Multimedia & Lab

Image-Denoising

Team 7

김주현 윤호찬
윤동균 서경호

CONTENTS

INTRODUCTION

Digital images are normally prone to additive white Gaussian noise
During image acquisition due to electronic circuits.

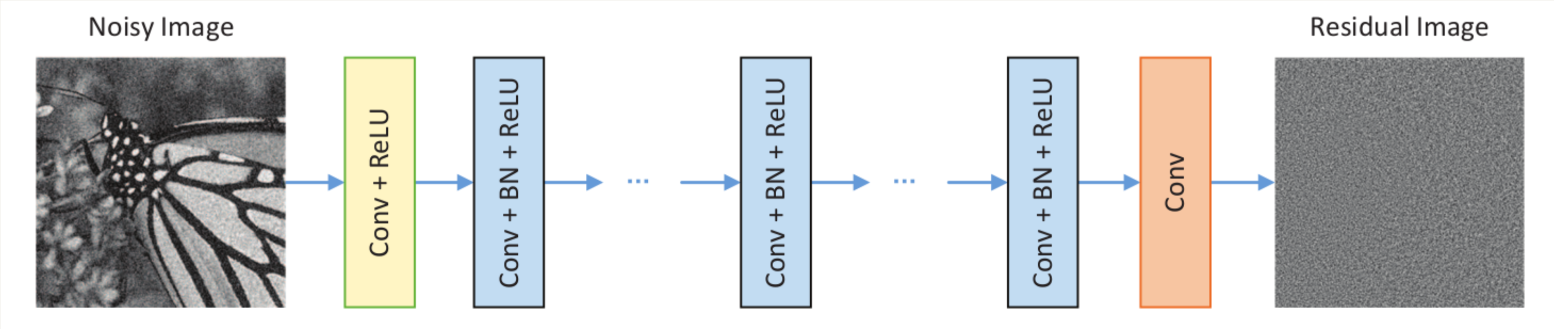
This noise increases over time and degrades image quality.
To maintain quality, it is important to reduce or eliminate noise.

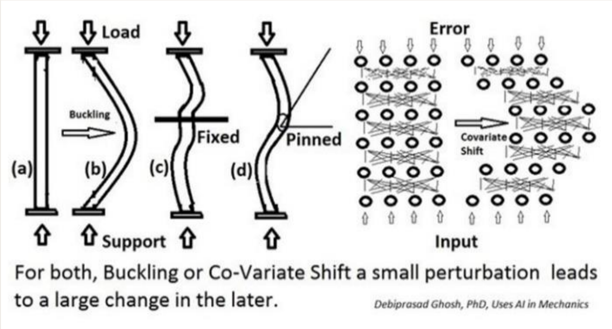
We have made various attempts to eliminate or minimize noise generated.

We tried to restore the image using Autoencoding.
It's a simple way, but We tried to improve the results through various changes.

RELATED WORK

DnCNN





There are three types of layers in DnCNN.

- Conv + ReLU
- Conv + BN + ReLU
- Conv

Predicting residential images instead of denied images.
Residual learning and BN are combined to achieve better performance.

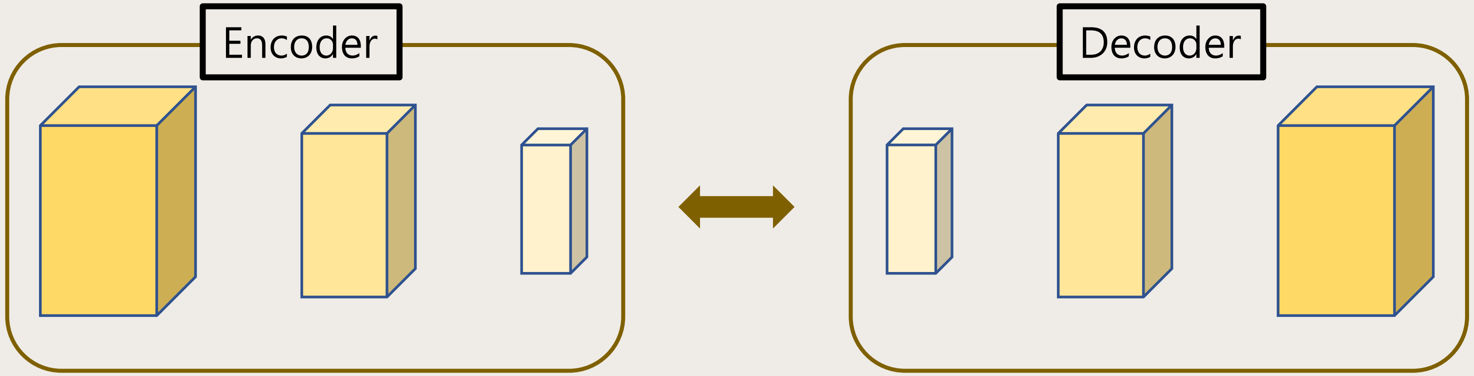
Similar to DnCNN, we used Conv, ReLU, and BN to construct the layer.

The difference is that DnCNN repeats the same hidden layer, but we don't.

https://arxiv.org/pdf/1608.03981.pdf

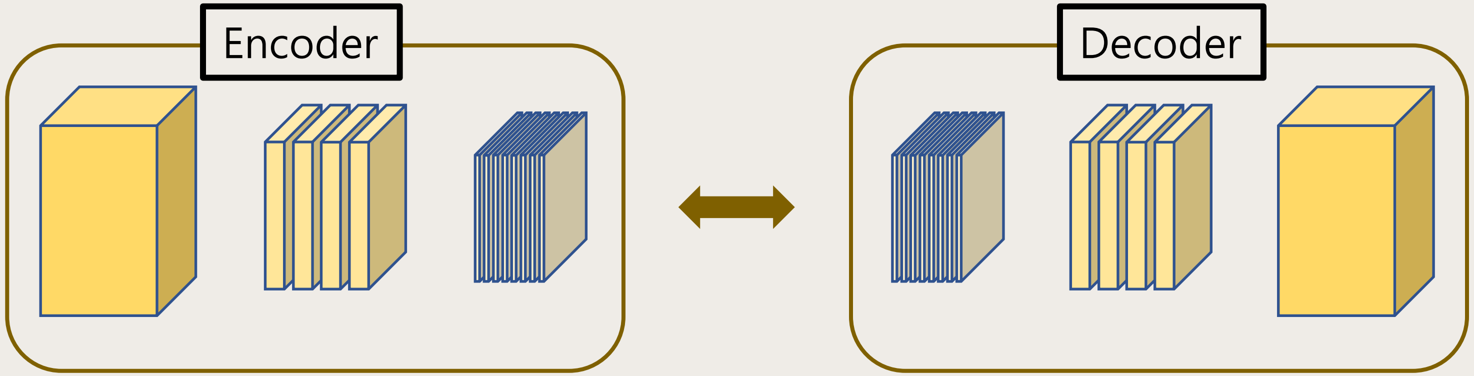
METHODS

We started with the simplest Autoencoder structure.



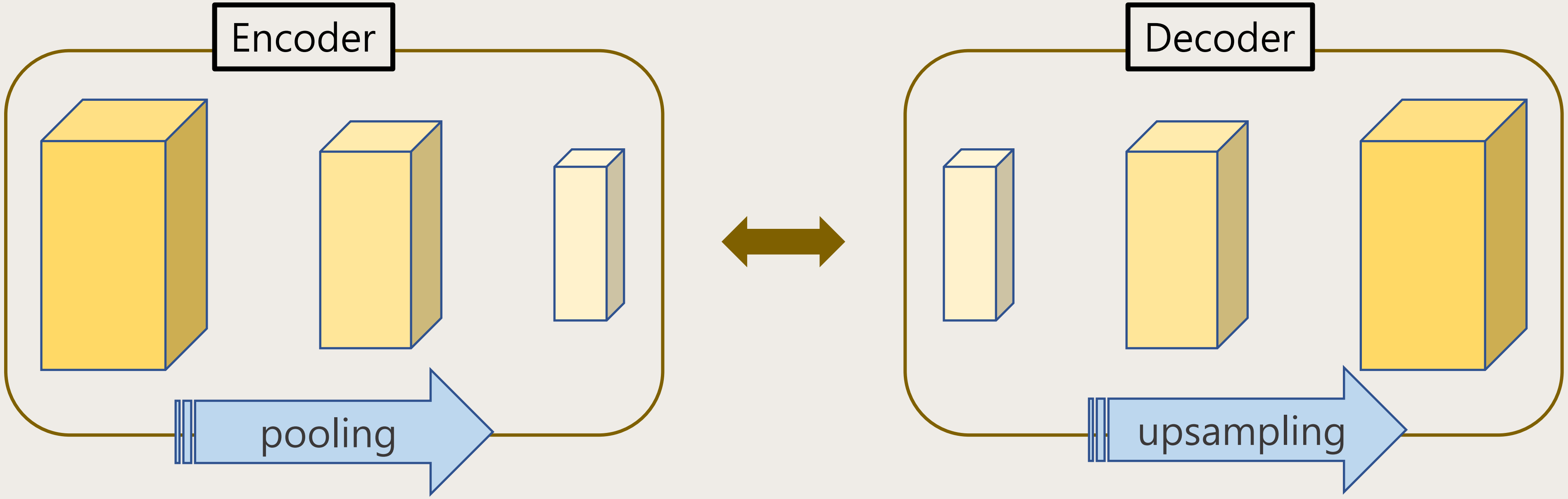
In the basic structure,
encoding and decoding are carried out
using pooling and upsampling.

Instead of using the basic method.
We use Conv that return various out channels



METHODS

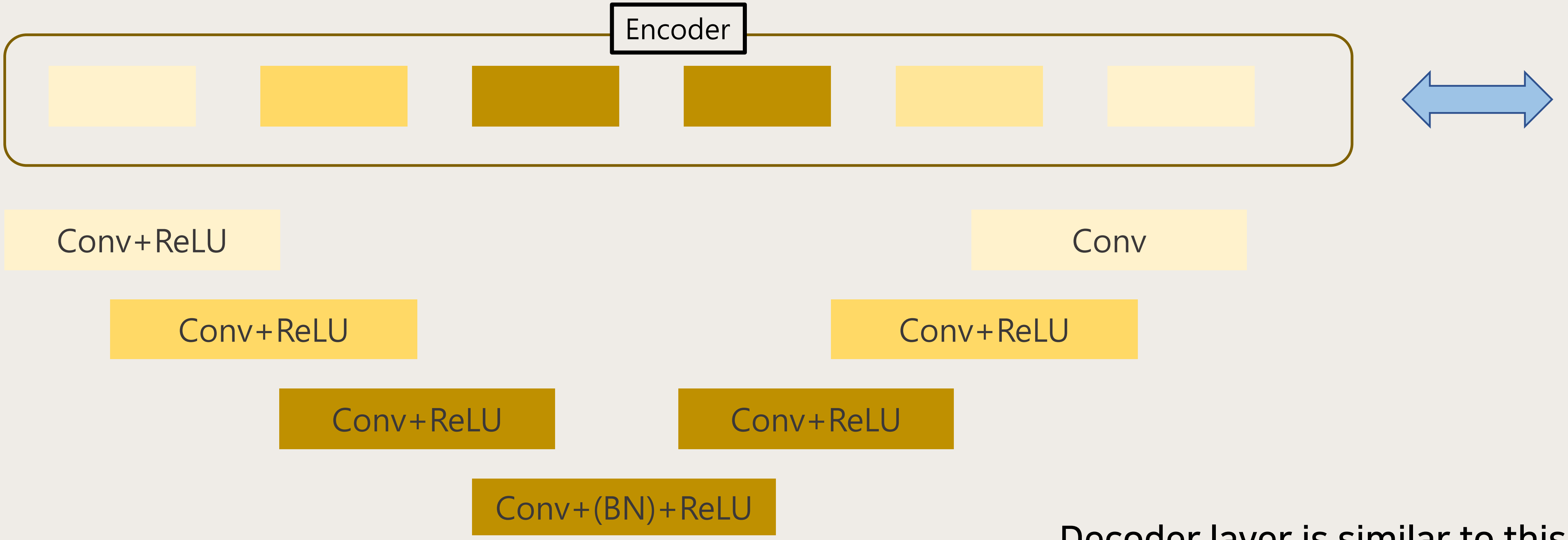
Of course, Pooling can reduce unnecessary parameters and extract features well.
However, in order to return a denoised image, Upsampling is required during decoding.



Upsampling is not lossless.

METHODS

So, we built a layer in the following way.



Decoder layer is similar to this.

METHODS

Network Code

Reference code

```python
import torch.nn as nn
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 256),
            nn.ReLU(True),
            nn.Linear(256, 64),
            nn.ReLU(True))
        self.decoder = nn.Sequential(
            nn.Linear(64, 256),
            nn.ReLU(True),
            nn.Linear(256, 64),
            nn.Sigmoid())

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Auto Encoder
(Reference code)

AutoEncoder denoising network

```python
import torch.nn as nn
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()


        self.encoder = nn.Sequential(
            nn.Conv2d(3, 96,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(96, 256,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(256, 384,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(384, 384,  kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(384),
            nn.ReLU(),
            nn.Conv2d(384, 12, kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(12, 3,  kernel_size=3, padding=1, bias=False),
        )
        self.decoder = nn.Sequential(
            nn.Conv2d(3, 12,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(12, 384,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(384, 384,  kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(384),
            nn.ReLU(),
            nn.Conv2d(384,256, kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(256,96,   kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(96,3, kernel_size=3, padding=1, bias=False),

        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return encoded, decoded
```

Auto Encoder Network structure
for Image Denoising

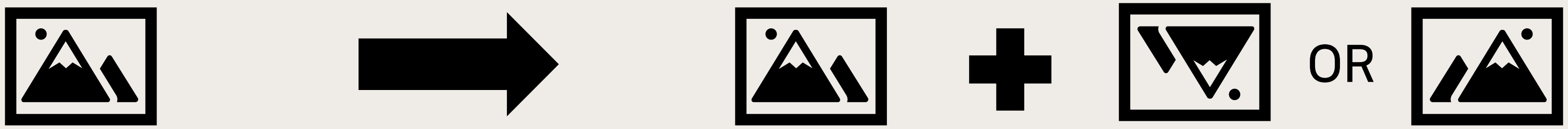
batch normalization 추가

```python
import torch.nn as nn
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()


        self.encoder = nn.Sequential(
            nn.Conv2d(3, 96,  kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(96),
            nn.ReLU(),
            nn.Conv2d(96, 256,  kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.Conv2d(256, 256,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(384, 384,  kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(384),
            nn.ReLU(),
            nn.Conv2d(384, 12, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(12),
            nn.ReLU(),
            nn.Conv2d(12, 3,  kernel_size=3, padding=1, bias=False),
        )
        self.decoder = nn.Sequential(
            nn.Conv2d(3, 12,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(12, 384,  kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(384, 384,  kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(384),
            nn.ReLU(),
            nn.Conv2d(384,256, kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(256,96,   kernel_size=3, padding=1, bias=False),
            nn.ReLU(),
            nn.Conv2d(96,3, kernel_size=3, padding=1, bias=False),

        )

    def forward(self, x):

        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return encoded, decoded
```

Auto Encoder Network structure
for Image Denoising
with batch normalization

METHODS

We doubled the size of the test set
by randomly flipping the test image horizontally or vertically.



```
[ ] import os
    import zipfile
    import tqdm

    file_name = "Multimedia_dataset.zip"
    zip_path = os.path.join('/content/drive/MyDrive/Multimedia_dataset.zip')

    !cp "{zip_path}" .
    !unzip -q "{file_name}"
    !rm "{file_name}"
```

Data Augmentation

```
[ ] from PIL import Image
    import random

    arg_root_path = "/content/"
    arg_train_dir = os.path.join(arg_root_path, "train")
    arg_img_list = [os.path.join(arg_root_path, "train", dirs) for dirs in os.listdir(arg_train_dir)]

    if input()=="ok":
      print("start arg")
      for idx,file_name in enumerate(arg_img_list):
        rand_ = random.randrange(1, 3)
        arg_image = Image.open(file_name)
        if rand_ == 1:
          flip_img = arg_image.transpose(Image.FLIP_LEFT_RIGHT)
          flip_img.save(arg_train_dir+"/fliped_%06d.png" %(idx+1))
        elif rand_ == 2:
          flip_img = arg_image.transpose(Image.FLIP_TOP_BOTTOM)
          flip_img.save(arg_train_dir+"/fliped_%06d.png" %(idx+1))
        else:
          print("wrong rand num")
          break

      print("done")
```

In addition,

We tested using Adam as optimizer while changing lr and eps values.

We tested within limited resources while changing the batch size and epoch values.

EXPERIMENTS

Result samples


noised


denoised


noised


denoised

score

52	Team7	kjh99723	AutoEncoder		2021-05-26 16:21:42.849414	28.981156	0.918853	0.288882

CONCLUSION

What's good about this project

- Building a network for image denoising in a simple structure using Auto encoder.
- Direct experience with various deep learning networks such as DnCNN, AlexNet, Auto Encoder, etc.

What’s unfortunate about our project

- Google Colaboratory was used as a development environment, and limited resources allowed it to run up to its current parameters.
- We have yet to find any distinct ways to improve the performance of our network.

The potential for this network to evolve

- Increase the number of epochs and add more layers.
- Increase the train data further using more different augmentation methods.
- It can be applied in other fields besides image Denoising by using another layer instead of convolution.

Multimedia & Lab

Thank you☺