The main use of an undercomplete autoencoder is not in its ability to perfectly reconstruct the input, but in the useful features it is capable of distilling.

In our case, we wanted to see what properties the autoencoder model could extract when given time series air pollution data for various cities throughout the United States. Using data from OpenWeather's Air Pollution API, we made 28,338 requests from U.S. state city’s coordinates or U.S. territories city’s coordinates for all air pollutant information from November 27, 2020 0600 GMT-4 to June 8, 2021 0500 GMT-4 in a hourly basis. This gave us a total of 26100 successful request. We divided the request in 6 different parts creating 6 different files, to minimize overall run time and could request only the files needed incase of error. We then created 8 new data frames for each pollutant, with the daily average of each, and combined form the 6 API call files. We then removed any row that had any missing values and removed any cities with the same name and pollutant daily averages since we did know which was the original city. While this made the data quickly useable, a more ideal solution would be to add a state/country code to distinguish between cities and to impute any missing values using k nearest values. For more on this see the future improvements section. Our final 8 data frames left us with 18526 rows and 195 columns. In the columns there was city, lat, lon, and 192 days.

We compared the autoencoder model to Principle Component Analysis, a much older technique used for dimensionality reduction that has been a part of statistical literature since the early twentieth century.[2] The main distinction between autoencoders and PCA is that autoencoders can span a nonlinear subspace, whereas PCA learns the principle subspace.

Autoencoders that employ nonlinear encoder and decoder functions can create a more powerful generalization than PCA. However, an autoencoder model that is allowed too much capacity is at risk of reconstructing the input "too perfectly" without extracting any useful information.

For PCA we set the y values as 2021/06/06. We also normalized the data frame values and did a 5 split K-fold cross validation. Then we ran a linear regression PCA for each pollutant.

For auto encoder we set the y values as 2021/03/28. We also normalized the data frame values. We ran a grid search first looking only for the best activation model. We ran 15 different dimensions, which 8 were between 2 and 25 dimensions, with different activations and the rest of the parameters the same. We then choose the activation that appeared more often as the best activation mode for each dimension threw the grid search. This was Leaky ReLU. We did a new grid search with Leaky ReLU and we changed threw the parameters of learning rate, batch and epoch. We once again choose the parameters that appeared more often as the best. The final parameters were learning rate at 0.001, batch size at 64, and epochs at 150. We then ran the auto encoder for each pollutant.