ML Projects

In this document you can find three of my representative ML projects. This page is just to attach links to the corresponding repositories.

- Neural Style Transfer: https://github.com/TAOGenna/pytorch-neural-style-transfer
- Ionospheric Echo Detection Using Semantic Segmentation: https://github.com/TAOGenna/ionospheric-echo-detection-with-convolutional-neural-networks
- Monte Carlo Tree Search based AI bot: https://github.com/TAOGenna/Connect4-MonteCarlo

For other minor projects not listed in this document refer to: ai notebooks implementation repository

README.md

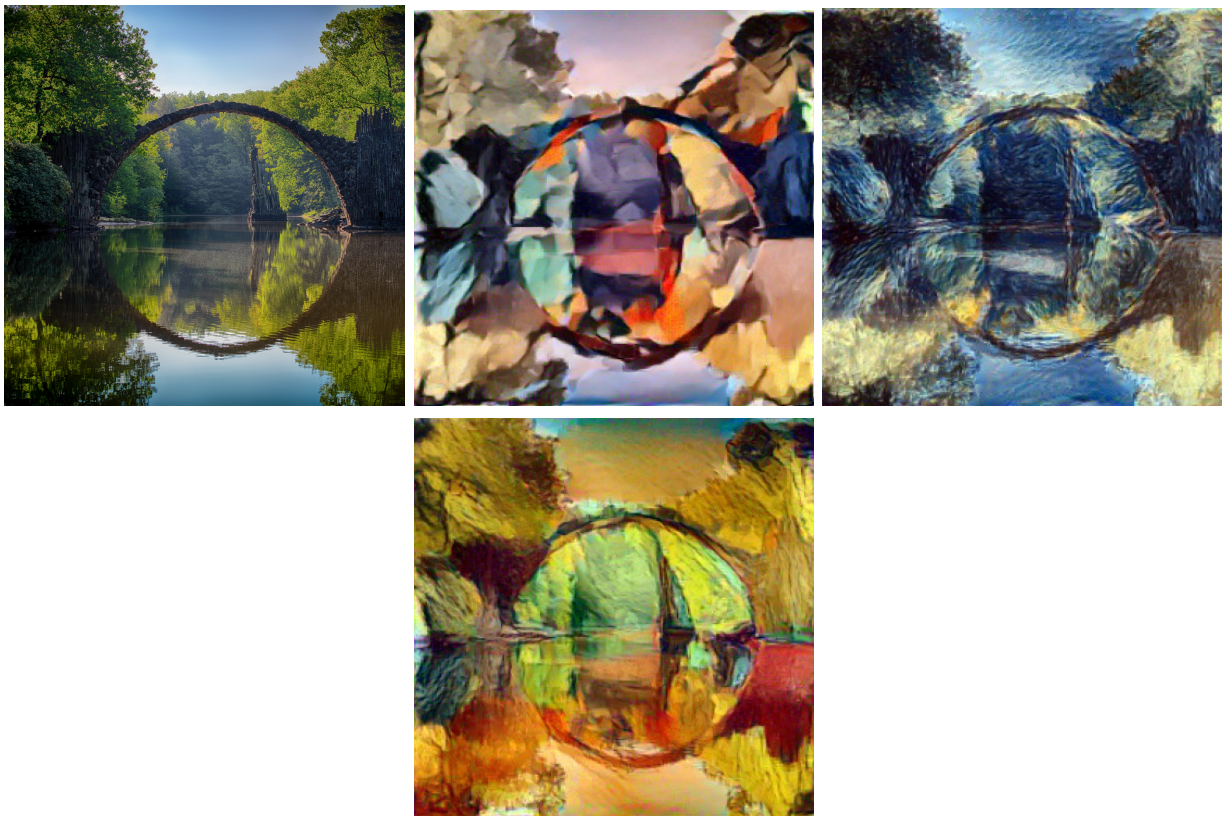
Neural Style Transfer

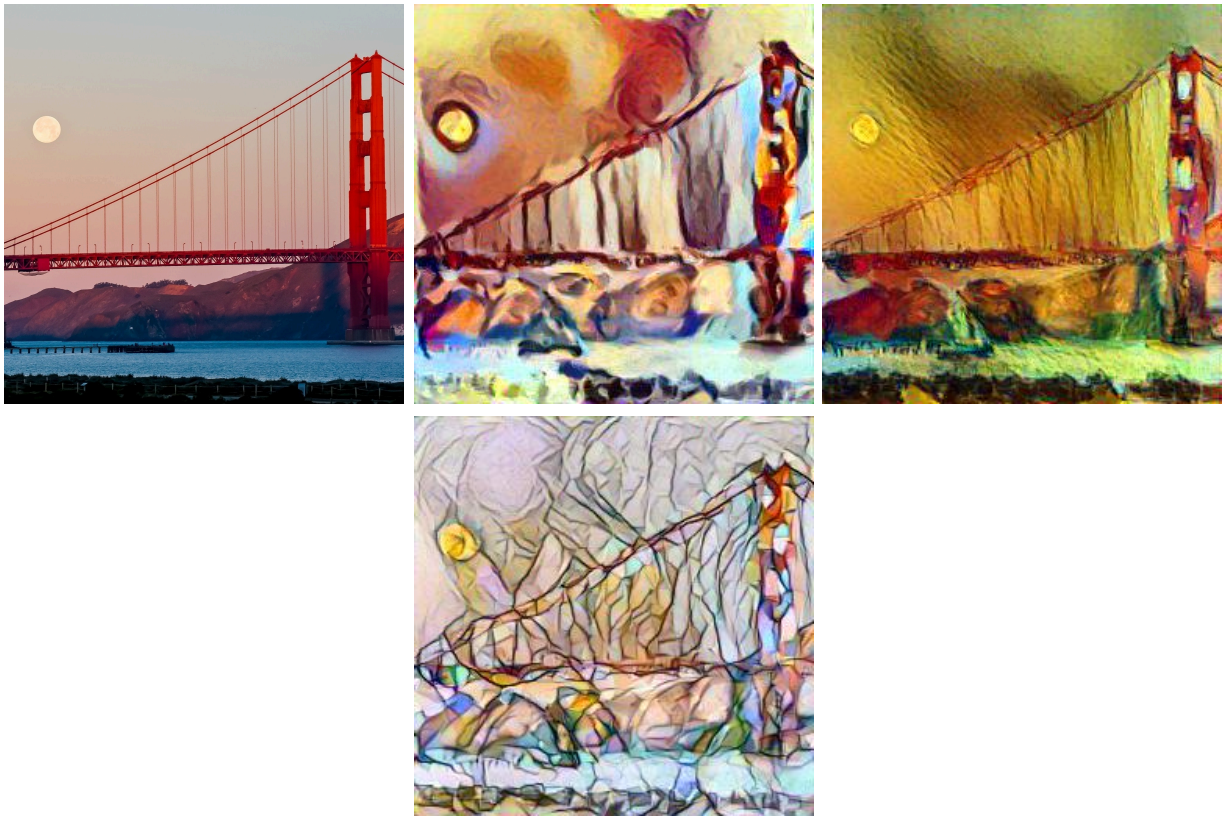
This repo contains a simple ~100 LoC implementation of the *Neural Style Transfer* paper by Gatys et al.

Original Paper's Idea

We have an style image whose texture we want to extract and subsequently embed it into a target image. Using CNNs (usually a Vgg-16/19) this algorithm allows us to produce new images that combine the content of an arbitrary photograph with the appearance of numerous well known artworks.

Examples

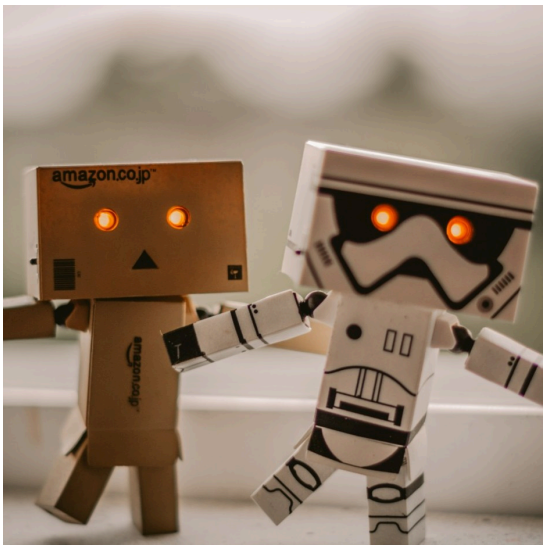


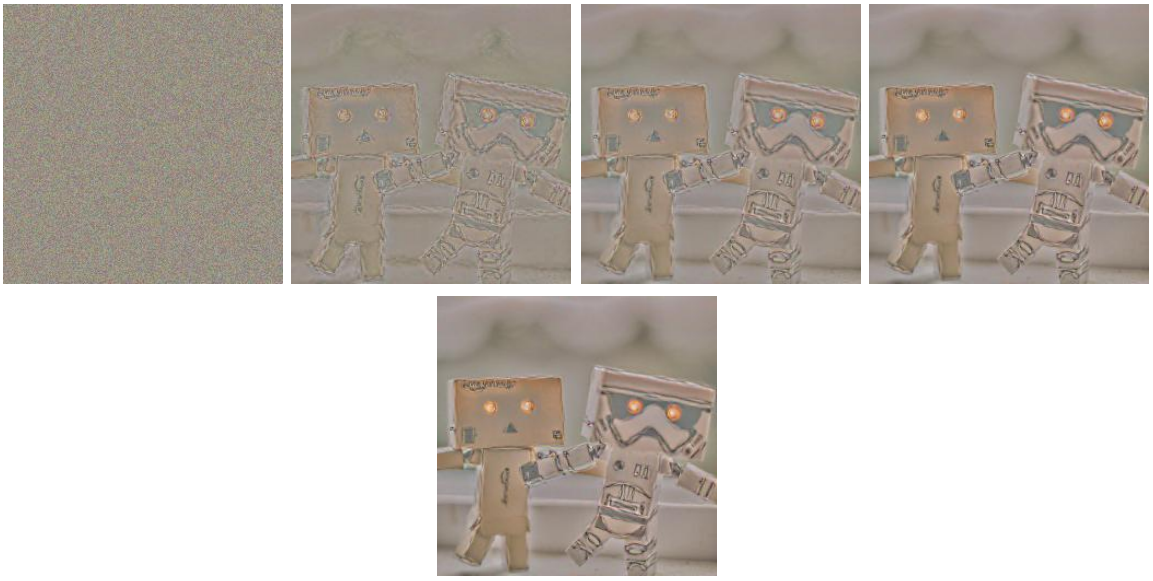


Because of memory issues I had to resize the images to 256x256, but I suspect that better results can be achieved by leaving the images as 512x512 as more details can be learned.

Content Reconstruction

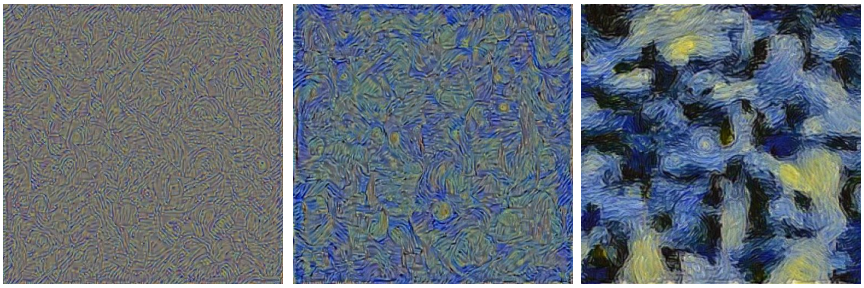
By using tranfer learning from a VGG19 net we can get deep feature representations of the target image, whose details can be reconstructed from a white image (i.e. white noise or a gaussian noise). While shallow layers gets almost perfect reconstruction of the image, deeper layers lose detailed pixel information. The stand alone code to play with content reconstruction can be found in the file `content_reconstruction.ipynb` .





Style/Texture Extraction

For the style extraction we don't want to keep details about the artwork but instead have an overall understanding of the texture/style. This is where *Gram matrices* come into play as the texture model needs to be agnostic to spacial information. See what happens for the artwork of *Starry Night* by Vincent van Gogh:



Usage

- Clone repository
- Change directions of `project_dir = '/home/rotakagui/projects/pytorch-neural-style-transfer'` in the `neaural_style_transfer.ipynb`
- That's about it

Acknowledgements

- Original paper https://arxiv.org/abs/1508.06576
- This project was a suggestion by @gordicaleksa as an intermediate level project. See his channel

README.md

Ionospheric Echo Detection Using Semantic Segmentation

We aim to replicate the experiments described in the paper "Ionospheric Echo Detection in Digital Ionograms Using Convolutional Neural Networks" from the Advancing Earth and Space Science Journal submissions 2021.

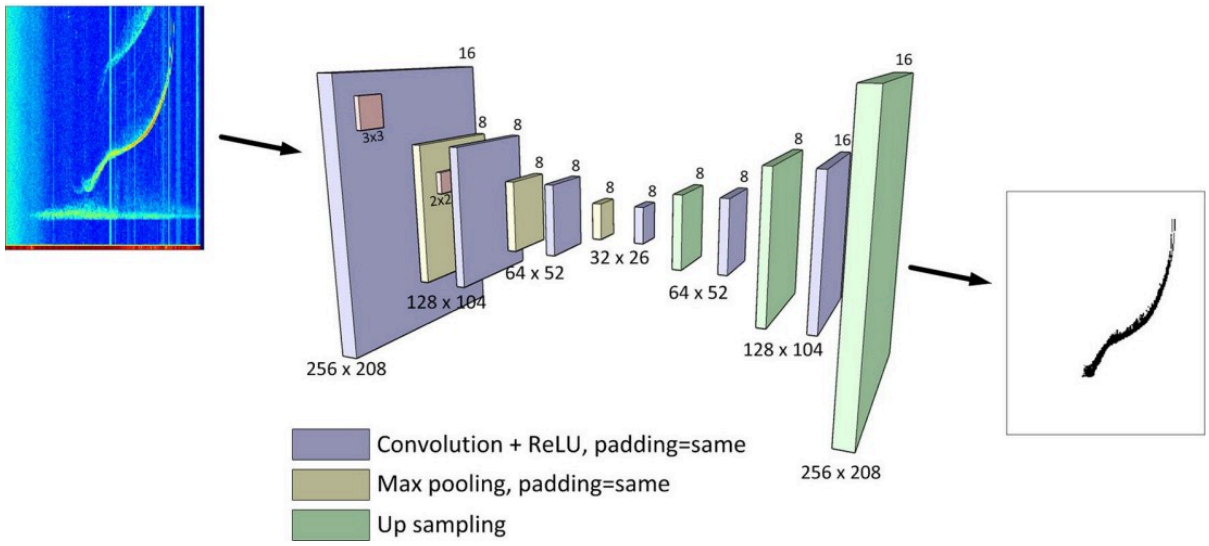
This project was done in my last two weeks at the Radio Astronomy Institute as part of the pipiline for the processing of ionograms measured at the Jicamarca Observatory and Pontifical Catholic University of Peru.

Abstract

An ionogram is a graph of the time that a vertically transmitted wave takes to return to the earth as a function of frequency. Time is typically represented as virtual height, which is the time divided by the speed of light. The ionogram is shaped by making a trace of this height against the frequency of the transmitted wave. Along with the echoes of the ionosphere, ionograms usually contain a large amount of noise and interference of different nature that must be removed in order to extract useful information. In the present work, we propose a method based on convolutional neural networks to extract ionospheric echoes from digital ionograms. From the extracted traces, ionospheric parameters can be determined and electron density profile can be derived.

Encoder-Decoder Architecture

The encoder-decoder is widely used for a number of tasks, trace retrieval included. While the encoder learns or captures features of the image and low its spatial resolution while learning more abstract representations of the image, the decoder upsample the very packed tensor to the original image resolution for further data processing depending on the task. In our case, we want to extract the trace which at the end is nothing more than seeing what pixels are on and off. Then we use the `BCEWithLogitsLoss` loss function as it suits well for binary classification tasks.



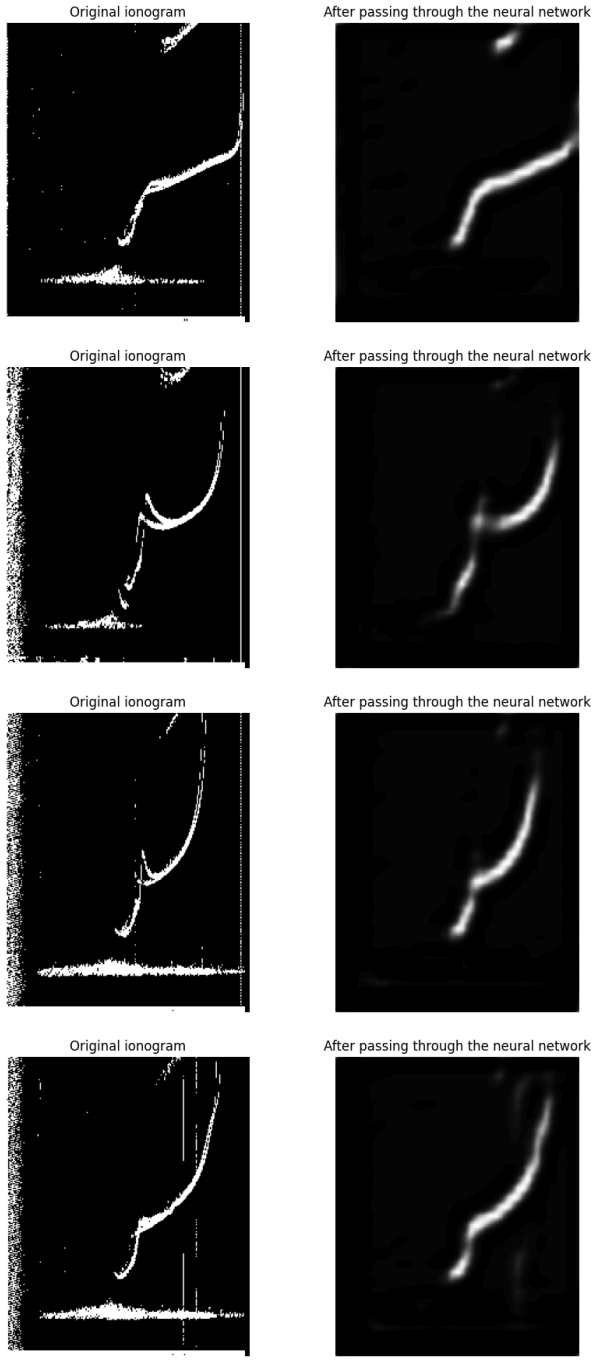
Data

The data consists of `.npz` files. They are masks, i.e. matrices, consisting of zeros and ones indicating noise, or an echo detection. The targets are the traces we want to retrieve. The original data set https://www.kaggle.com/datasets/cdelajara/ionograms is a bit unorganized so I had to curate it myself. You can find it in:

- original ionogram mask
- target maks

Results

The results are promising even without fintunning:



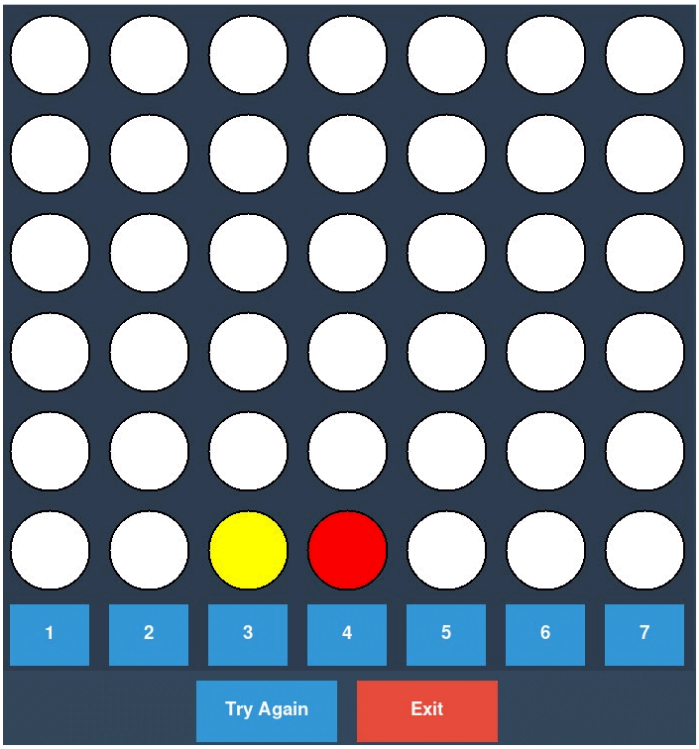
References

- Ionospheric Echo Detection in Digital Ionograms Using Convolutional Neural Networks
- Fully Convolutional Networks for Semantic Segmentation
- U-Net: Convolutional Networks for Biomedical Image Segmentation

README.md

Connect4 - MonteCarlo

A Monte Carlo Tree Search (MCTS) based bot for the game of Connect4



I wanted to learn Reinforcement Learning so I thought this would be a good ~~first~~ exercise. Plus, the algorithm used in this project is present in *MuZero* and *AlphaGo* so I really wanted to learn about it!

Previously I did a minimal version with a simpler game: tictactoe or "michi"(for spanish speakers). If you are a newcomer to RL as I am, I may suggest first revising the K-armed Bandit problem to better understand what we mean by "policy".

The Game

Connect Four is a two-player strategy game in which players take turns dropping colored discs into a vertical grid, aiming to be the first to form a horizontal, vertical, or diagonal line of four discs of the same color.

Monte Carlo Tree Search

Monte Carlo Tree Search is a decision-making algorithm that builds a search tree by simulating random playouts from a given state, progressively refining its choices based on the results. It combines exploration of new moves with exploitation of known successful moves to find optimal actions. In particular, we employ the *Upper Bound Confidence* to set weights between these two.

Depending on the amount of computation budget that you give the program it may behave better or worse as it can have the opportunity to explore a larger space of possible states.

Usage

```
git clone https://github.com/TAOGenna/Connect4-MonteCarlo.git
python3 interface.py
```

If you want the algorithm to search for deeper states, you can change that number in `interface.py` . Search for

```
def ai_move(self, budget=3000):
        if self.game_over:
            return
```

and change the variable `budget` to a bigger number.

Future Work

There is room for improvement beyond the UBC policy. See papers:

- Monte-Carlo Tree Search and Minimax Hybrids, Hendrik Baier and Mark H.M. Winands
- Score Bounded Monte-Carlo Tree Search, Tristan Cazenave and Abdallah Saffidine

Acknowledgments

- Principal reference: A Survey of Monte Carlo Tree Search Methods, Cameron Browne et al.
- To Alfredo de la Fuente. I was looking through his projects and this one seemed suitable for someone trying to learn the basics of RL, so thanks for open-source it and

also for the references