Dimensionality Reduction for Studying Diffuse Circumgalactic Medium[1]

Mentors -

Jeremy Bailin, Jacob Morgan, Sergei Gleyzer, Jianghao Huyan, Varsha Kulkarni, Shravan Chaudhari.

Name : Venkata Pavan Kumar Turlapati

Mail Address : pk1842000@gmail.com | vt4873@srmist.edu.in

Contact : +91-9545077346 | +91-8788247794

GitHub Handle : /TheSkrill

LinkedIN : /Pavan-Kumar

Location (City, Country and/or Time Zone): Chennai, India, UTC +5:30

Proposal Idea :

Title : An AutoEncoder based GAN approach for dimensionality reduction in Quasar Absorption Spectra Datasets.

To implement machine learning-based dimensionality reduction models applicable to the quasar absorption spectra datasets.

Synopsis :

The primary objective of dimensionality reduction is to compress data while preserving most of the meaningful information. The aim is to uncover the "hidden variables" that can successfully expose the underlying structure of the data. This proposal elaborates on 4 techniques - Random Forest, PCA, AutoEncoders and a novel GAN architecture which have the potential to successfully reduce the dimensionality of the quasar absorption spectra datasets. The Random Forest model is a feature selection type reduction model, PCA is a components based reduction model, while AutoEncoders and novel GAN is a generative type model. The novel

[1] https://ml4sci.org/gsoc/2021/proposal_CGM.html

GAN network is based on the idea of replacing generators (normal neural networks) with an autoencoder architecture. This idea would enable us to capture the probabilities learnt by the autoencoder and the regular neural network. I have hand-picked each architecture from varied fields to find the best and the tailored match for the quasar absorption spectra dataset.

Proposal Goals :

To test the compatibility of the quasar absorption spectra dataset upon :

1) 3 types of Principal Component Analysis (PCA)
    a) Robust-Kernel PCA
    b) Sparse PCA
2) Random Forest with other feature selection methods.
3) AutoEncoders
4) A novel architecture of a Generative Adversarial Network.
5) CLI and Python Binding for all the above methods.

Why did I choose only these methods?

There are a myriad of dimensionality reduction techniques in the machine learning community. Each dataset has certain specific dimensionality reduction techniques which are tailored for it.

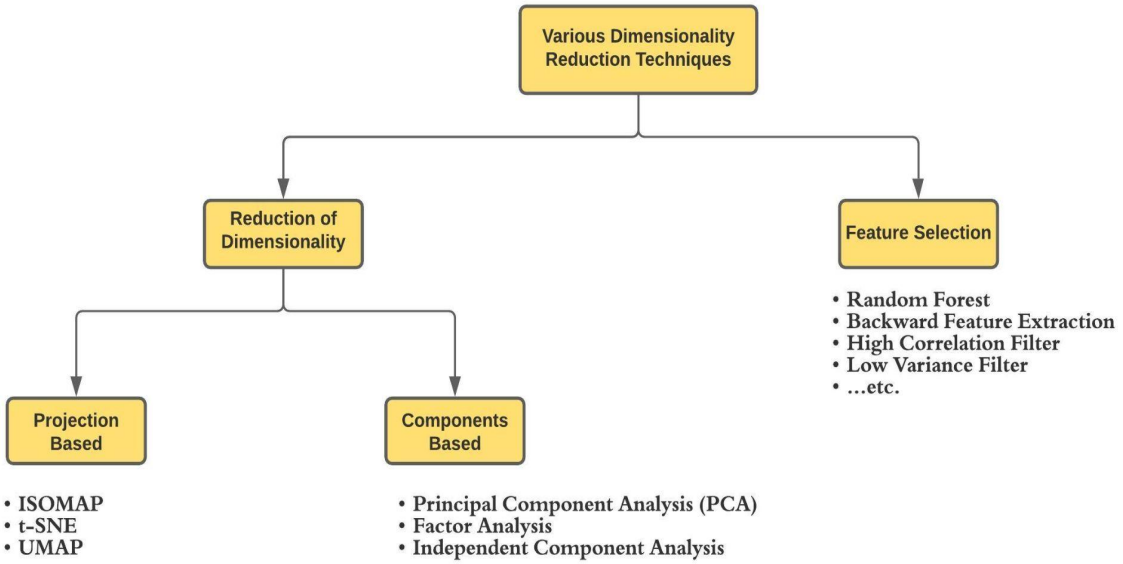

Fig. 1 : Types of Dimensionality Reduction Techniques

After going through 20+ research papers on dimensionality reduction and Quasar absorption spectra, I chose to choose one method from each subsection in Fig.1 to ensure that we can find a perfect dimensionality reduction algorithm.

1. **PCA (and its 2 versions) :**
    - I chose to include 2 variants of this algorithm because the traditional PCA is extremely sensitive to outliers. To overcome these limitations I chose to include the ***Robust-Kernel PCA.***
    - I chose ***Sparse PCA*** to be the second variant of PCA because it penalizes the Euclidean length (L2 norm) so that it does not get too large. Sparse PCA also has a scope of upgradation. If our data is high-dimensional, Euclidean distance may fail to perform. Then I have shortlisted Manhattan, Mahalanobis or Bhattacharya distance as a replacement. Clubbing them with Sparse PCA has a massive potential to boost their performance.

2. **Random Forest and other feature selection methods:**
    - ***Random Forest*** is the most robust algorithm with the capability to handle large data sets with higher dimensionality. It also outputs the

feature importance which can come in handy. Works pretty well with imbalance datasets and missing labels and also extends to unlabeled data, where PCA falls short.

- All the other feature selection methods such as high correlation filter, low variance are fantastic methods to get an eagle's eye view of the data and filter it accordingly.

3. **AutoEncoders :**
    - With highly complex data the autoencoder has a better chance of unpacking the structure and storing it in the hidden nodes by finding hidden features. In contrast to PCA, it can model complex nonlinear functions. PCA features are maybe uncorrelated because their features are on an orthogonal basis. Autoencoders surpass this limitation.
    - The only tradeoff is that AutoEncoder can prove to be computationally expensive.
4. **A novel Generative Adversarial Network (GAN) :**
    - Using GANs for dimensionality reduction is a fairly unexplored area, with research papers coming up only in the past 1-2 years. I included GANs as a dimensionality reduction technique mainly because of their uncanny ability to converge quickly upon the solution and because of my extensive research experience with it (refer my CV).
    - They would also help me add a novel architecture to the research of dimensionality reduction.

Reasons for eliminating the following dimensionality reduction techniques :

1. **t-SNE :** When there are more than 2-3 dimensions, it gets stuck on local optima like gradient descent.

2. **Iso-mapping :** Performance dropped drastically after testing on the data provided during the evaluation test. Slightly wrong parameters and the performance got hit badly. Did not show any result after tuning hyperparameters.

3. **LLE :** It was too sensitive and unstable for testing out on high-dimensional data.

While completing the evaluation tasks, I had the chance to gauge the properties of the dataset I would be working on.

CLI and Python Binding :

For a developer, algorithm design is easier than API design. But from a user point-of-view, it is not feasible for him/her to replicate our results from raw code. Therefore, API design and documentation is a pivotal part of algorithm design.

I would create a CLI and Python Binding for each algorithm with properly documented code, for every one of the 4 algorithms I am going to experiment upon.

The sample CLI code would look like :

```
$ python dim_red_main.py  -- algo str1  --b_size num1  --l_r num2  --b1 num3
--n_hl num4  --g_h_l1 num5  --d_h_l1 num6
```

```
--algo   : The algorithm we would be using (AutoEncoders, PCA, GAN, etc.)
--b_size : Batch size
--l_r    : Learning rate
--b1     : Beta-1 hyperparameter
--n_hl   : Number of hidden layers
--g_h_l1 : No of hidden units in the first hidden layer of generator
--d_h_l1 : No of hidden units in the first hidden layer of discriminator
```

The Python binding code would look like :

```
DRA = DIM_RED(algorithm=str1, batch_size=num1, learning_rate=num2,
beta_1=num3, n_hl=num4, g_h_l1=num5, d_h_l1=num6)
```

Related Works

I included this section specially to highlight the 2 papers which inspired me to come up with the novel Generative adversarial network's idea.
The first paper is titled VAE-SNE (variational autoencoder stochastic neighbor embedding) [Link]. This work can be used for both dimensionality reduction and clustering. VAE-SNE simultaneously compresses high-dimensional data and automatically learns a distribution of clusters within the data — without the need to manually select the number of clusters. Being a mix of VAE and SNE algorithms it is able to capture the global scenario of every dimension in the data. VAE-SNE can be used to detect outliers when embedding out-of-sample data.

The second paper being Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction [Link]. Here, the intrinsic dimensionality of the data set is estimated by the maximum likelihood estimation method. Fisher Score and Information Gain based feature selection are used as multi-strategy methods to remove irrelevant features after which PCA is applied on selected clusters for dimensionality reduction.

Implementation :

PCA:

```
[ ]  1  from sklearn.decomposition import PCA
     2
     3  # scikit-learn choose the minimum number of principal components such that 95% of the variance is retained.
     4  pca = PCA(0.95)
     5  pca.fit(X_train)
     6  print(pca.n_components_)
     7  X_train = pca.transform(X_train)
     8  X_test = pca.transform(X_test)
     9
    10  print("\n\nX_train shape : "+str(X_train.shape))
    11  print("\n\nX_test shape : "+str(X_test.shape))

    13


    X_train shape : (400000, 13)


    X_test shape : (100000, 13)
```

Fig. 2 : Implementation of a basic PCA which I had done in the evaluation test (Task 3)

```
def fit(self, tol=None, max_iter=1000, iter_print=100):
    iter = 0
    err = np.Inf
    Sk = self.S
    Yk = self.Y
    Lk = np.zeros(self.D.shape)

    if tol:
        _tol = tol
    else:
        _tol = 1E-7 * self.frobenius_norm(self.D)

    #this loop implements the principal component pursuit (PCP) algorithm
    #located in the table on page 29 of https://arxiv.org/pdf/0912.3599.pdf
    while (err > _tol) and iter < max_iter:
        Lk = self.svd_threshold(
            self.D - Sk + self.mu_inv * Yk, self.mu_inv)                    #this line implements step 3
        Sk = self.shrink(
            self.D - Lk + (self.mu_inv * Yk), self.mu_inv * self.lmbda)     #this line implements step 4
        Yk = Yk + self.mu * (self.D - Lk - Sk)                              #this line implements step 5
        err = self.frobenius_norm(self.D - Lk - Sk)
        iter += 1
        if (iter % iter_print) == 0 or iter == 1 or iter > max_iter or err <= _tol:
            print('iteration: {0}, error: {1}'.format(iter, err))

    self.L = Lk
    self.S = Sk
    return Lk, Sk
```

Fig. 3 : Sample implementation of Robust PCA (https://github.com/dganguli/robust-pca)

In Figure 3 Robust PCA is implemented. I would take inspiration from this architecture, and develop the Robust Kernel PCA.

Sparse PCA algorithm has been implemented in scikit-learn [Link]. I would explore the source code and try to add compatibility for the Manhattan and Mahalanobis distance.

Random Forest (RF):

```
rfc = RandomForestClassifier()
rfc.fit(X_train_scaled_pca, y_train)
```

Also, I would do extensive hyper-parameter tuning for the RF model using RandomSearchCV and GridSearchCV.

RandomSearchCV:

```
param_dist =
 {'n_estimators': n_estimators,
 'max_features': max_features,
'max_depth': max_depth,
'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf,
 'bootstrap': bootstrap}
```

```
rs = RandomizedSearchCV(rfc_2, param_dist, n_iter = 100, cv = 3, verbose = 1, n_jobs=-1, random_state=0)
```

GridSearchCV :

```
param_grid = {'n_estimators': n_estimators,
'max_features': max_features,
'max_depth': max_depth,
'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf,
'bootstrap': bootstrap}
```

```
gs = GridSearchCV(rfc_2, param_grid, cv = 3, verbose = 1, n_jobs=-1)
gs.fit(X_train_scaled_pca, y_train)
rfc_3 = gs.best_estimator_
```

The advantage of using the RandomForest model is its useful functionality : FEATURE IMPORTANCE (aka Gini Importance)

```
importance = model.feature_importances_
#Gives a score of every feature denoting its importance.
```

```
# summarize feature importance
for i,v in enumerate(importance):
        print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
```

```
pyplot.show()
```

AutoEncoders:

```
encoding_dim = 40

input_df = Input(shape=(171,))
encoded = Dense(encoding_dim, activation='relu')(input_df)
decoded = Dense(171, activation='sigmoid')(encoded)

# encoder
autoencoder = Model(input_df, decoded)

# intermediate result
encoder = Model(input_df, encoded)

autoencoder.compile(optimizer='adadelta', loss='mean_squared_error')

autoencoder.fit(X_train, X_train,
                epochs=150,
                batch_size=256,
                shuffle=True,
                validation_data=(X_val, X_val))
```

This is a sample of a very simple AutoEncoder network. I would design an autoencoder architecture that is tailored to the quasar spectra dataset to enable maximum reduction in dimensionality without compromising upon the performance.

A novel Generative Adversarial Network (GAN) :

I have been working and innovating in the field of GANs since the past 2 years. I love modifying and tweaking their architecture to see the magical results they produce. I believe that a GAN is the simplest architecture which has the potential to resemble a human brain due to its simplistic yet efficient nature. I have co-authored 2 research papers and have done 1 internship where-in I delved deep into their architectures and modified them to produce varied results.

While searching about dimensionality reduction, I noticed not many papers had utilized the power of GANs in dimensionality reduction. I saw GSoC as a perfect opportunity for me to do the same!

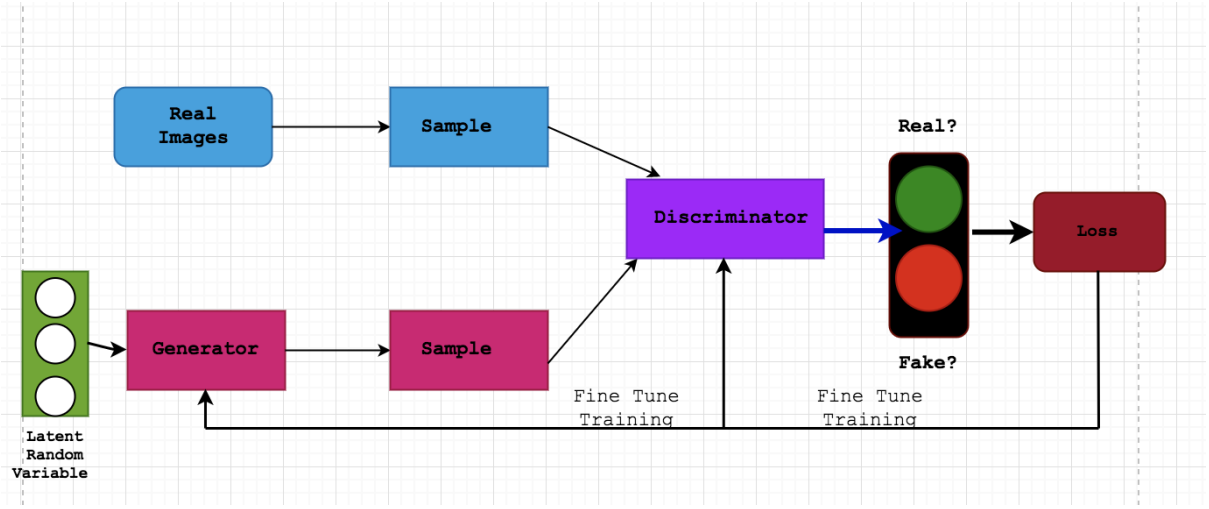

Fig. 4 : The Basic Structure of GAN.

There are 2 neural networks : The Generator and the Discriminator competing against each other in a minimax game. The generator produces fake samples and mixes those with the original data. The discriminator tries to distinguish between the fake and the real samples.

I had the idea of **replacing generators (normal neural networks) with an autoencoder architecture.** This idea would enable us to capture the probabilities learnt by the autoencoder and the regular neural network. This may turn out to be novel and efficient.

- ➔ Additionally, I would like to use my *Progressive Step Training* method which I came up with in my internship at Defence Research and Development Organization (DRDO).
- ➔ Plus, I would also test this architecture with the Manhattan and Mahalanobis distance I am using in Sparse PCA method.

Note : ROC Curve evaluations, Recall and F1 Scores will be provided for each method (PCA, Random Forest, etc.) to gauge which is better.

All in all, our final architecture may look like this,

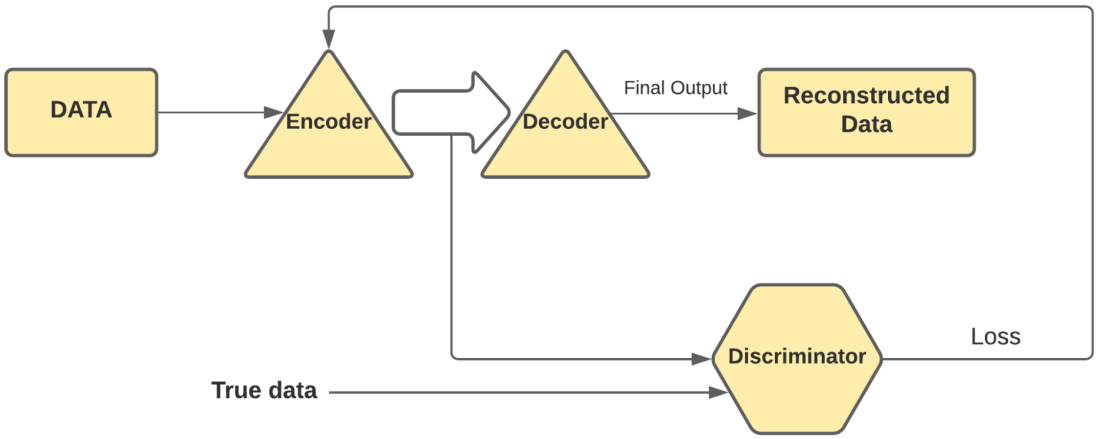

Fig. 4 : Proposed architecture of the Encoder-GAN.

Timeline :

- ➔ **May 17 - June 7**
  - Getting introduced to the community and bonding.
  - Studying the quasar absorption spectra data in detail from a physicist's perspective.
  - Study in depth about Autoencoders, initial preparation.
- ➔ **June 7 - June 22**
  - Implementing Random Forest and other minor feature selection methods.
  - Testing them rigorously on various hyperparameters.
  - Initiate the design of GAN, for reducing the workload in Week 5.
  - Writing the documentation, CLI and Python Binding and an example on how to use it.
- ➔ **June 23 - July 8**
  - Implementing the 2 variations of PCA.
  - Testing their performance on Manhattan and Mahalanobis distance.

- Writing the documentation, CLI and Python Binding and an example on how to use it.

➔ **July 9 - July 23**

- Implementation of AutoEncoders.
- Testing their performance.
- Writing the documentation, CLI and Python Binding and an example on how to use it.

➔ **July 24 - Aug 8**

- Implementation of the novel GAN network.
- Testing its performance on Inception Score, etc.
- Writing the documentation, CLI and Python Binding and an example on how to use it.

➔ **Aug 8 - Aug 16**

- Cleaning up the code, bug fixes and finalizing the documentation.
- Cross-checking for any missed out components.

Stretch Goals :

In case, due to any unexpected errors, implementation is going slow, I would use the time of Aug 8 - Aug 16 as a buffer time to complete the project. Being extremely excited about the opportunity, I am already improving on the performances of my model in Evaluation Test - Task 3 and am fine-tuning the architecture of the proposed model of GAN (Fig. 4). Since the only major work would be while designing the GAN, I am working on keeping the basic model of the architecture ready before May 17th to ease my process. Therefore, I would probably complete a major part of coding before time.

Being ardently involved and dedicated to research, I am certain I would complete the project on time. There are some chances I may be left with some time, where I would like to apply *independent component analysis* on the dataset. It could not find a place in the proposal goals because of its faded usage over the years. It holds a reputation of performing well on signal processing

datasets, and hence I believe it has the calibre to show some good results on the quasar absorption spectra dataset too!

About me :

Hello there! I am Venkata Pavan Kumar Turlapati. *(shortened as Pavan)*
Thanks a million for reading my proposal!
I am a final year student, pursuing B.Tech (CSE) from SRM Institute of Science and Technology, Chennai, India. I hail from Hyderabad, Telangana.

I have been programming for the past 8 years. I fell in love with Machine Learning based research way back in my first year, when my professor challenged me to find a way to increase the accuracy of ML models in high-dimensional imbalanced datasets. After pondering upon the problem for a year, me and my professor designed an algorithm - named *Outlier-SMOTE* - which led us to publish our work in Intelligence Based Medicine - Elsevier Journal ! Plus, we applied our algorithm to the COVID-19 dataset, and it surprisingly out-performed the traditional algorithms!

After publishing my first research paper, I wanted to delve deeper into the field. Therefore, I co-authored 2 more research papers which are currently under review in Springer Nature - Computer Science Journal during the lockdown. I have extensively studied GANs in this process, where I have improved the training process in my internship in Defence Research and Development Organization(DRDO), and used them in my major project to achieve improved results in the image inpainting domain. The major project work has been accepted in IEEE International Conference on Emerging Trends in Industry 4.0 (2021 ETI 4.0), organized by the OP Jindal University. Currently, I am interning with Indian Institute of Hyderabad (IIT-H) where I am improving the autonomous driving algorithm - Trajectron++.

I cherish studying mathematics and computer science. I am proficient in Python, Java, C++ and Flutter. I have recently started doing open-source contributions and have completed the Hacktoberfest challenge. I am fluent to work on Linux, macOS and Windows. Given a chance to work on this project, I assure dedication of at least

40-45 hours per week to the work and that I do not have any other obligations during the period of the program. I am applying to only one organization, i.e., ML4SCI for GSoC-2021. Also, I would love to continue to work on this project even after GSoC, where-in I would contribute to the ML4SCI community. My long term goal is to pursue a PhD focused on theoretical machine learning, and to inculcate *artificial general intelligence* in machines, where systems would be able to take conscious decisions based on intuition, rather than just data.

Why am I the perfect fit for this project?

1. Considering the prerequisites, I have been practicing to code on Python and C++ from my 8th grade and I have been doing research in ML/DL since the past 3 years.
2. While writing my 1st and 3rd research paper, I had studied data-processing and dimensionality reduction in depth, due to which, I have decent experience handling huge datasets.
3. I am fluent with version control systems, like GIT and SVN which will help a lot to keep track of our progress in designing algorithms.
4. I have the experience of using GPUs if needed. I have worked on AWS Sagemaker, Google Cloud GPUs and Colab-PRO GPUs. If the project training is taking more time than needed, I can easily shift the code into one of the GPUs and get the outputs exponentially faster.
5. Being a research-oriented person, I would work towards successfully implementing and improving the novel GAN architecture by the end of GSoC period so we could try to submit our work to a reputable journal.
6. I am well versed with the concepts of AutoEncoders, GANs and PCA as I completed various courses on them and I am fluent with all the basics. I would straight-away start implementing them after the community bonding period.

LINKS :

MY CV Evaluation Task 1 and 2 Evaluation Task 3 Link to trained models