CNN-Attention Autoencoder for Image Compression

Ilham Prasetyo Wibowo (13520013)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail) : 13520013@std.stei.itb.ac.id

***Abstract*— Image compression is the process of reducing the size of an image without compromising its quality significantly. Image compression is crucial for data transmission and managing data size. This paper explores the use of a CNN-Attention autoencoder, to selectively reduce the size of an image to lower size but maintaining more quality in some areas.**

Keywords—Image, CNN, Attention, Compression

I. Introduction

In this digital era, the growth of visual data, especially images, has given rise to the challenges associated with storage and processing. The volume of image data imposes a significant burden on storage capacities. Image compression is a technique to efficiently manage image size by reducing image size but not losing too much information of the image.

Deep learning is a type of machine learning algorithm which consists of interconnected multiple layers. Convolutional Neural Networks (CNNs) is a type of deep learning with the ability to process grid-like data. Hence, CNNs are effectively used for image related tasks. The notable difference between CNNs and ANNs is CNNs are primarily used for learning patterns within images [1].

Autoencoders are a specific type of neural network, designed to encode the input into a compact and meaningful representation, and then decode it back to the input [2]. Autoencoders facilitate unsupervised learning and capturing the main features of an input, providing means to enable it as a compression method for images.

The CNN-Attention autoencoder is a model to capture the features of the image and reconstruct it back to its original representation. The additional attention mechanism contributes to selectively emphasizing some areas to perform more compression and other areas with less quality reduction. The idea is simple, first we encode the image into its compact representation, then use the attention mechanism to emphasize some parts of the image, and finally approximately reconstruct the image.

II. Theoretical Background

A. Digital Image

Images we see in the real world are analog visual information that our brain perceives. Digital images are representation of images in discrete form. Digital images comprised of pixels where each pixel holds the color information. Images are usually represented in two-dimensional grids. Pixels are the fundamental building blocks of images. They can represent a lot of information depending on the image. Following this, images usually hold a lot of information resulting in a higher burden for storing the image.

B. Image Compression

Image files, especially high-resolution ones, contain vast amounts of data. Storing such data requires significant storage space. Image compression addresses that problem by reducing the size of the image without losing significant quality. The goal of image compression is to reduce spatial storage needs and represent images with near similar quality but with more compact representation.

Applications of image compression is as follows:

1. Reducing the storage needed to store the image in the secondar storage.
2. Reducing the cost of transmitting images in data communication.

There are two primary types of image compression:

1. Lossless compression

   This compression method retains all information of the image, pixel-by-pixel when compressing the file. It uses algorithms to remove the redundancy without losing any loss of information.

2. Lossy compression

   This type of compression reduces file size by losing some information of the data. Lossy compression can get a very low compression ratio compared to lossless compression methods. But, this method loses some quality from the image.

There are lots of different compression algorithms. Some compression algorithms frequently used in the image compression are as follows:

1. Huffman Encoding

   This method is an example of a lossless compression algorithm. It is based on the greedy algorithm. At its core, Huffman encoding algorithm encodes the frequently showed pixel with smaller bit length, and fewer bit length for other pixel values progressively.

2. Run Length Encoding

   The run length encoding method is another common example of lossless compression algorithm. It represents the image as value pairs of pixel value and the length of the pixel present in the image continuously.

3. Joint Photographic Experts Group (JPEG)

   JPEG is a standard compression algorithm for images since 1992. It is a type of lossy compression algorithm. JPEG compression contains a lot of stages including quantization, DCT, and Huffman encoding.

C. Neural Networks

Neural Networks are type of machine learning algorithms which consists of many simple, connected processors called neurons [2]. Each of the neurons produces real activations. The neural network architecture is inspired by how the human brain works. In neural networks, the model learns by flowing the information from the input layer to the output layer in a progressive way. After obtaining the output, it performs a backward computation to adjust the weights of the neurons. The backward computation is called backpropagation.

Deep learning is a type of machine learning that uses multiple layers (hence the name deep) to extract features and learn the information from the input. Deep learning models can have hundreds of layers within a model.

D. Deep Learning for Computer Vision

Deep learning is a suitable architecture for computer vision tasks such as image classification, recognition, or object detection. It is suitable for computer vision due to following reasons:

1. Easy access to large, labeled dataset.
2. Availability of highly performant GPU to increase computing power.
3. There are pretrained models built by experts.

E. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are popular deep learning algorithms, mainly used for inputs with grid-like topology. One of the examples of grid-like topology input is image. CNNs are an architecture for deep learning that learns directly from input data utilizing filters, discarding the need of manual feature extraction.

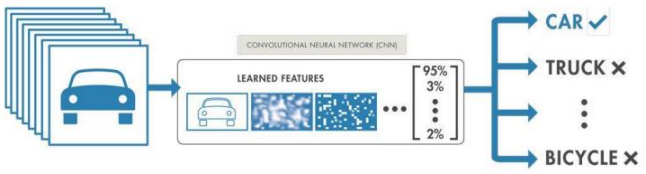


Figure 1. CNN in Image Classification [4]

CNNs mainly comprised of three main elements, convolutional layers (including activation functions), pooling layer, and fully connected layer. Convolutional layer performs convolution to the input image with some defined filters. Each filter produces a single feature map. We can change some hyperparameters that affect the output feature map with number of filters, stride, and padding.

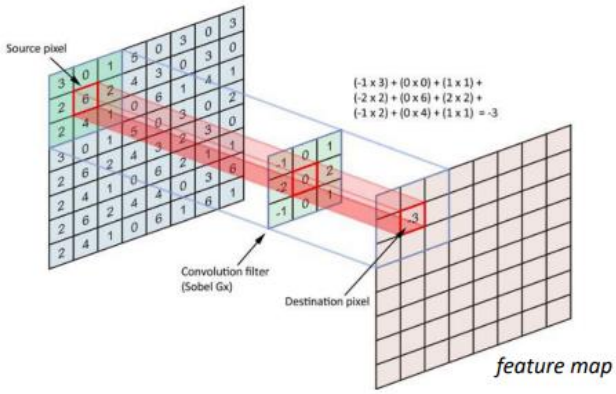


Figure 2. Illustration of Convolution [4]

Pooling layers also performs convolution on the image. But, in the pooling layer, there is no filter used to convolve. Instead, it reduces the size of the input by some defined parameters. There are two types of pooling:

1. Max pooling, returning the maximum value within a window kernel.
2. Average pooling, returning the average value within the window kernel.

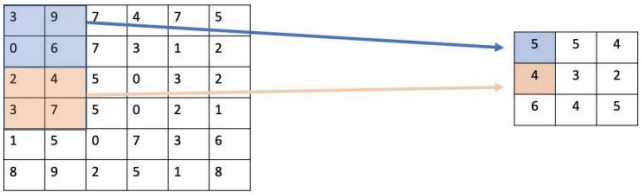


Figure 3. Pooling Illustration [4]

The convolutional layers and pooling layers are mainly used for feature extraction. To classify images, CNNs used fully connected layers. The overall architecture of CNNs is illustrated in the following figure.

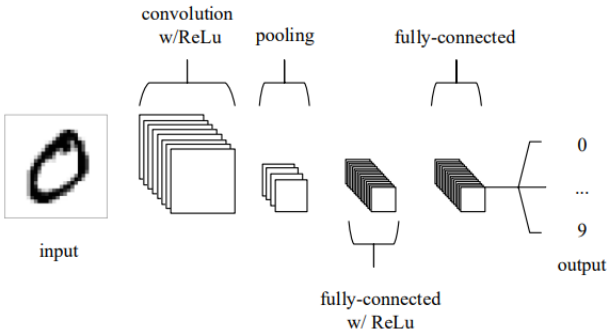


Figure 4. CNN General Architecture [1]

F. Autoencoders

Autoencoders are a specific type of neural network, designed to encode the input into a compact and meaningful representation, and then decode it back to the input [2]. Autoencoder consists of an encoder and decoder, working together to reconstruct input data.

Autoencoder models are trained with unlabeled data. In image case, it can train with only images without any supervision required. It performs encoding to a compact representation of the input, then performs decoding to reconstruct the input. In the training procedure, it compares the output of decoding and the input, then computes some training loss, and updates its weights. Autoencoders can be used as compression method by learning image's compressed representation. One of the examples of autoencoders is denoising autoencoders. The example of denoising autoencoders is in the following figure.

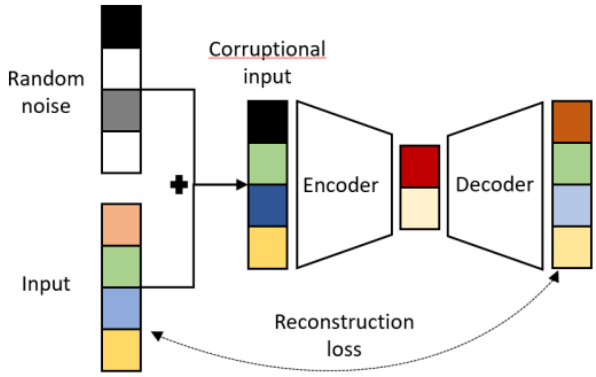


Figure 5. Denoising Autoencoder [2]

G. Attention

Attention mechanism is a component in neural networks that enable models to focus on specific parts of the input data. Spatial attention, on the other hand, refers to a mechanism that allows models to selectively focus on specific spatial regions or locations in the input data. Self-attention refers to an attention mechanism that connects positions within a sequence to generate a representation of that sequence [3]. Spatial attention is especially useful to be applied on image data. Within CNNs, attention assists in focusing on specific image regions, allowing the network to work on that region more attentively.

III. MODEL IMPLEMENTATION

There are three stages to build the final model. First, we need to acquire the dataset. Second, code the model and define the architecture. And lastly, evaluate the model visually and computationally.

A. Dataset

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI

The dataset used for this experiment is beans dataset taken from hugging face. The dataset contains bean leaf images with images of healthy and diseased bean leaves. The training dataset contains 1295 images with each image is 500 by 500 pixels. Before going into the model, we need to preprocess the data first with the following code.

```
from datasets import load_dataset

dataset = load_dataset("beans")

image_paths =
dataset['train']['image_file_path']

def load_images(paths, target_size=(100,
100)):
    images = []
    for path in paths:
        img = Image.open(path).convert('L')
        img = img.resize(target_size)
        img = np.array(img)
        img = img.astype('float32') / 255.0
        images.append(img)
    return np.array(images)

images_np = load_images(image_paths,
target_size=(500, 500))

autoencoder.fit(images_np, images_np,
epochs=10, batch_size=32, shuffle=True)
```

B. Model

The main model architecture consists of encoder and decoder. But we train it altogether within a single training procedure. The developed model architecture is in figure 6.

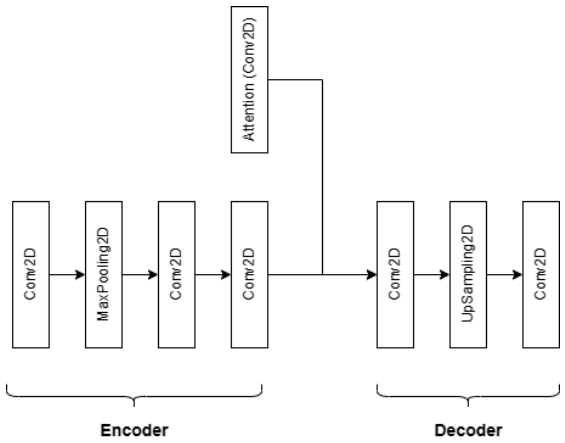


Figure 6. Model Architecture

The architecture as well as the code is as follows:

```
input_img = Input(shape=input_shape)

x = Conv2D(32, (3, 3), activation='relu',
padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(16, (3, 3), activation='relu',
padding='same')(x)
encoded = Conv2D(1, (3, 3),
activation='relu', padding='same')(x)

attention = Conv2D(1, (3, 3),
activation='sigmoid',
padding='same')(encoded)
attention = UpSampling2D((2, 2))(attention)

def resize_attention(x):
    return tf.image.resize(x,
(K.int_shape(encoded)[1],
K.int_shape(encoded)[2]))

inverted_attention = Lambda(lambda x: 1 -
x)(attention)
resized_attention =
Lambda(resize_attention)(inverted_attention)

encoded_attention = Multiply()([encoded,
resized_attention])

x = Conv2D(16, (3, 3), activation='relu',
padding='same')(encoded_attention)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3),
activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam',
loss='binary_crossentropy')
```

The encoder architecture consists of three convolutional layers and a single max pooling layer. The decoder has two convolutional layers and a single up sampling layer. Between the encoder and the decoder, there is an attention mechanism to capture the features of the image with convolutional layer, and then multiply it with the encoded features before feeding it to the decoder.

C. Training

Training the model requires a simple step using the whole dataset as the training data. Because of the limitations of the hardware we use, we set the epochs to 10 epochs and batch size of 32.

```
autoencoder.fit(images_np, images_np,
epochs=10, batch_size=32, shuffle=True)
```

IV. Experiments Result

A. Test Result

At each epoch of the training procedure, the model produced around 0.6 of loss. The achieved loss metric indicates that the autoencoder successfully minimized the difference between the reconstructed images and the original input images. An example of using the training data as input is in the following figures.

Figure 7. Original Test Image

Figure 8. Compressed Test Image

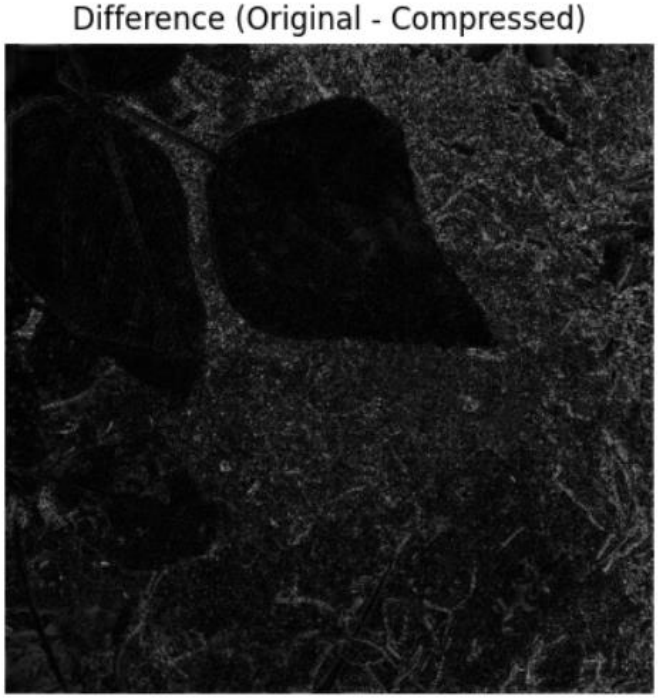


Figure 9. Difference Between Original and Compressed

From the figures, we can see that the image produced has different depths of compression. In the difference image, we can see that some areas are darker than the others indicating that some areas are compressed more than the lighter areas. The compression ratio of the above compressed image acquired from the compression process is 58.9 percent, with the details in figure 10.

```
File size of the original image: 205111 bytes
File size of the compressed image: 120902 bytes
Nisbah pemampatan : 58.94466898411104 persen
```

Figure 10. Compression Ratio of Test Image

For more comparison, we can take the output of only the encoder before feeding it into the decoder.

Figure 11. Encoded Image

As we know, the encoder gives a lot of compression ratio compared to the final output of the decoder. The compression ratio of the encoding process is as follows.

```
File size of the original image: 205111 bytes
File size of the compressed image: 55945 bytes
Nisbah pemampatan : 27.275475230484957 persen
```

Figure 12. Compression Ratio of Encoded Image

But we can see that the image is not similar at all compared to the original image. In fact, it is almost a completely different image. Now if we compare between the attention map and the difference image, we can see that the two of them correlates to each other.

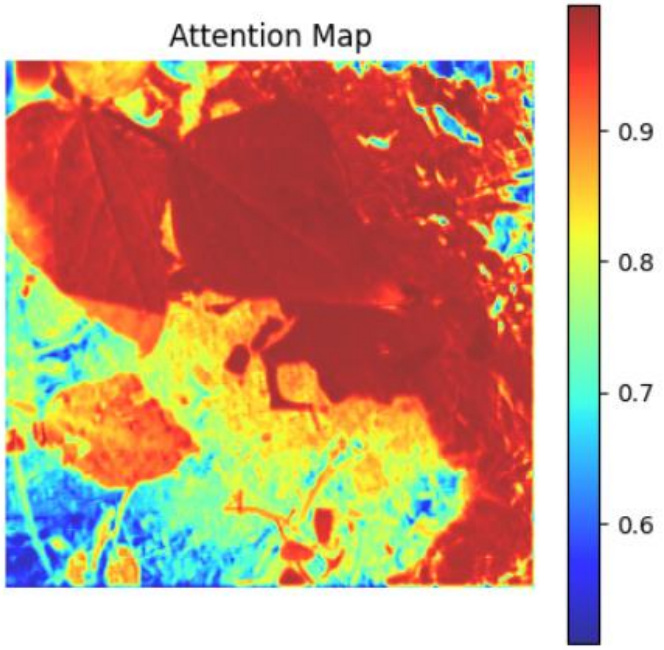


Figure 13. Attention Map

B. *Compression Size*

To further test the performance of the model, we tested it on test data with128 images and computes the average of the compression ratios. We got an average of 68.49 percent of compression ratios.

```
Rata-rata nisbah pemampatan : 0.6849762403517835
```

Figure 14. Average Compression Ratios

V. Conclusion

In this experiment, we implemented an autoencoder model, featuring encoder, decoder, and an attention mechanism in between. Experimental results show that the autoencoder can be used as a compression method. The selective mechanism provided by the attention in the model gives selective focus on the reconstruction procedure. In the compression test, we show that the compression rate mean is around 68 percent.To enhance further experiment, a more complex model architecture might perform better as it is able to explore deeper into the image. Additionally, exploring regularization techniques and different attention mechanisms could potentially refine the selective compression.

Code Repository

The implementation of CNN-Attention autoencoder can be accessed at the following link :

https://colab.research.google.com/drive/1l41fXKYSPPv99csGcTb6-6HkeDDtf_Vr?usp=sharing

Acknowledgment

The author wants to express gratitude to Allah SWT for His blessings, grace, and mercy enabling the competition of this paper. Sincere appreciation is also extended to Mr. Rinaldi Munir for invaluable guidance during the IF4073 Image Interpretation and Processing course. Furthermore, the writer conveyed heartfelt thanks to those whose encouragement and assistance contributed to this paper.

References

[1] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," arXiv:1511.08458 [Online]. Available: https://arxiv.org/abs/1511.08458.

[2] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders”, arXiv: 2003.05991 [Online]. Available: https://arxiv.org/pdf/2003.05991.pdf.

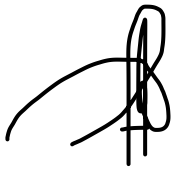
[3] A. Vaswani et al., "Attention Is All You Need," arXiv:1706.03762 [cs.CL], Jun. 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762.

[4] R. Munir, Slide Kuliah IF4073 Interpretasi dan Pengolahan Citra, [Online]. Available: informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/citra23-24.htm.

STATEMENT

I hereby declare that the paper I have written is my own work, not a translation or reproduction of someone else's paper, and it is not plagiarized.

Bandung, 19 December 2023



Ilham Prasetyo Wibowo 13520013