foundation in CNNs, and successfully tackled technical challenges, equipping ourselves with essential skills for further exploration in the field of deep learning. As we move forward, we remain excited to delve deeper into this evolving technology and explore its potential for creative expression and real-world applications.

FINAL PROJECT REPOSITORY REPORT

Introduction

Autoencoders, a special type of deep learning architecture, consist of two networks: an encoder and a decoder. The encoder learns a reduced-dimensional representation of input data through CNN and downsampling, while the decoder attempts to regenerate the data from these representations using CNN and upsampling. Autoencoders find applications in anomaly detection, image denoising, and image colorization. In this project, landscape images are colorized using an autoencoder.

Image Colorization

Traditional image colorization methods require significant human effort, time, and skill. **Autoencoders**, specifically convolutional neural networks (CNNs), have simplified this task. They distill salient features from an image and regenerate it based on these learned features.

Code Overview

The provided code implements an image colorization task using an autoencoder. The landscape images are loaded, resized, and converted to arrays. Grayscale and color images are processed and prepared for training and testing. The model has encoding and decoding layers, followed by training and evaluation.

Dataset

The dataset consists of landscape images available in the "**landscape-image-colorization**" Kaggle dataset. Images are loaded from the 'color' and 'gray' folders, representing color and grayscale versions, respectively.

Model Architecture

The autoencoder model is defined with encoding and decoding layers. The encoding layers use convolutional operations and downsampling, while the decoding layers use transposed convolutional operations and upsampling.

Model Definition

The model is defined using the **Keras**. It consists of encoding and decoding layers. Encoding layers downsample the input image, while decoding layers upsample to reconstruct the colorized image.

Model Training

The model is compiled using the **Adam** optimizer with a learning rate of **0.001**. **Mean absolute error** is chosen as the loss function, and accuracy is used as a metric. The model is trained using the `fit` method on the training data (`train_g` and `train_c`). Training is performed for **50 epochs** with a **batch size of 50**.

Prediction and Visualization

The code includes functions to plot colour images alongside their grayscale counterparts and model predictions. Predictions are generated for a subset of test images and visualized using **Matplotlib**.

Results

The colorized images are plotted alongside their grayscale and original color counterparts. The model's predictions are visualized for a set of test images.

Issues

The code is a modified version of a Kaggle source, and the original repository had errors. Debugging was attempted with mentor assistance but was unsuccessful. The original and modified repositories, along with alpha and beta models, can be found in the provided link: Colorization Using DNNs

References

1. https://www.youtube.com/playlist?list=PLqnslRFeH2UrcDBWF5mfPGpqQDSta6VK4
2. https://pytorch.org/docs/stable/index.html
3. https://chat.openai.com/
4. https://github.com/patrickloeber/pytorch-examples
5. https://pytorch.org/vision/stable/index.html
6. https://netron.app/