COSC440 Project

Deep learning objectives

- Explore and understand unsupervised learning using an autoencoder
- Implement and test an encoder-decoder architecture on a scientific dataset
- Refine / upgrade network to test different model parameters and report on them such as...
  - custom loss functions (different terms in the reconstruction loss, possibly using a physical model as guide)
  - extra constraints on latent layers
  - different configurations / sizes / functions for encoder/decoder network
  - variational or generative sampling on the latent layer
  - and other possibilities

This project will be completed in 3 parts.

Part 1: Build an encoder/decoder model [10 points]

Explore and understand unsupervised learning using an autoencoder

In this part you will work through building the components of an autoencoder in TensorFlow using extensions to Keras classes. You will train your model to de-noise images and visualize the clustering of the latent vector.

Complete the Convolutional Autoencoders workbook here:
https://drive.google.com/file/d/1jDbKoBWX_ff0lOCjOSTDPBVdC5aL_I1m/view?usp=sharing

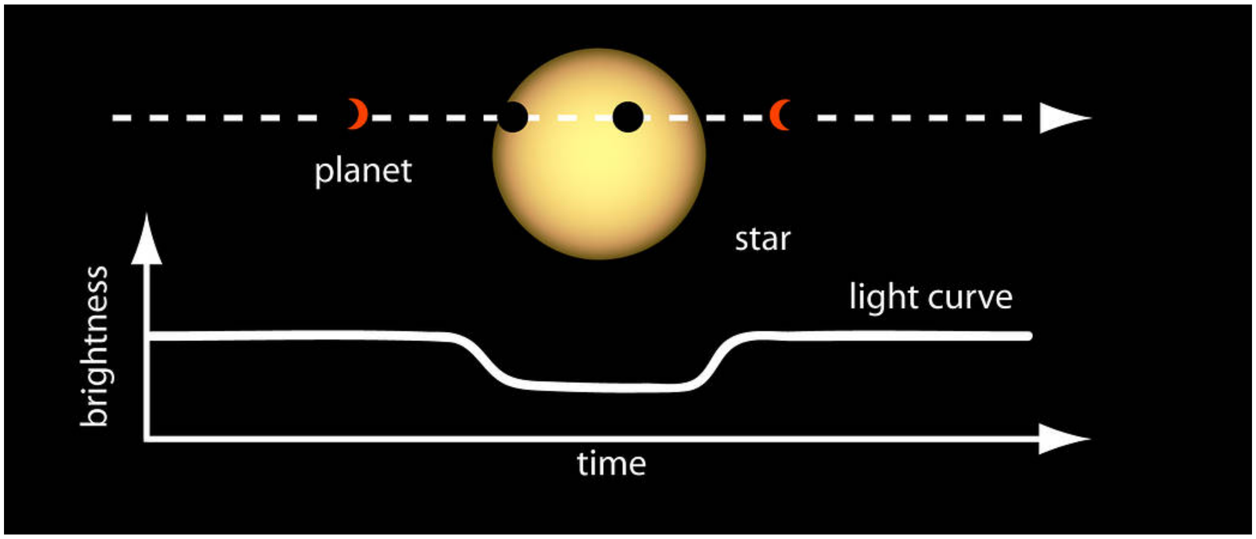
You will then apply and extend this code in Parts 2/3.

Part 2: Implement the model on a scientific dataset [10 points]

Start from the basic data processing pipeline here:
https://colab.research.google.com/drive/1hgIx6dPc2VC5vpP8YrT7PtmhMtyHmjwK

Our project this year is to examine how deep learning can assist with the detection of new exoplanets! You might think that there are so many stars and planets in the universe that it would be easy to find the exoplanets -- it is actually a really difficult task and it took until 1992 for the first planet outside of our solar system to be detected. Since then we have confirmed close to 4,330 exoplanet discoveries. In this project we will focus on the transit method of exoplanet discovery. I encourage you to read some of the background science at these links:

https://exoplanets.nasa.gov/what-is-an-exoplanet/about-exoplanets/
https://en.wikipedia.org/wiki/Methods_of_detecting_exoplanets



This is fundamentally an object tracking problem from computer vision, but the scene is so far away that we can’t see any objects in it directly because the spatial resolution and amount of light reflecting off the planet are too low.

With a high enough resolution imaging system we can get a few pixels of information about the star (which is much larger than the planet and far brighter). If we then take pictures with a fast enough sampling rate (cadence) we can measure the effect of the planet’s silhouette on the shift in brightness of the star. In actual practice we have to account for many issues to successfully confirm an exoplanet discovery: instrument noise, solar flares, drift effects, and unknown physical parameters of the star and exoplanet(s) in its system.

In part 2 you will implement and test an encoder-decoder architecture on sky survey images taken from these two recent and ongoing scientific missions:

- TESS: https://tess.mit.edu/
- Kepler: https://archive.stsci.edu/missions-and-data/kepler

We will access these datasets through a library, lightkurve, which also includes a data processing pipeline for sky survey time series analysis.
https://docs.lightkurve.org/index.html

In the basic starter code linked earlier we can see the general data processing pipeline for detecting an exoplanet:

1. Download a sequence of images of a star over time. These are 11x11 pixels recorded every 2 minutes (cadence) for a target star.
2. Convert each image into a flux measurement for the star, creating a light curve which is a 1-dimension time series of flux measurements. The basic code sums the flux values from the image inside an aperture (mask).
3. Account for flux drift, background noise, and other signal processing / data conditioning.
4. Test and fit parameters to the observed data. The primary parameter is period -- does there exist a periodic darkening in the flux measurement that would indicate an orbiting planet? Other parameters include the size of the planet (ratio to the star), duration of transit, and stellar limb darkening coefficients.
5. Plot and report parameters and guess at the model including overall statistics such as signal to noise ratio (SNR), Combined Differential Photometric Precision (CDPP), and Chi-Square goodness of fit (chi2)

The provided starting code looks for the known exoplanet HAT-P-11-b. You can change this and see if you can find other confirmed exoplanets:
https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PS

Your task for this part is to denoise the sequence of target pixel file images of HAT-P-11-b's star over time using the same network you created in Part 1. You should change the dimensions of your layers to match an 11x11 image. Your new denoising step in the pipeline should go between step #1 and step #2 above.

NOTE: For this part we are not marking for quality of your solution, simply that you have modified your network to correctly work for an 11x11 image as input and are using target pixel files from the real dataset. You can write your training/testing loop however you choose as long as it shows your network doing something. Because the pattern of noise in stellar imaging is mostly between frames it is unlikely that your initial network will improve the result (at best it might learn the identity function; more likely it will smooth over important features such as the transit and create worse SNR/CDPP/chi2 metrics).

Part 3: Refine your the model and report the effects [10 points]

[Note: See https://colab.research.google.com/drive/1qrLpzHZuXYWM8vygovsUJ2krUWy85Q44?usp=sharing for a dataset of 31 targetpixelfile sets and lightcurves for known exoplanets]

In part 3 you need to produce a study of the effects of some architectural refinement to your network. If you completed part 2 successfully you have a basic encoder-decoder network model, a denoised light curve, and different output metrics. Each parameter (and component) of your encoder-decoder network can be changed to generate different output. You should propose some refinement or upgrade to your model and then produce a graph with a short report on the effects of your changes. In general I want you to test architectural refinement -- not manual pre-processing or post-processing of the data.

There is some published research in the area (for example, see https://arxiv.org/pdf/1901.01298.pdf), but it is still a relatively new topic so I will provide some ideas:

- First, you need to choose an evaluation metric to help guide your study. In general we know the following and you can compute them using the given fit_and_report function (although you are welcome to change the metric to something of your choosing):
    - higher SNR is better (although the signal is “hard coded” to be the transit duration and the noise the remaining phases)
    - lower CDPP is better
    - lower chi2 is better
- Second, you will need to refine your existing model / architecture.
    - Your network can work at any stage in the pipeline that you choose. For example, in part 2 your network is between stage 1 and 2 and takes in the original target pixel files and outputs denoised/modified target pixel files. You could convert it to a 1D network and use the original aperture generated lightcurve and TESS pipeline provided lightcurves (your network would then go between stage 2 and 3 or between 3 and 4 in the pipeline). You could change your network to take in target pixel files and the aperture and have it output a light curve (would go between stage 1 and 3 and skip stage 2). You could also have it learn a custom/per frame aperture or weighted aperture to replace stage 2.
    - You can modify anything you’d like to test as your refinement. Some ideas:
        - You can take in additional information (such as the aperture or known star parameters)
        - different configurations / sizes / functions for encoder/decoder network
        - you can increase the channels of data to input a window of 2d images which form a 3d spatial-temporal image (i.e. 100 sequential time points would have a 11x11x100 “video”)
        - you could convert your network into a 1d convolutional network and use the lightcurve generated with the aperture to denoise windows of 1d samples (i.e. 100 sequential flux images would be a 1d lightcurve of length 100 which is kind of like a 1x1x100 input size network -- this would place your network between steps 2 and 3 or steps 3 and 4 instead of steps 1 and 2)
        - custom loss functions
            - different terms in the reconstruction loss, possibly using a physical model as guide - i.e. you can penalize the reconstruction loss

            more for regions of time when the planet transit is known, or you could mask the reconstruction loss only for the known aperture
            - with windowed sequential data you can add a total variation loss
        - extra constraints on latent layers or output
        - variational or generative sampling on the latent layer
- Third, write a short report that includes the following sections:
    - Experimental Design and Methods (1-2 paragraphs)
        - Design - include a simple description of what refinement you made. Make sure to include the value ranges, new / modified layers, changes to functions or other modifications you have made.
        - Hypothesis - describe what effect you expect to see in the result given your changes. Under what conditions do you expect to see a significant difference in the results?
        - Method - describe how you have measured that effect (what is your metric?). Make sure to include some details on the dataset you have used to train and test.
    - Results (1-2 paragraphs)
        - Produce at least one graph that shows the effects of your modifications (for example if you used a 1D autoencoder and SNR and tested different values for window size this would be a graph of 1D autoencoder window size VS SNR)
        - Discuss the trends in your result. Was there a positive or negative or neutral effect from your modifications on your metric?
    - Conclusions (1 paragraph per question)
        - Were you able to answer your hypothesis? Explain how and why (or why not).
        - What implications do your results have for others working on similar problems?
        - What future questions or directions would you take with your project?

[alternate] Part 2/3: Implement/refine a denoising model on a different dataset [20 points]

You are welcome to complete the implementation, refinement, and report for a different imaging or signal processing dataset.

[alternate] Part 2/3: Review a denoising research paper [10 points], and additionally implement, refine, report on their model [20 points]

You are welcome to review a denoising research paper (you can generally follow the report structure in Part 3 and answer the questions from the point of view of the paper authors, “Under what conditions do you expect to see a significant difference in the results?” → “Under what conditions do the authors expect to see a significant difference in the results?”). This would be worth up to 10 points as a substitute for Part 2.

You could additionally obtain the authors implementation code, make some refinement, and report. This would be worth up to 20 points and substitute for Parts 2 and 3.

https://paperswithcode.com/task/denoising
is a reasonable place to start