Lab report – TNM112

Deep learning for media technology, Lab 3

Oliver Lundin (olilu316)

Lab partner: Filip Bägen (filma379)

Friday 2nd February, 2024 (11:10)

Abstract

This report delves into the application of convolutional neural network (CNN) autoencoders for image denoising, specifically focusing on grayscale and RGB images. The aim was to assess the performance of the autoencoder in denoising images corrupted with Gaussian noise of varying intensities. Results revealed the ability of the model to successfully reconstruct both grayscale and RGB images from noisy counterparts. The model demonstrated comparable denoising capabilities across both types of images. Despite initial expectations of degraded performance for RGB images, the loss function analysis indicated equal performance for RGB, hinting that noise characteristics might not significantly impact the model's denoising capacity across different color channels.

1 Introduction

The aim of this final lab is to explore how to work with autoencoders and specifically work with images and autoencoders. To experiment with this, we used colored and grayscale images, added noise and then tried to denoise them using a CNN autoencoder.

2 Background

The architecture of an autoencoder consists of two main parts: an encoder and a decoder.

- The encoder is the part of the network that compresses the input data into a latent space representation. It takes in the input data, such as images, text, or any other form of data, and transforms it into a lower-dimensional representation. This process involves extracting the most important features from the input and creating a compressed or encoded representation of the data.
- The decoder, on the other hand, takes the compressed representation produced by the encoder and attempts to reconstruct the original input data from this representation. The goal of the decoder is to generate an output that is as close as possible to the original input.

3 Method

In this lab we chose to focus on autoencoders, specifically one that can denoise noisy images for both grayscale and RGB images. The autoencoder is made up of a CNN. The encoder part of the autoencoder is composed of convolutional layers followed by pooling layers. These layers are responsible for extracting hierarchical features from the input images. Convolutional operations involve sliding a filter (kernel) across the input image to detect features like edges, textures, and patterns, while pooling layers reduce spatial dimensions, retaining important information. The decoder part of the autoencoder comprises transposed convolutional layers (upsampling layers) [1] that aim to reconstruct the original input from the learned latent representation. These layers work in reverse, gradually expanding the dimensions of the encoded representation back to the original input size.

3.1 Image Denoising

We used the cifar10 which is a collection of labeled images that is commonly used for machine learning and computer vision research. It consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is split into a training set of 50,000 images and a test set of 10,000 images.

The task is to try both denoising images that are RGB and to denoise images that are grayscale and compare the result of the autoencoder. The noise applied to the images is a gaussian noise with a factor of 0.1 and later a factor of 0.3 to compare the difference.

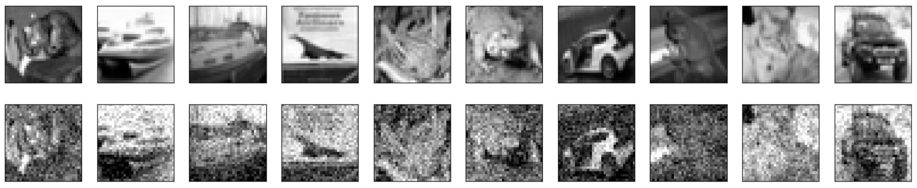

Figure 1: Original grayscale and noisy images with 0.1 noise factor

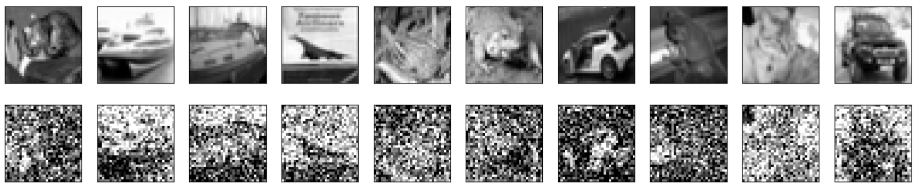

Figure 2: Original grayscale and noisy images with 0.3 noise factor

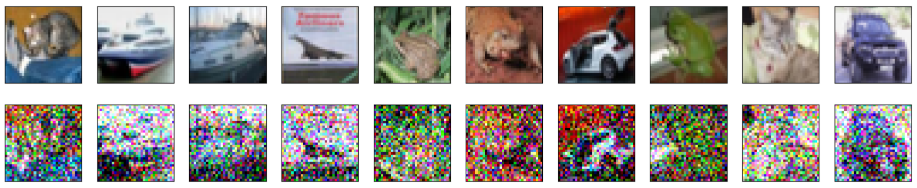

Figure 3: Original RGB and noisy images with 0.3 noise factor

The model used is split up into an encoder and decoder. The encoder consists of four layers [2]:

- Conv2D Layer (E1): 32 filters/kernels of size 3x3 with ReLU activation function. Padding is set to 'same' to ensure the output has the same spatial dimensions as the input.
- MaxPooling2D Layer (E2): Performs max pooling with a 2x2 window, reducing the spatial dimensions by half.
- Conv2D Layer (E3): Another convolutional layer with 32 filters of size 3x3 and ReLU activation. Padding is 'same' to maintain the spatial dimensions.
- MaxPooling2D Layer (E4): Another max pooling layer, reducing the spatial dimensions by half again.

And the decoder:

- Conv2DTranspose Layer (D1): Uses transpose convolution to upsample the input. 32 filters of size 3x3 with ReLU activation and strides of 2 for upsampling.
- Conv2DTranspose Layer (D2): Another transpose convolutional layer with similar specifications to upsample further.
- Conv2D Layer (Output): A final convolutional layer with a single filter when using the grayscale images and with three filters for the RGB images. Uses the sigmoid activation function to squash output values between 0 and 1.

We used a batch size of 128 and 20 epochs to configure our model.

4 Results

The results are displayed as orignal noised images above the denoised images.

4.1 Grayscale

The results for the denoised grayscale images can be seen in figure 4 and figure 5. The reconstructed images show that for both noise factors the model is able to reproduce images that are quite similar to the orignal grayscale one.

4.1.1 Noise Factor 0.1

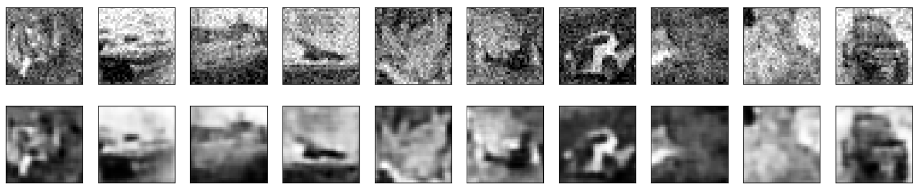

Figure 4: Reconstructed images from noise with 0.1 factor

4.1.2 Noise Factor 0.3

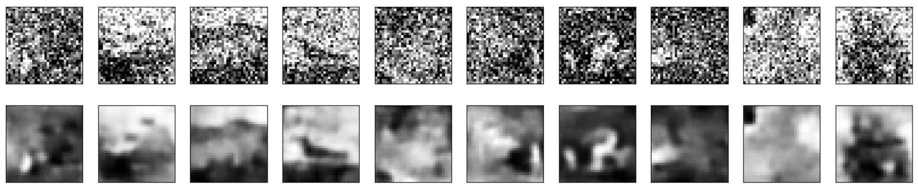

Figure 5: Reconstructed images from noise with 0.3 factor

4.2 RGB

For the RGB images, the results show that the model is still retaining detail even tough the task is more complex because of the extra channels and information the model has to take into account.

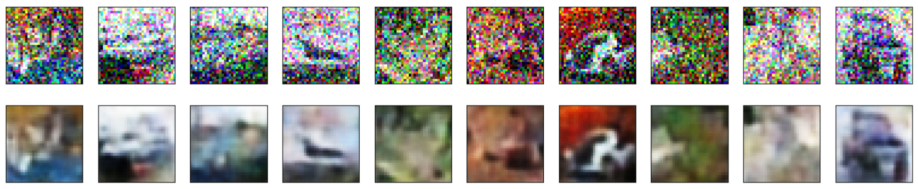

Figure 6: Reconstructed RGB images from noise with 0.3 factor

4.3 Loss

Too compare the results from the grayscale images with the RGB images we can look at the loss functions for the model for the different input data.

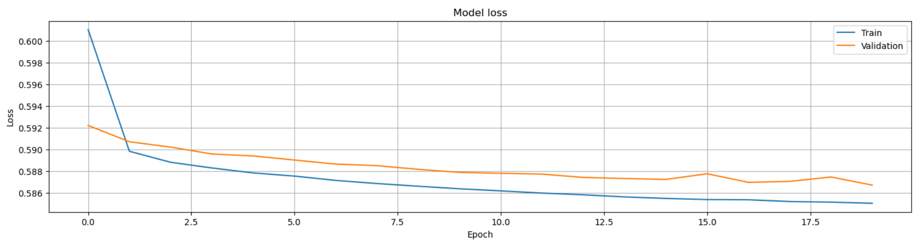

Figure 7: Loss function for grayscale images with 0.3 noise factor

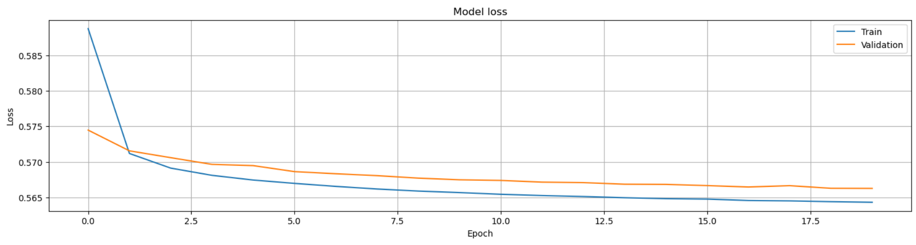

Figure 8: Loss function for RGB images with 0.3 noise factor

5 Conclusion

RGB images contain more information than grayscale images. In RGB, each pixel contains three color channels (Red, Green, Blue), whereas grayscale images have a single channel representing intensity. Denoising RGB images involves preserving and restoring color information along with details, textures, and edges across multiple channels, which we considered would be more difficult for the model achieve and thus we expected a worse performance than with the grayscale images.

However, when comparing the loss functions we see that the loss function for the RGB images shown in figure 8, is slightly lower. Altough the difference is so low that it might not be substantial, it is interesting to see the model perform equally good with 2 more channels. The cause of this could be various things but one key aspect is the type of noise being used. The type and intensity of noise added to grayscale and RGB images might not significantly impact the model's ability to denoise. If the noise characteristics are uniform or similar across channels, the model might learn to handle noise effectively irrespective of color information.

Use of generative AI

I have used generative AI to describe our model in text and formulate various sentences throughout the report. All decisions and conclusions have been made by me.

References

[1] TNM112. Lab 3 – deep learning mini-project, 2023. Lab description – Deep learning for media technology.

[2] TNM112. Teaching session 08 dec – autoencodeer, 2023. Powerpoint slides about autoencoder and notebook.