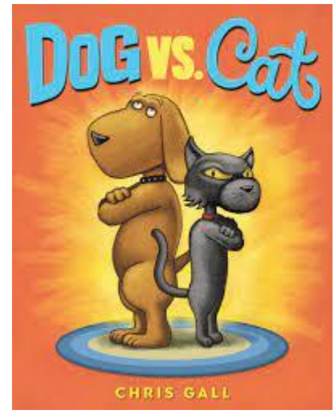



# Auto-Encoder
# "Dog vs Cat" Dataset

**Moriya Bitton || Yuval Ben Yaakov**

## Abstract

Neural networks are used in many tasks today. One of them is images processing. Auto-Encoder is a very popular neural network for such problems. Denoising Auto-Encoder is an important Auto-Encoder because in some tasks we need a preprocessed image to get a less noisy result.

Noise reduction of an image can be used for visually impaired people to see better, recognize old handwriting, and many more features.
When the noise in the image is increasing then the chance of a human eye to know what should be in the image is lowered, That's we need to rely on computers and especially ML models that will do the job.
This report describes ways to analyze noisy images and how to reduce that noise.

## Introduction

In this project, we created a model that could study the DNA of the image so he can get a damaged image and complete it by itself the damaged area. in this case, we "sabotaged" our dataset, in the real world damage can be cut, dirt, etc. We will use Auto-Encoder to build, test, and train our model.

## Related work

Autoencoders are surprisingly simple neural architectures. They are a form of compression, similar to the way an audio file is compressed using MP3, or an image file is compressed using JPEG. Autoencoders are closely related to principal component analysis (PCA). Generally, the activation function used in autoencoders is non-linear, typical activation functions are ReLU (Rectified Linear Unit) and sigmoid. The math behind the networks is fairly easy to understand, so I will go through it briefly. Essentially, we split the network into two segments, the encoder, and the decoder.

There are several other types of autoencoders. One of the most commonly used is a denoising autoencoder, which will analyze with Keras later in this tutorial. These autoencoders add some white noise to the data before training but compare the error to the original image when training. This forces the network to not become overfit to arbitrary noise present in images.

## Required background

We need to know how to build the Auto-Encoder model, how to build each sub-model– Encoder and Decoder. for each model we will fit input, who contain image size, color scale (RGB or Gray) and noise size, activation function, loss function, optimizer. to train and test our models we will fit several epochs and batch sizes.

Moreover, we need an anaconda with a Thensorflow version 2.0 and we use the Keras model build function.

# Dataset

| | *Dogs vs Cats* | *fashin_mnist* |
|---|---|---|
| ***[Train : Test] size*** | [17,500 : 7,500] | [60,000 : 10,000] |
| ***Image size*** | 64x64 Grayscale | 28x28 Grayscale |
| ***Normalize pixels to range(0, 1)*** | Yes | Yes |
| ***Noise size*** | 0.3 | 0.4 |
| ***link*** | https://www.kaggle.com/c/dogs-vs-cats/data | https://github.com/zalandoresearch/fashion-mnist |

# Description

For our Auto-Encoder Model, we used "binary cross-entropy" as a loss function with the "Adam" optimizer. To train and test, we used 'Dogs vs Cats' and 'fashion_mnist' datasets.

# Attempts summary

## Fashin_Mnist Dataset:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 14, 14, 32)        9248
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 32)        9248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 1)         289
=================================================================
Total params: 28,353
Trainable params: 28,353
Non-trainable params: 0
_________________________________________________________________
```

## Dogs vs Cats Dataset:

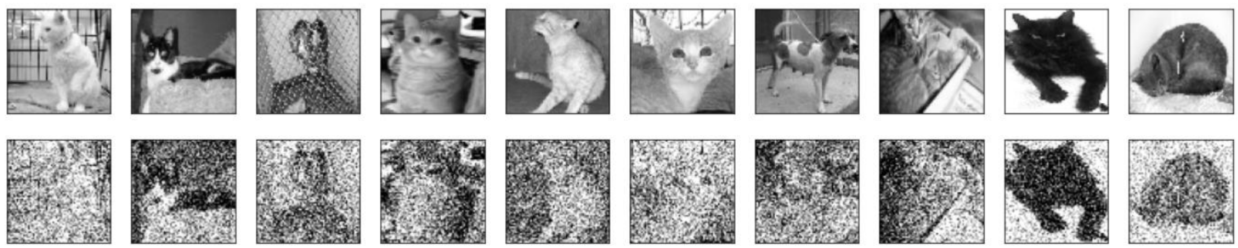
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 64, 64, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 64, 64, 48)        480
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 48)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 96)        41568
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 96)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 192)       166080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 192)         0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 16, 16, 32)        55328
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 32, 32, 32)        9248
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 64, 64, 32)        9248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 1)         289
=================================================================
Total params: 282,241
Trainable params: 282,241
Non-trainable params: 0
_________________________________________________________________
```

# Results

The results that came out of the mode seems promising while the input was completely scrubbed images and outputting the correct image.

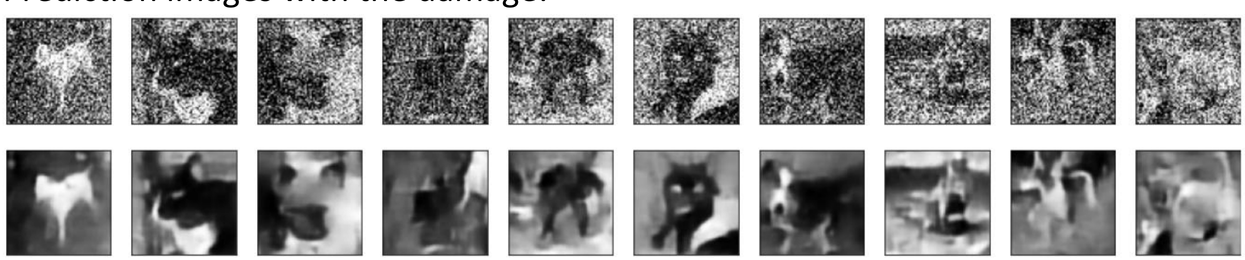
**Dogs vs Cats Dataset:**

Original photos before and after the damage:



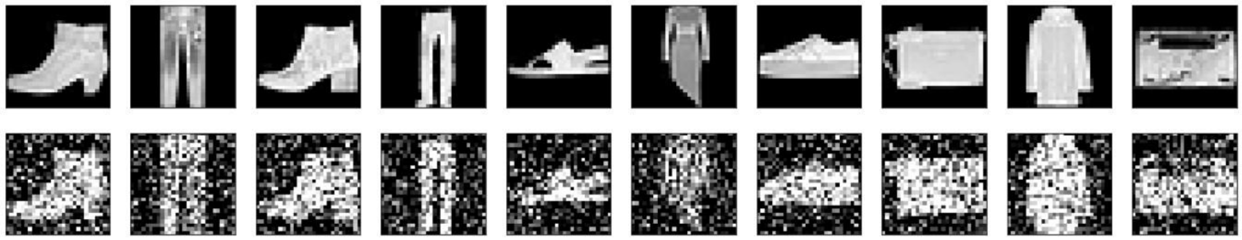
Prediction images without the damage:

Prediction images with the damage:



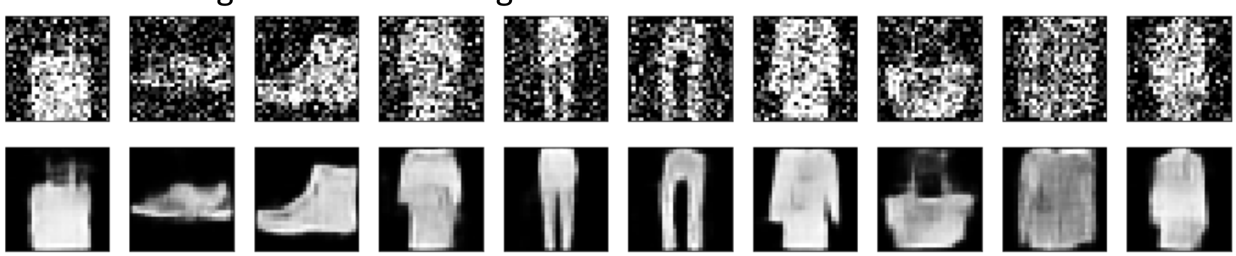
**Fashin_Mnist Dataset:**

Original photos before and after the damage:

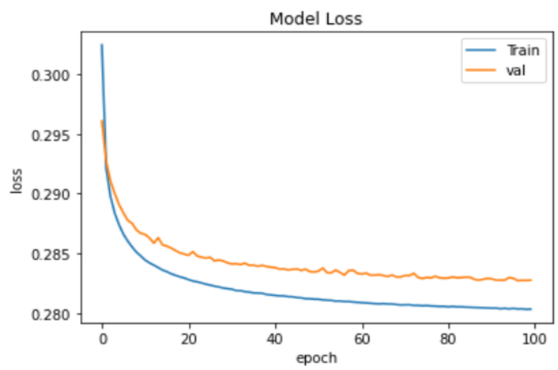
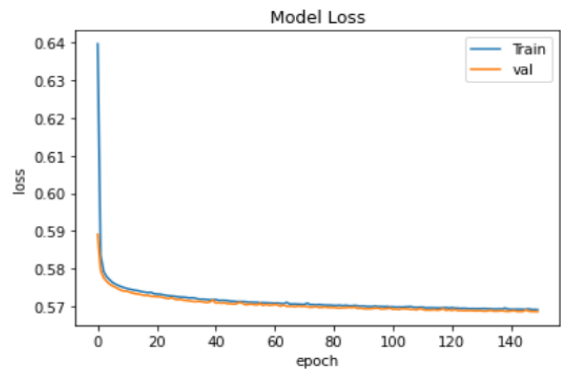


Prediction images without the damage:

Prediction images with the damage:



# Estimation of loss

| **Fashin_Mnist Dataset:** | **Dogs vs Cats Dataset:** |
| --- | --- |
|   |   |

# Conclusions

In this project, we are presenting a solution for fixing noise from images using deep learning.
Build a model that takes a damaged image and use the Auto-Encoder model to learn the Image DNA and "solve" the missing area of the image.

# Github link

https://github.com/MoriyaBitton/Deep_learning_and_natural_language_processing

# Thank for reading! 😋