PyTorch/Matlab Project

- Topics eligible for the projects are:

• Protein simulation, both in oncology and https://en.wikipedia.org/wiki/Bacteriophage;
• Methods for representation of DNA / RNA string for feeding a neural network (either CNN or 1D input networks);
• Creation of artificial images (i.e. data augmentation step) for Lymphoma diagnosis;
• Image filtering to improve Plankton classification;
• Create artificial examples (i.e. data augmentation step) to train a bioacoustics sound classification system (bird classification);
• Development of an activation function for deep network;
• 1D descriptor for representing audio file classification;
• Develop a loss function to segment polyps in colonoscopies;
• Develop a method to encode a RGB image starting from a multi-channel image for the identification of planktic foraminifera;
• Develop multilabel neural network topology to predict drug side effects;
• Develop a semantic segmentation system of 3D point clouds of urban environment;
• Develop an object detection system for Risiko! game;
• Develop a deep learning method to distinguish brown dwarfs (substellar objects that have more mass than the biggest gas giant planets, but less than the least massive main-sequence stars) from objects of other spectral and luminosity classes;
• Combine image and DNAbarcoding data for insect classification;
• Instrumental and environmental noise detection for boosting gravitational waves discovering;
• DNA sequences for identifying viral genomes in human samples.

Python lecture notes: https://mega.nz/file/xBQhyJjJ#xrNlMjss16l3zY9l8QukKtSCDML7DEguv_H1i5JhaWs
Numpy/PyTorch materials are available at https://mega.nz/file/IdZTSaaL#nPogl0TAv1mTO-3DtmNL6OuVe5ObdsXiTIojvxZ1Ans

- Each project can be done by one->three people (no more), if multiple projects are based on the same idea and the code is almost identical they will be invalidated until proven otherwise. If in discussion among you, you have a similar idea and want to test it in different groups, each one will have to proceed with the implementation/test independently.
- Ideas can obviously be taken from the scientific literature, it is important that they are NOT those of the reference papers / tools (see next lecture notes).

- **You may discuss problems with your classmates, but when you write down the solutions, you should do so by yourself. You can use the internet and books for reference material but you should cite every source that you consulted (the name of the book or web page suffices). You should also cite any classmates with whom you discussed solutions.**

- In the next pages each project is detailed, for them baseline Matlab code is already available, for PyTorch project you can read

it as baseline

For any question related to the PyTorch project, please contact the tutor daniel.fusaro@phd.unipd.it.

For any question related to the matlab project, write to me loris.nanni@unipd.it.

For both projects, please send me a single .rar/zip file containing the final paper, the code and the README to use the code.

Further notes

- **PyTorch** project is MANDATORY:

PyTorch, it must be in before exam date

- **Matlab** project is NOT mandatory

The Matlab work can be delivered when you want, even after exam date. In this case, immediately before the exam, you must notify me not to record the grade, but to wait for the delivery of the project (then you will have to register for the next session so that I can record the grade).

You have to submit both code and short paper that describe your work, example of short paper: https://mega.nz/file/gBxGUJhI#-62kPJFJveZnkBUdD1BWI6vu-lWNFUdQK_J1IfIIPV8

- BOTH the projects are evaluated [1-3], this evaluation will be added to the final mark, only if the exam will be > = 18. Also the ease of reading the developed code will be part of the vote (e.g. well commented code) and the formalization of the method in a rigorous way.

- If you submit **both PyTorch and Matlab** projects and the **topics** of these two projects are **different** then you can write two **short papers**. If you want to work in the **same topic** using both **PyTorch and Matlab** you must implement **different ideas** in matlab and PyTorch, moreover you have to **submit a standard scientific paper and not a short one.**
- If you were one of **my students in the AI course**, and you had done the optional project, you can also choose the same topic, but you have to test a different approach than that proposed in the project submitted for the AI course.
- Many **datasets are available as .mat file**, you can read them in Matlab, using python there are several tools for read .mat files, e.g. https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.loadmat.html
- Dataset in the .mat file are often stored in **cell array**, for **python** project you have to **manage them** in your PyTorch environment:

https://stackoverflow.com/questions/40363277/import-matlab-cell-array-into-python-for-scikit-learn

Proteins

- The problem is to develop/implement a method for coding a protein and feeding a neural network, given a sequence of amino acids to obtain its vector or matrix representation of fixed length. Starting from the methods reported in https://www.dropbox.com/s/3webtggi38mxa2r/236717.pdf?dl=0 try to develop a new method (obviously it is not important to create a method that goes better than the existing ones). You can use as classifier one of the approaches explained in the course (e.g. using a matrix representation you can feed a CNN).

There will be two case studies (choose one of the two):

Predicting cancerlectins https://www.dropbox.com/s/upgilflob9in2er/Predicting_cancerlectins_by_the_optimal_g-gap_dipe.pdf?dl=0

Dataset https://www.dropbox.com/s/fgjyr9jorh1zj7d/CancerLectin.mat?dl=0

If you want to use PSSM (https://en.wikipedia.org/wiki/Position_weight_matrix) for feature extraction, the PSSMs are already precomputed

https://www.dropbox.com/s/zd08x1yz37qfjj1/PSSMtraining_Lectin.mat?dl=0

https://www.dropbox.com/s/b58q2on9tlfk08e/PSSMtest_Lectin.mat?dl=0

b) Identification of Bacterial Cell Wall Lyases

https://www.dropbox.com/s/8dj67eu1gjid94n/1654623.pdf?dl=0

Dataset https://www.dropbox.com/s/myv5edk6jdblsrq/Phage.mat?dl=0

Main project https://www.dropbox.com/s/eypmndv6tkfflrt/MainPhage.m?dl=0

If you want to use PSSM arrays to extract features, the PSSMs are already precomputed

https://www.dropbox.com/s/g850mnv6jg724sc/PSSMtraining_Phage.mat?dl=0

https://www.dropbox.com/s/cjjmpyzxvvsxwml/PSSMtest_Phage.mat?dl=0

Proteins

- in the .mat file the proteins are saved in a cell array https://it.mathworks.com/help/matlab/cell-arrays.html
- with Protein = Training {prot}; extract the sequence of the prot-th protein and save it in the string 'Protein'
- therefore each cell is a different protein (Training {1} is the first protein; Training {2} the second and so on)
- each protein has its own length (different proteins -> different lengths)

DNA

- The problem is to develop/ implement a method of coding a DNA / RNA sequence, given a sequence of nitrogenous bases to obtain its vector/matrix representation of fixed length.
- Application problem:

Recombination spot identification
https://www.dropbox.com/s/zr94ta51t9eu8nb/srep23934.pdf?dl=0

- Dataset https://www.dropbox.com/s/80qwn7gf19vu0ak/Recomb.mat?dl=0

- Useful paper to get ideas on how to represent DNA in vector format:

https://www.dropbox.com/s/ixbsjo69kpiegx2/DNA.rar?dl=0

Lymphoma – Data augmentation

- A classic problem in the classification of medical images is the small number of examples for setting the parameters of the classification method. The goal is to implement an algorithm to create "artificial" images starting from the original ones, examples of methods normally used for this purpose will be provided.

- Application: “LYMPHOMA dataset. LY includes 375 images of malignant lymphoma subdivided in three classes: CLL (chronic lymphocytic leukemia), FL (follicular lymphoma), and MCL (mantle cell lymphoma). " Download: https://www.dropbox.com/s/elfn1jd63k94mlr/DatasColor_29.mat?dl=0

- I suggest to read:

https://www.dropbox.com/s/6k0421kh9xz1kma/DataAugmentation.pdf?dl=0 section 3

https://www.dropbox.com/s/v4rvwclk5wmkar2/PCAdataAugmentation.pdf?dl=0 section 2.A

https://www.dropbox.com/s/0yj8oawdhzf9w6w/4%29SAR%20Target%20Recognition.pdf?dl=0 section 3

https://www.dropbox.com/s/vkq5xl2hpm2vm4c/3%29ImageNet%20Classification%20with%20Deep%20Convolutional.pdf?dl=0 section 4.1

https://www.dropbox.com/s/onc8dt5dil6oq48/General%20Purpose%20%28GenP%29%20bioimage%20ensemble%20of%20Handcrafted%20and%20Learned%20Features%20with%20Data%20Augmentation.docx?dl=0

- Example for training a CNN in matlab: (it requires AlexNet download https://en.mathworks.com/help/deeplearning/ref/alexnet.html, matlab should guide you directly when you run the code) https://www.dropbox.com/s/9emmggrwz1a5ys9/ SimpleAlexNetDataAugmentation.m? Dl = 0

See the comments of the .m file for understanding where to insert the function that creates the additional images

- Other option, change way to use discrete cosine transform to create new images, starting code (matlab) https://www.dropbox.com/s/22lrx84oww48y5t/Code_DCT.rar?dl=0 read pages 10-12 of https://www.dropbox.com/s/3y56thgneqvjr5s/1904.08084.pdf?dl=0

Plankton – Image preprocessing

- Filtering images can be useful for extracting information otherwise difficult to extract https://www.dropbox.com/s/gph8d9yuzbpizeg/PlanktonPreProcessing.pdf?dl=0

The problem is to test methods to filter plankton images in order to improve the classification (see section 3.1 of the previous link).

- https://www.mdpi.com/1424-8220/20/9/2592/pdf

it can be useful as an additional starting point

- Starting code (Matlab) https://www.dropbox.com/s/s40xj985e01ni4w/SempliceAlexNet.m?dl=0

See the comments of the .m file for understanding where to insert the function that applies filter to the images

- Dataset: https://www.dropbox.com/s/gdm8n8lqxn6v7iq/Datas_44.mat?dl=0   The data are stored in a cell array, see https://www.dropbox.com/s/s40xj985e01ni4w/SempliceAlexNet.m?dl=0 to understand how the data are stored.

ZooScan is a dataset of 3771 images collected from the Bay of Villefranche-sur-mer using the Zooscan teconlogy (G. Gorsky, M.D. Ohman, M. Picheral, S. Gasparini, L. Stemmann, J.B. Romagnan, et al., Digital zooplankton image analysis using the ZooScan integrated system, J. Plankton Res. 32 (2010) 285–303. doi:10.1093/plankt/fbp124) . The images belong to 20 categories with different number of samples for each category (from 28 to 427 samples per class). Most categories are zooplankton, other species of Medusae, and eggs of zooplankton; the remaining categories are non-zooplankton and images with bad focus.

Audio classification – Data augmentation

- It involves creating artificial examples, starting from spectrograms, to improve deep network performance for audio traps (bird classification).
- In the .rar file https://www.dropbox.com/s/7vwb4h8c9etyl52/EsampleAudio.rar?dl=0 there are both examples of data augmentation and related code, from which to take inspiration
- The reference database is https://www.dropbox.com/s/53mkeraqqa0p6nb/DatasBIRD_1_1.mat?dl=0
- The main code from which to start (MATLAB) is https://www.dropbox.com/s/8y5kkcyt90ouf5i/SempliceAlexNetDataAugmentatio nAUDIO.m?dl=0 , you just need to add the functions to create the additional poses

Activation function

Modify / create activation function to train deep neural network, in addition to the lecture notes of the course, as a further starting point you could read the following two papers:

https://www.dropbox.com/s/1p59yo8qszot76h/1905.02473.pdf?dl=0

https://www.dropbox.com/s/i70oq4xl7voct3c/GaLU.docx?dl=0

The code of the methods detailed in the above links is available: https://www.dropbox.com/s/jfjoisowp5v0c26/FunAct.rar?dl=0; starting from this you can test variants that you will invent.

Dataset to validate performance:

"LYMPHOMA dataset. LY includes 375 images of malignant lymphoma subdivided in three classes: CLL (chronic lymphocytic leukemia), FL (follicular lymphoma), and MCL (mantle cell lymphoma). " Download link: https://www.dropbox.com/s/elfn1jd63k94mlr/DatasColor_29.mat?dl=0

Baseline Matlab code https://www.dropbox.com/s/jfjoisowp5v0c26/FunAct.rar?dl=0:

SimpleExample.m is the file that launches the tests (it is the main).

Add code to ChangeLayerAllLayers.m to invoke your function (see below line 22)

Implement the activation function (take a cue from those already implemented, see the two folders learnableActivations and StandardActivation)

PyTorch: you can couple your developed activation function with one of the available networks (you can choose which one).

For both Matlab and PyTorch project you have to compare at least the performance of your approach with that obtained by ReLu.

1D descriptor for Audio Classification

- dataset: https://www.dropbox.com/s/c7pad0eniwkrwav/ESC-50.rar?dl=0

for details on the dataset, read:

https://github.com/karolpiczak/ESC-50

The aim is to improve the 1D descriptor proposed in https://www.dropbox.com/s/08twfgnhugvbk1d/SpiralPooling.pdf?dl=0

baseline code:
https://www.dropbox.com/s/x42aycnmxpegpez/Spiral.rar?dl=0

Then the developed 1D descriptor is used for feeding a neural network (you can use any approach)

Spiral performances reported in the paper are incorrect due to their error in the testing protocol. 45% real performance, always good to be hand crafted, I reached 51% using data augmentation (both for training and test set).

Loss Function - Segmentation

- To implement a new loss function for a segmentation network
- Application: Polyp detection in colonscopy https://arxiv.org/abs/2112.12955
- Training set: https://zenodo.org/record/5834916#.YfcOp-rMI2w polyp dataset-> only the training set without data augmentation
- Test sets: https://zenodo.org/record/5579392#.YdzC1f7MKUk
- Matlab baseline code: https://mega.nz/file/xUp3TCZC#yqo9UxuK7gt8hY3zaVp1gct3iV_T_X8myWNEIsiUlW0

trainSegmentationPreTrained.m main file to run the tests

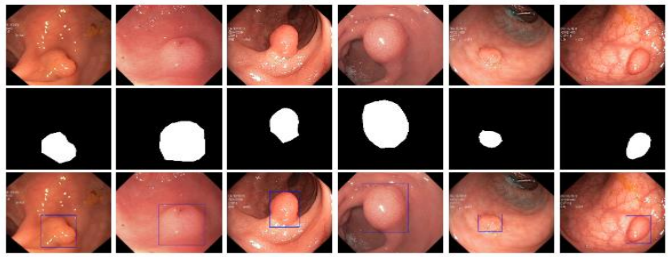
designDeepLabV3plus.m file to update for replacing the loss, see rows 22-40

the folder \NewLoss stores some examples of loss functions (https://arxiv.org/abs/2112.12955)

Be careful, the data have to be managed as dlarray (inside loss function), in this way matlab calculates automatically the gradient

PyTorch: modify loss of the following network https://github.com/james128333/HarDNet-MSEG

Moreover, see https://mega.nz/file/8EhU1LYR#jyhfwQPuJHEpRN84KpcE5qaEwHrwspYJfPkUXDngOYA for a further help for modifyng the loss of HardNet



Encode information from n grayscale images to a single "RGB" image (for training a CNN)

- To implement new way to encode information of a set of 16 grey level images in a single three channels image, then to train/test a CNN using the obtained images
- Application: identification of planktic foraminifera using convolutional neural networks. Picking foraminifera from sediment samples is an essential, but repetitive and low-reward task that is well-suited for automation. The first step toward building a picking robot is the development of an automated identification system. https://par.nsf.gov/servlets/purl/10107559
- Notice that:

the starting illumination angle for the sixteen images seems to change partially for different classes of foraminifera in the naming scheme used to sort the pictures. We believed the start angle could lead to biased results in classification if a specific class always had the same starting angle compared to the others. This problem was addressed, when pre-processing the images, by randomly sorting them while keeping the relative illumination angles ordered to avoid the insertion of bias within methods that leverage the light positional information.

- Dataset set (run a 4-fold cross validation): https://mega.nz/file/NE5zzSzQ#PxXg4Y2yG-NbQaWNEs8xizRXcqrqIZZ6GLSvF2gZD9Q
- Matlab baseline code https://mega.nz/file/IVozhBCR#pP4AYvFyaPcpv0m6k1XB9BEMIzuvKJuYq8v3PlxsOdA

training.m to train/test alexnet

percIMGProcessing.m to create the RGB images as proposed in the paper

Multilabel classification

- To implement new topologies based on GRU/LSTM/TCN for multilabel classification, baseline approaches https://arxiv.org/abs/2110.04414

https://www.scirp.org/pdf/ns_2020051314251266.pdf

- Application: Predicting drug side effects is an important topic in the drug discovery. Although several machine learning methods have been proposed to predict side effects, there is still space for improvements. The side effect prediction is a multi-label learning task, and we can adopt the multi-label learning techniques for it. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-015-0774-y
- Dataset: https://mega.nz/file/IF4xTQgT#OqxxCAdKPFLhww3HI5yLLf6ku0Benl34LOHS87HzYf8

```
load('Liu_dataset')
clear X
X{1}=Enzymes;
X{2}=Pathways;
X{3}=Targets;
X{4}=Transporters;
X{5}=Treatment;
X{6}=chemical;
atc_fea=[X{1} X{2} X{3} X{4} X{5} X{6}];
atcClass=side_effect;
```

otherwise, you can download the .mat that stores several multilabel datasets https://mega.nz/file/gBgTFCKD#04VtDIautDKUho2txNATurTI886h_D1V6gC-1Yt2jWY
, the following matlab code uses this .mat

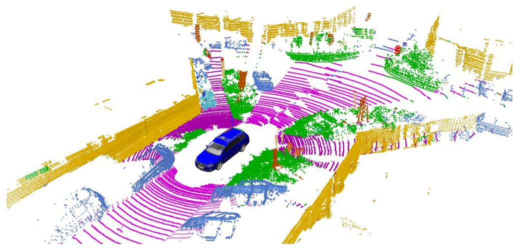
- Matlab baseline code: https://mega.nz/file/ZR5TXY7I#Gzb_8-ySf-QZjerqzj_GMfm2eJ_Lfm4NiXTISG0tw1Q

TestsCascataMultiLabel.m main for running the tests

GRU_TCN and GRU_LSTM are two examples of multilabel classifiers already coded to be coupled with TestsCascataMultiLabel.m

The folder EsempiModificheTopologia contains other example of topologies (also if not suited to be coupled with TestsCascataMultiLabel.m)

Semantic Segmentation of Urban Environment



- The goal is to develop a semantic segmentation system of 3D point clouds of urban environment.
- There are plenty of methods already available. Common techniques are:
  1. Process directly the 3D point cloud as a list of 3D points
  2. Pre-process the 3D point cloud projecting it to a different space, e.g. using spherical projection
  3. Pre-process the 3D point cloud voxelizing it in order to reduce the computational effort then back-propagate the segmentation to the original cloud
- Start from existing methods and develop a new one (based on one method among the three above)
  - obviously, it is not important to create a method that goes better than the existing ones
  - **put effort on the Neural Network aspects**, like model architecture, loss function, activation functions
  - Try different strategies and report the resulting performance in tables (mIOU, TP, FP, mAP, … see papers)
- Suggested datasets (both require a free account for the download) :
  - **SemanticKitti** (link to the dataset) http://www.semantic-kitti.org/dataset.html
  - **NuScenes** (link to the dataset - Use the mini version (visible after login)) https://www.nuscenes.org/nuscenes
- Suggested existing approaches are:

1) **PointNet** (link to the paper https://arxiv.org/abs/1612.00593) (link to the github implementation of both PointNet and PointNet2 (*aka PointNet*++) https://github.com/yanx27/Pointnet_Pointnet2_pytorch)
   - You will process the full point cloud, hence you will have a "heavy" model, definitely taking more than 100ms (the usual maximum runtime)
   - At this link (https://mega.nz/file/LapjHLID#-HvMOQRHB-0QKaFe_WDn8hWQCIcGYieDgxHF9rNoLzk) you can find a Jupyter Notebook: use it as a baseline for implementing the project
   - The notebook contains also the instruction for downloading a sample dataset, already pre-processed for the task
   - you can use it as it *is* (a subset extracted from SemanticKitti) or work with a full dataset directly
2) **RangeNet** (link to the paper https://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/milioto2019iros.pdf)(link to the github implementation https://github.com/PRBonn/lidar-bonnetal/tree/master/train/tasks/semantic)
   - You will implement a 2D CNN to process the so-called 2D range image obtained from the 3D point cloud
   - Suggestions:
     - use this parser.py (https://github.com/PRBonn/lidar-bonnetal/blob/master/train/tasks/semantic/dataset/kitti/parser.py) to load the data: the spherical projection (aka *range image*) is built on the fly
     - In the Github repo, scratch some super useful scripts to compute the statistics
3) **Again PointNet,** but processing a **voxelized** input point cloud (link to open3d voxelization tutorial http://www.open3d.org/docs/latest/tutorial/Advanced/voxelization.html)
   - But you'll handle a lighter model. You will also implement the back-projection method (start from a simple rule: all points in the labeled voxel get the same label)
   - Suggestion: use the Jupyter Notebook at point 1

Risiko Army Detector: Synthetic2Real

- The goal is to develop an **object detection** system for Risiko! game (https://en.wikipedia.org/wiki/RisiKo!)
  As you may know, in Risiko! game there are two (usually) types of pieces: **tanks and flags**.
- For this project, you need to design and implement a method for detecting those two pieces in the YOLO style, i.e., localizing them with a bounding box.
- Let's define with army either a tank or a flag. The problem will consist in the **detection of 12 different classes** (6 tanks, 6 flags, each in the different color of blue, red, yellow, purple, black and green).
- Since the collection of a real dataset is prohibitive in terms of *time spent in labelling*, you're given a Jupyter Notebook for producing a **synthetic dataset** that will automatically produce and label some data.
- By iterating, you can obtain a full synthetic dataset! Then, you just train on it instead of using big, expensive, real datasets.

So, requirements of this projects are:

1) Produce the synthetic dataset using the Notebook script (if you want, you can modify it!) and split it in Training/Validation/Test (you're given also an example of synthetic dataset. Take a look at it, and if you want, integrate it in your synthetic dataset)
2) Split the real images dataset you are provided in Training/Test (70/30 rule)
3) Design a suitable Deep Learning architecture
4) Train the model (first time: only using synthetic training set)
5) Test the model on the test split of the synthetic dataset: report the performance
6) Test the model on the real images that you are provided: report the performance
7) Re-do the steps 4-6 by introducing the real images training set. How does the performance change?

For comparison, you can find a **YOLOv7 model** (scripts are taken from this repo https://github.com/WongKinYiu/yolov7) trained on this task. Use the notebook Risiko! Test.ipynb and report the performance of this model on your test set (both synthetic and real) in your project report.

Try to think how to integrate this system in a fully autonomous player of Risiko! **You'll never lose a match again!**

You can download the project folder containing all the referred files from here
https://mega.nz/file/XXZg3DKL#SOvWrs1kBXf8NgIQNAO4vp5YCq3sXHO-75yCQIMrsDg)

Brown dwarfs detection

- According to various estimates, brown dwarfs (BD) should account for up to 25 percent of all objects in the Galaxy. However, few of them are discovered and well-studied, both individually and as a population. Homogeneous and complete samples of brown dwarfs are needed for these kinds of studies. Due to their weakness, spectral studies of brown dwarfs are rather laborious. For this reason, creating a significant reliable sample of brown dwarfs, confirmed by spectroscopic observations, seems unattainable at the moment. Numerous attempts have been made to search for and create a set of brown dwarfs using their colours as a decision rule applied to a vast amount of survey data, for details see https://arxiv.org/pdf/2308.03045.pdf
- The goal is to propose a deep learning method to distinguish brown dwarfs from objects of other spectral and luminosity classes.
- As performance indicator you have to use the Matthews correlation coefficient (MCC) (explained in the above paper). For calculating using it in Matlab: https://github.com/preethamam/MultiClassMetrics-ConfusionMatrix

- You can donwload the data here: https://mega.nz/file/YYZXwJjS#w1OpCr9eNQPnY3EaLcuKKZojN66gijGpdxImdhb261A
- data() stores the features (each row is a pattern), we have 5 fold (fold=1:5)

Training set: data(idTR{fold},:)

Test Set: data(idTE{fold},:)

Label Training Set: labelTR{fold}

Label test set: labelTE{fold}

It is a binary classification problem: brown dwarfs vs objects of other spectral and luminosity classes

- The data is provided before the augmentation and imputing the missing values. I.e. a value of 0 in Training/Test set means that the given features is missing and you have to impute it (see the above paper for description of data imputation), so you have to implement (or propose a new one), not the one used in the above paper, a data imputation method and a neural network to distinguish brown dwarfs from objects of other spectral and luminosity classes.
- The original dataset is here https://github.com/iamaleksandra/ml-brown-dwarfs so you can check the name of each feature

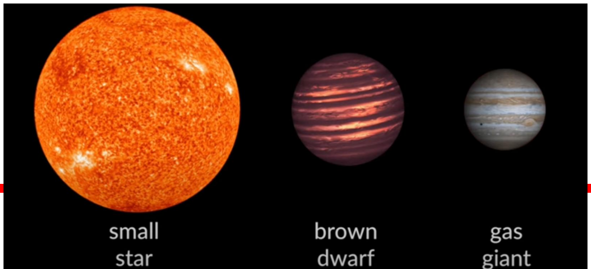


Image and DNA data for insect classification

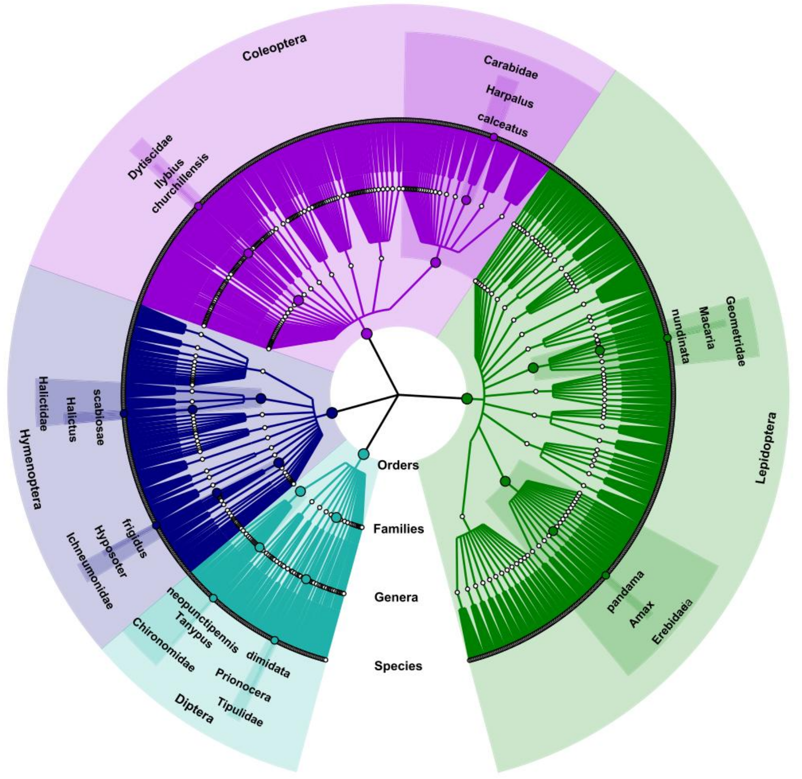
- Understanding biodiversity for insects requires both discovery and identification. Insects are one of the largest and most diverse animal groups on the planet with an estimated 5.5 million species, yet only 20% are described and many are disappearing faster than they can be identified, making it difficult to assess biodiversity.
- This study used paired insect image and DNA sequence data obtained from the Barcode of Life Data System (BOLD), from four major Insecta orders: Diptera, Coleoptera, Lepidoptera and Hymenoptera. Each insect data point contained a 658 bp DNA barcode sequence (cytochrome oxidase subunit I— COI); an image; and additional information such as country of origin, life- stage, order, family, subfamily, genus and species names.
- for details see paper available at: https://mega.nz/file/AcomRYpD#jAs71xX0j-H7LxDihKsW4hOYwG0WxDRkCravh2gYjjE
- Data used in this study is available at IUPUI DataWorks

https://doi.org/10.7912/D2/27

from the above link you download both the images/barcoding sequences and a dataset with the features already extracted as in the paper (see the structure data.mat and splits.mat), you can choose how to approach the problem: i.e. to use the set of extracted features or to feed your method with images/barcoding sequences.

For reading data.mat/splits.mat see

https://mega.nz/file/UVYH1TYA#XBhKVsXh_-3wWPCM_PR71xKVAsLpaxm__LnHYkueBGA



you can solve the problem as you like, for example you can create two separate networks, one for images and one for DNAbarcoding sequences and then to combine the output, or propose a single topology with both information (i.e., images and DNAbarcoding) as input

Detection of gravitational waves

- Gravitational waves are waves of the intensity of gravity that are generated by the accelerated masses of binary stars and other motions of gravitating masses, and propagate as waves outward from their source at the speed of light. https://en.wikipedia.org/wiki/Gravitational_wave
- The detection of gravitational waves with ground-based laser-interferometric detectors requires sensitivity to changes in distance much smaller than the diameter of atomic nuclei. Though sophisticated machinery and techniques have been developed over the past few decades to isolate such instruments from non-astrophysical noise, the detectors are still susceptible to instrumental and environmental noise transients known as "glitches," which hinder searches for transient gravitational waves.
- The goal is to proposed a deep learning glitch classification system.

for details see: https://mega.nz/file/8U5EEQaK#qvGWeLUsPrxFCJXaqz5g-kJVv6zCpTp3EKLCnrojjHM

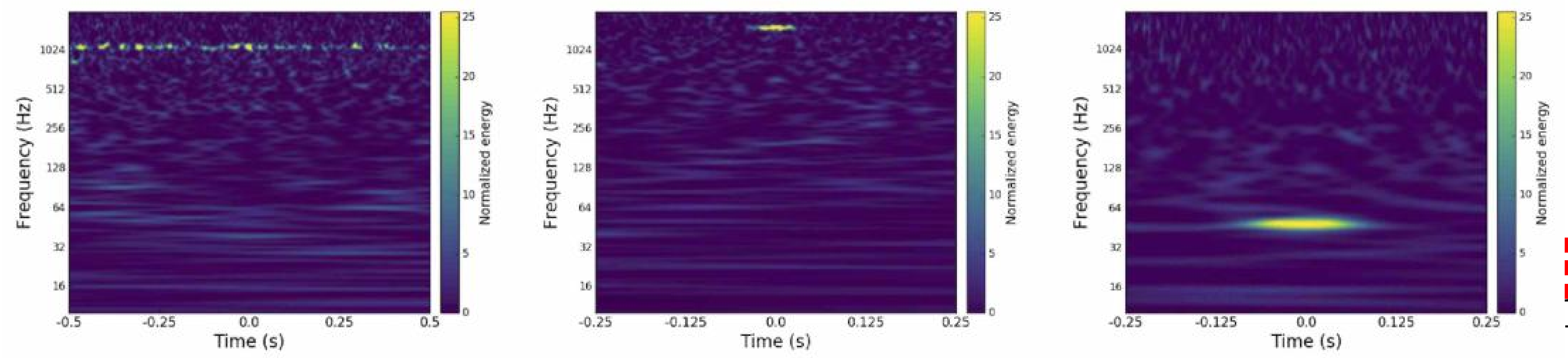
the datasets are availabled at: https://mega.nz/file/RAY2xDCC#Ky0WeV72GsHujEsWQ_w6AsS20-kKPFU9yNW9dLAdzA0

there four files, each glitch is visualized using four different time windows (± 0.25, 0.5, 1.0, and 2.0 seconds),

each of the four datasets is related to a given time window (e.g. DatasGravity1.mat is related to ± 0.25, DatasGravity4.mat is related to ± 2).

In each dataset the data is stored as in https://www.dropbox.com/s/s40xj985e01ni4w/SempliceAlexNet.m?dl=0 (you have to change the name of the loaded dataset).

You can solve the problem as you like, for example you can create four separate networks (one for each time window), and then to combine the four output, or propose a single topology where all the four time window images are used as input.



DNA sequences for identifying viral genomes in human samples

- The human virome is the collection of all viruses that reside in and on the human body. Many different viruses are present in human samples and their composition appears to be different in diseased individuals. The detection of potential viral genomes in human biospecimens is usually performed by NCBI BLAST, which implements alignment-based classification where sequences are aligned to known genomes from public databases and then estimates how much similarity they share. However, metagenomic samples might contain a large number of highly divergent viruses that have no homologs at all among known genomes.

for details https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0222271

- The aim of this project is to proposed a neural network based approach to identify potential viral sequences across different DNA human samples.

- All necessary data files are available from the GitHub repository: https://github.com/NeuroCSUT/ViraMiner.

- Use the following file of the previous cited GitHub repository:

ViraMiner/data/DNA_data/fullset_test

ViraMiner/data/DNA_data/fullset_train

ViraMiner/data/DNA_data/fullset_validation

the label is the last value of each row