Sparse classification and reconstruction methods on non-traditional datasets

Group members: Daksh Sinha, Sree Prasanna Rajagopal

1. Introduction

Optimal sparse sensing is the process of finding a reduced number of optimal data points to capture a majority of the information. By selecting the most important features using techniques like Principal Component Analysis (PCA), it is possible to represent the data in a low-rank basis. This basis is then used to extract the optimal sensors using the QR pivoting algorithm. This is possible because of prior knowledge about the patterns in the data [1].

The aim of our project is to apply these techniques to non-numerical datasets:

- Image Datasets
    - Emojis
    - MNIST, with masked images
- Categorical Datasets
    - Cancer dataset
    - FIFA dataset

We then compare the performance of the sparse sensing algorithm with the traditional machine learning approaches which work well on these specific datasets. We also try to figure out scenarios where sparse sensing outperforms these approaches, by modifying test data. We use the PySensors package for sparse sensor reconstruction and classification, and the scikit-learn library and the keras framework for implementing conventional machine learning algorithms.

We had the following questions:

- How well do sparse classification methods work on non-physics datasets?
- How well do sparse reconstruction methods work when compared to GANs?
- How well do sparse classification methods work on mixed datasets containing categorical and numerical data compared to NNs?
- Are there any situations where sparse classification outperforms NNs?

2. Literature Review

We mainly took inspiration from [1] and [2] for the mathematical foundations of the optimal sensor placement algorithm for reconstruction and classification. Optimal sensor placement is an important decision in various fields, from control theory to biological

processes. For most moderate-sized search spaces, optimal sensor placement was achieved through a combination of expert knowledge and intuition, and optimal experiment design, Bayesian criteria etc.

However, for arbitrarily large problems, choosing globally optimal placements is an intractable computation, in which the number of possible placements grows combinatorially with the number of candidates. By using modern techniques in machine learning and sparse sampling, optimal sensor placement can be solved for these large search-spaces.

$$\mathbf{y} = \mathbf{C}\mathbf{x} = \mathbf{C}\Psi\mathbf{s} = \Theta\mathbf{s}$$

$$\mathbf{s} = \mathrm{argmin}_{\mathbf{s}'} \|\mathbf{s}'\|_0$$

We use the PySensors package for optimal sensor placement, defined in [3]. PySensors has defined methods for Sparse Sensor Placement Optimization for Reconstruction (SSPOR) and Sparse Sensor Placement Optimization for Classification (SSPOC). It also has a basis submodule for defining the basis for these algorithms. The basis can be either of the following:

- Identity : Uses input data directly, without any modification
- SVD : Uses only the first n POD modes, where n is passed as a parameter
- RandomProjection : Multiplies data with random Gaussian vectors. Used mainly for compressed sensing.

PySensors is very convenient to use as it's built with scikit-learn compatibility and uses the commonly used fit() and predict() methods that are associated with most scikit-learn models. The SSPOC() and SSPOR() methods both take in number of sensors and basis modes as arguments, making cross-validation for selection of these parameters straightforward.

For reconstruction, PySensors uses the SSPOR method. It works in the following way:

- Fits a basis object, decided by the user, to the data (Identity by default).
- Implements the QR Pivoting algorithm for sparse sensor selection

For classification, PySensors used the SSPOC method. It works in the following way:

- Fits the basis object to the data (Identity by default).
- Fits a linear classifier to the data (Linear Discriminant Analysis (LDA) by default)
- Uses orthogonal matching pursuit implementation for binary classification and multi-task Lasso for multi-class implementation.

Both of these methods return a list of the first n optimal sensors, where n is also a hyperparameter decided by the user. The learned sensors depend on the basis mode used, and other hyperparameters.

3. Implementation and Results

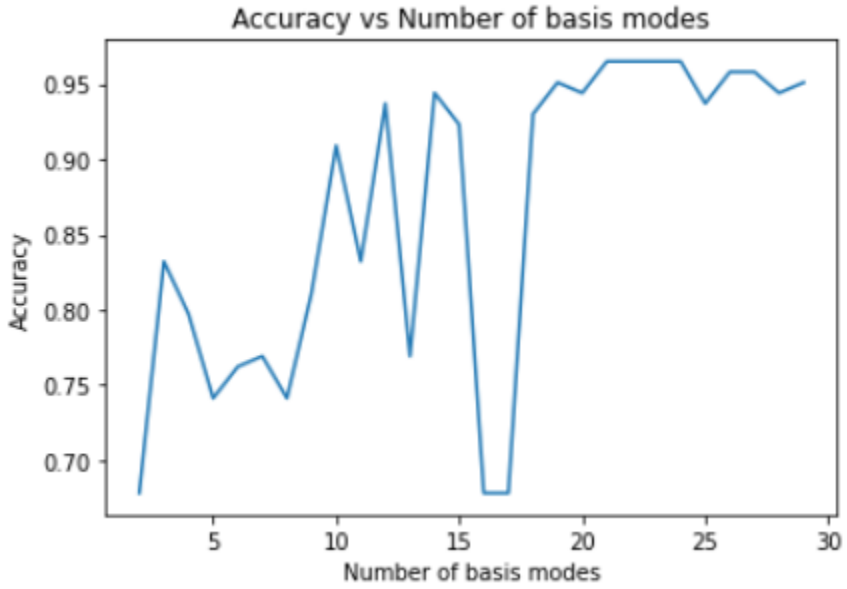
We compare the accuracy of the SSPOR and SSPOC algorithms with traditional methods that are known to perform well on our chosen datasets. We perform classification on the Cancer dataset, FIFA 20 player dataset, MNIST dataset (with masked images) and reconstruction on the emoji dataset.

a. Cancer Dataset

The cancer dataset contains 10 real-valued features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. Each sample is classified as either benign or malignant. This is a pretty well explored dataset, with multiple models achieving accuracy of over 99%. All the data has been cleaned and there are no categorical features, so no preprocessing was required. The goal is to classify a cell nucleus as malignant or benign based on the features provided.

i. SSPOC

For this dataset, we explored the effect of varying the number of basis modes on the accuracy of predictions. We also compared the maximum accuracy achieved by the SSPOC model with a simple neural network.



ii. NN

We used a 3-layer dense NN with 100 dimensions in the hidden layer. The resulting accuracy comparisons are in the following table:

Architecture	Specifics	Accuracy
Sparse sensor placement optimization for classification (SSPOC)	19 POD modes	94%
Dense NN	3 layers with 100 hidden dimensions	95%

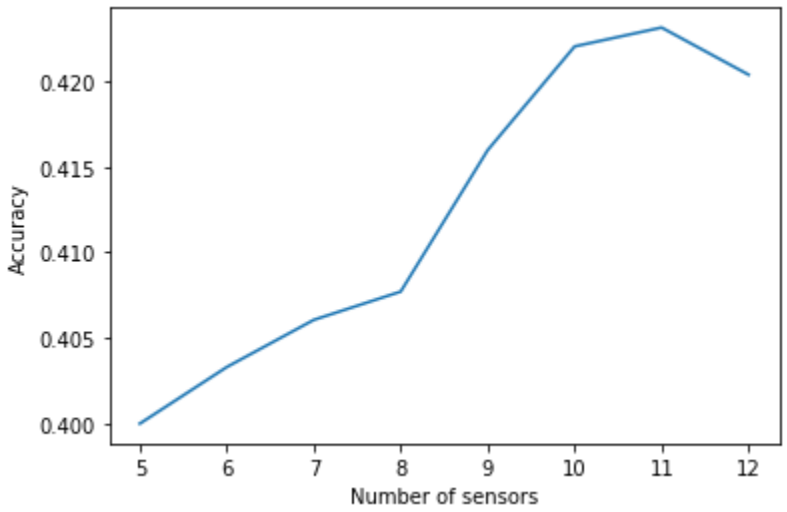
The model performs well, with accuracy approaching the neural network accuracy when a larger number of POD modes are used. This could be due to the fact that the dataset is very simple, with most of the features exhibiting a linear relationship with the class labels.
Increasing the number of POD modes led to higher accuracies, as was expected. We then moved on to more complex datasets, starting with the FIFA 20 Player dataset.

b. FIFA 20 Player Dataset

The FIFA 20 Player Dataset contains a list of all the players in EA Sports' FIFA 20 and their respective stats, like shooting_accuracy, passing_accuracy, etc. The goal is to predict the position a player plays in based off of his stats. We use a total of 37 features, out of which 3 are categorical. The categorical features were label encoded for the SSPOC classifier. There were a total of 27 classes, corresponding to different positions on the football pitch.
We evaluated the model on 25% of the dataset and compared accuracies with a RandomForestClassifier and a dense neural network.
For the RandomForestClassifier, we perform a grid search over a number of values of the parameters 'n_estimators' and 'max_depth'. We also used a 4-layer fully connected neural network implemented using the keras framework.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1000)              38000

 dense_1 (Dense)             (None, 100)               100100

 dense_2 (Dense)             (None, 50)                5050

 dense_3 (Dense)             (None, 27)                1377

=================================================================
Total params: 144,527
Trainable params: 144,527
Non-trainable params: 0
```

For this dataset, we determined the effect of the number of sensors on the accuracy. accuracy generally increases with the number of sensors.

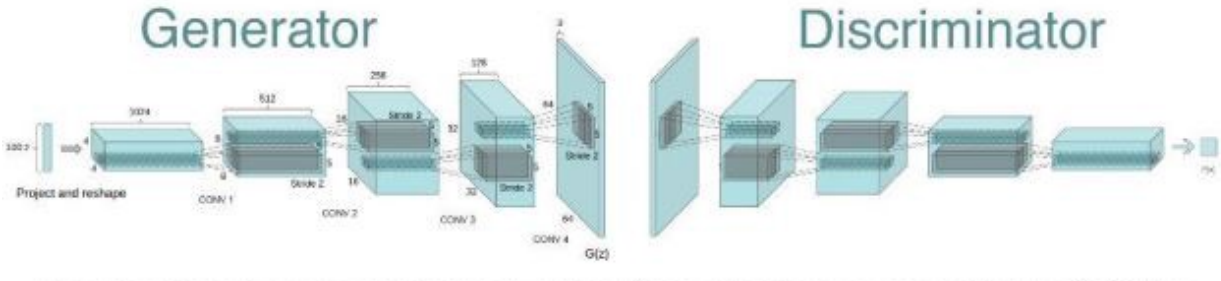


A 42% accuracy is impressive for a linear classifier, considering this is a multi-class classification with 27 classes. For comparison, a random classifier which just predicts the most frequent class gives 8% accuracy.
It still paled in comparison to the neural network, which achieved an accuracy of 70%. Since the SSPOC method uses an LDA classifier, we tried mapping the input data into higher dimensional spaces with the help of Polynomial and RBF kernels. A marginal increase in accuracy was observed, with the RBF kernel giving an accuracy of 46%. SSPOC still performed poorly compared to neural networks. We started thinking about the limitations of neural networks and if SSPOC can be used to address these problems. For example, neural networks are heavily dependent on finding relationships between features from training data and do not perform well when there is a lot of noise in the test

data. SSPOC on the other hand tries to reduce the number of measurements, and tries to accurately perform classification using this smaller subset of measurements. We implemented this hypothesis on the MNIST dataset.

c. Reconstruction on emojis dataset

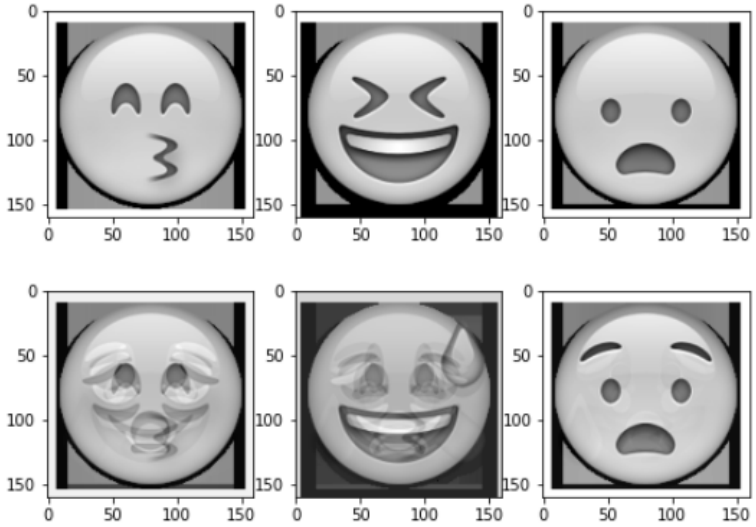
For reconstruction, we use a simple dataset containing 19 pictures of popular emojis. We compare the SSPOR method with GANs, which are considered the state-of-the-art for reconstruction tasks. Considering this dataset contains simple PNG images with very few differences between samples, we expect the SSPOR algorithm to perform as well as GANs.



These images are ideal for sparse reconstruction as the difference between most of them is often just a few pixels, as can be seen in the image below.

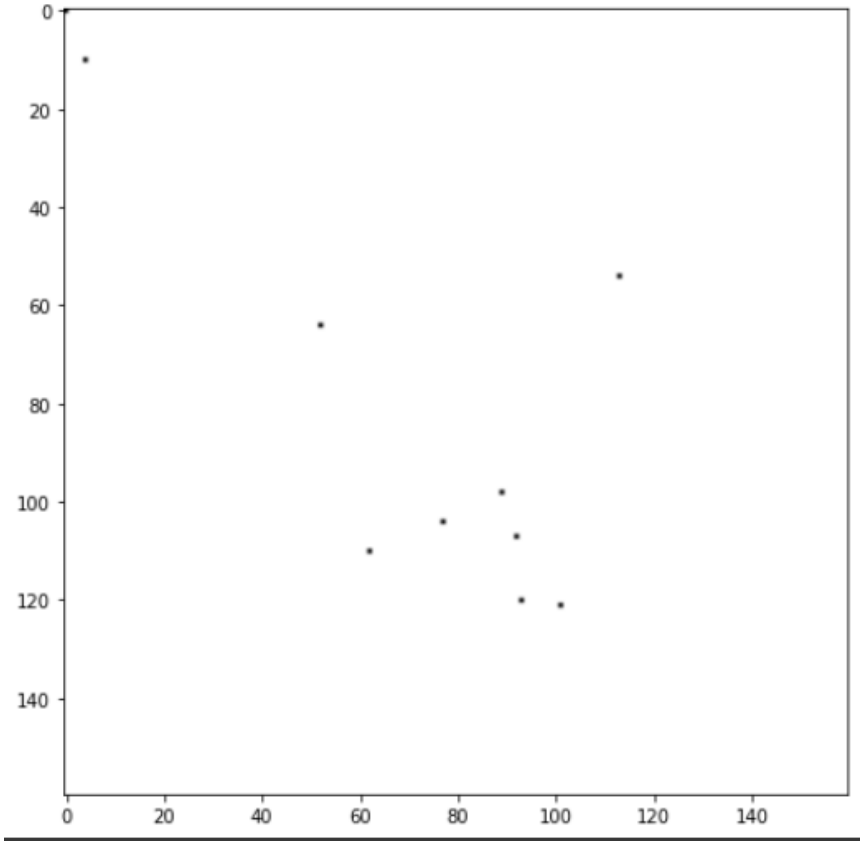
Hence, ideally SSPOR would place sensors at the few pixels where these images differ and high reconstruction accuracy can be achieved with very few sensors.

GAN generated image



SSPOR generated images. Original ones in the top row.

The above results are obtained using only 10 sensors for a 160x160 image size. It can be seen from the reconstructed images that even with such a low number of sensors, the algorithm is able to accurately generate some parts of the image, like the eyes and the mouth in the third image. We plotted the chosen sensors by the algorithm in the same image space:



As expected, most of the sensors are concentrated in the center of the image where the most of the important distinguishing features of the emojis are present.
Thus, sparse reconstruction compares well with neural networks in this case and was much more cheap computationally.

d. MNIST Dataset

The MNIST dataset is a well known dataset containing 28x28 images of handwritten digits. We trained the SSPOC classifier and a 4 layer convolutional neural network (CNN) with max pooling on this data, and then added noise to the test data before evaluation.

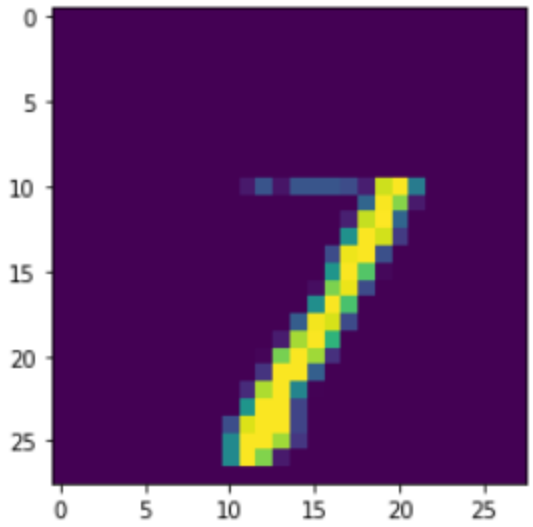
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dropout (Dropout)           (None, 1600)              0

 dense (Dense)               (None, 10)                16010

=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

CNN model summary

We used the following methods to obfuscate the images:

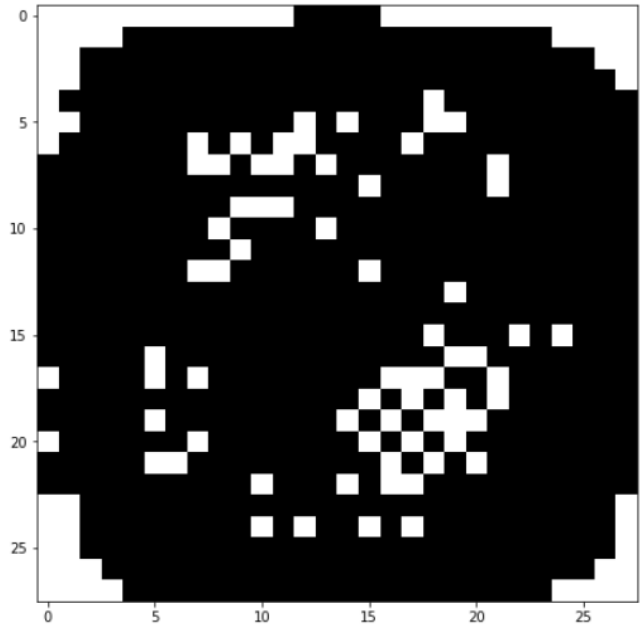
- Masking the lower/upper half of the image
- Changing pixel values for 50% randomly selected pixels

For the original test set without any addition of noise, the SSPOC classifier is comfortably beat the neural network, which achieves accuracies greater than 97%. SSPOC is only able to achieve accuracies ~82% when all sensors are used.



Example image of the digit 7 with top half masked

However, the obfuscated test data yields surprising results. SSPOC consistently outperforms the neural network when more than >50% of the image pixels are disturbed. When the number of sensors lies in an optimal range, which was discovered to be 600-700, the SSPOC achieves accuracies in the range of 40-43%, while the best accuracy the neural network achieves is around 35%.
Our intuition is that since the SSPOC always makes predictions based on the most important sensors, it is more robust to unseen noise. The neural network learns the features from the ‘denoised’ data and is unable to handle these unseen perturbations in the test set.



It is yet to be seen how general this result is. A lot of factors need to be considered, for example, other noise patterns, comparison with deeper NNs, comparison on other datasets, etc.

4. Conclusions

In this project, we compared the performance of sparse reconstruction and classification models on multiple non-traditional datasets. We started with simple datasets, the cancer dataset for classification and the emoji dataset for reconstruction. It was found that for these simple datasets, the sparse sensing models perform as well as neural networks while using only a fraction of the features to make predictions. We explored the effect of the number of basis modes and the number of sensors on accuracy.
We then moved to a more complex dataset, the FIFA 20 players dataset, that contains both categorical and numerical features. We compared the sparse classifier's model accuracy with a RandomForestClassifier and a dense neural network. It was found that, even after multiple manipulations of the data such as casting to a higher dimensional space, the sparse classifier was comfortably beat by the neural network.
We then explored scenarios when sparse sensing could outperform neural networks. We performed these experiments on the MNIST dataset. When a significant amount of noise was added to the test data, SSPOC was able to beat neural networks when both models were trained on the same unperturbed data. Thus, the SSPOC was found to be more robust to noise, which intuitively also makes sense as it used only a few random sensors to make predictions.
Hence, while neural networks still beat sparse sensing algorithms in most cases, their lack of interpretability, susceptibility to noise and computational cost still leaves multiple scenarios where they can be replaced by sparse sensing algorithms.

5. Future work

Our main goal with this project was to rediscover the common notions about NNs, sparse sensing, and which computational methods are suitable for what datasets. We are curious to see how far we can push SSPOC performance with kernelization methods.
We are also interested in identifying more image related tasks which involve train-test disparities for which SSPOC may be more suited. Some common data augmentation transforms such as random contrast, random brightness, patched color inversions may in fact be better handled by SSPOC than standard NNs.

6. Bibliography

1. Krithika Manohar, Bingni W. Brunton, J. Nathan Kutz, Steven L. Brunton, “Data-Driven Sparse Sensor Placement for Reconstruction”
2. B. W. Brunton, S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Sparse sensor placement optimization for classification,” SIAM Journal on Applied Mathematics, vol. 76, no. 5, pp. 2099–2122, 2016
3. Brian M. de Silva , Krithika Manohar , Emily Clark , Bingni W. Brunton , Steven L. Brunton , J. Nathan Kutz, “PySensors: A Python Package for Sparse Sensor Placement”
4. R.Semaan, “Optimal sensor placement using machine learning”
5. Keigo Yamada, Yuji Saito, Koki Nankai, Taku Nonomura, Keisuke Asai, Daisuke Tsubakino, “Fast greedy optimization of sensor selection in measurement with correlated noise”
6. Kaggle FIFA 20 Complete Player Dataset https://www.kaggle.com/datasets/stefanoleone992/fifa-20-complete-player-dataset
7. Goewey, Gerrit. “Generating Emoji with GANs part 1: Starting with the basics.” *Datamensional*, 23 June 2021, https://www.datamensional.com/generating-emoji-with-gans-part-1-starting-with-the-basics/.
8. Kaggle. “Breast Cancer Wisconsin (Diagnostic) Data Set.” *Kaggle*, https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
9. Kaggle. “Full Emoji Image Dataset.”

10. Group Contributions

Daksh: Cancer dataset (sparse classification), Emoji dataset (sparse reconstruction), FIFA dataset (random forests and sparse classification), MNIST (cnn and sparse sensing), Project report and presentation

Prasanna: Cancer dataset (neural network), Emoji dataset (GAN), FIFA dataset (Kernelization and neural network), MNIST(test data perturbation), Project report and presentation