Applications of Autoencoder asset pricing models to a highly dimensional cross-section

Marco Molinari
Research Fellow of Qi4M LAB
molinari@qi4m.com

September 16, 2022

Abstract

We test Autoencoder asset pricing models, Kelly, Gu, and Xiu (KGX, 2019)[13], with a dataset that is smaller than the one they used by two orders of magnitude and has higher dimensionality; specifically, the new dataset has 123 variables as opposed to the original dataset, which has 94. It's also more geographically diverse: it comes from Qi4M and thus includes EMEA-based securities vs. us-based the Center for Research on Security Prices (CRSP) for the original. Lastly, we fit the model on both the original and the Qi4M dataset, and we probe the solidity of the model's performance when confronted with challenging data by comparing their respective R^2s. We check the degree to which the increase in the number of predictive characteristics impacts the model's tendency to overfit the training dataset.

1 Introduction

The literature initially tackled the stochastic discount factor through multi-factor asset pricing models such as the original Fama, and French (1993)[4] three-factor model. These models failed to satisfactorily explain the variability within the cross-section of returns, and authors thus added new factors to their models, such is the case with Fama and French (2015)[5] expanding their model to include five factors. This factor overload has plagued traditional asset pricing models, even going as far as destroying the predictability that such factors held, McLean and Pontiff (2015)[10].

In order to tackle the factor overload, the literature has introduced methods aimed at reducing the dimensionality of the data while avoiding losing information. Contributions to the literature in this direction have been operated by Chamberlain and Rotschild (1983)[6], and Connor and Korajcyzk (1988)[7], who have employed Principal component analysis to reduce dimensionality and allow for the linear combination of factors and betas into returns. Later, Kelly, Pruitt, and Su (2018)[1] introduced Instrumented principal component analysis to account for the latent factors previous models mistook for alpha.

IPCA and PCA rely on an unwarranted linearity assumption in their computation of betas. KXG have overcome the linearity assumption with autoencoders: neural networks that allow for computing characteristics into covariates non-linearly. Autoencoders have shortcomings; namely, they suffer from a high propensity to over-fit the training data. It has been shown that large enough deep neural networks can even memorize the entire training dataset and perfectly fit random data, Zang, Bengio, Hardt, Recht, Vinyals (2015)[2].

In order to address these shortcomings, a vast array of regularization techniques have been proposed by the literature, such as normalizing the data, penalty functions like LASSO, Tibshirani (1996)[11], and batch normalization Ioffe, Szegedy (2015)[8], which improve generalization and smooths the optimization landscape of a neural network accelerating training in the process, Santurkar, Tsipras, Ilyas, Madry [12].

We set out to stress the model's capacity to perform dimensionality reduction by presenting it with a dataset that is smaller than the original dataset by two orders of magnitude and presents more variables, including some macroeconomic indicators. This paper is intended for consumption both by practitioners new to the literature, as it focuses on the implementation of the model, and by more experienced readers looking for validation of the model's performance in harsher conditions.

2 Methodology

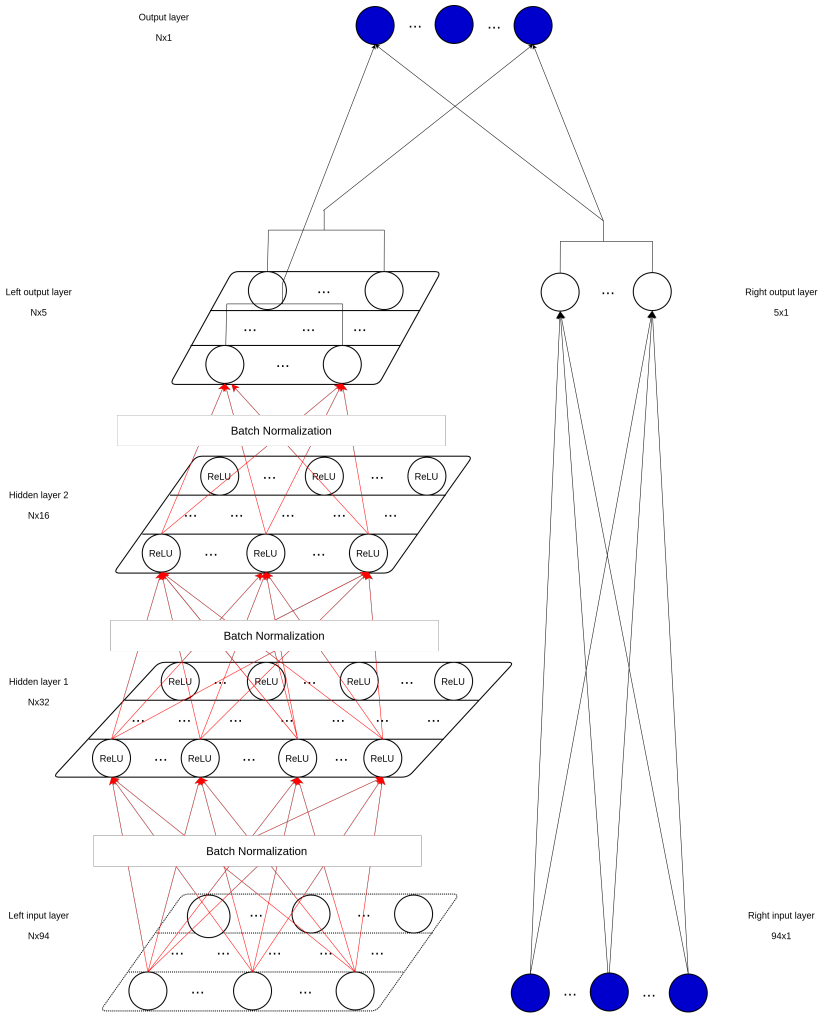
In this section, we describe our implementation of the model from Kelly, Xiu, and Gu (KXG, 2019)[13] as an Autoencoder expanded to incorporate covariates. We then elaborate on the implementation of the regularization techniques KXG[13] employ to prevent overfitting. We describe the optimization algorithms we employ in each case. Lastly, we mention the training, validation, and testing processes.

2.1 The Autoencoder Models

The general structure of the Autoencoder is comprised of two branches. The first is a shallow Autoencoder that takes P asset-level predictive characteristics as inputs, and it combines them non-linearly into K factors, with $K < P$; this is the case because KXG[13] chose to maintain the economic interpretation of factors and thus limited the portfolios' branch to one layer. The second branch is comprised of up to three layers of neurons, with each one having fewer neurons than the one before it. Both branches have K neurons in their final layer, with K ranging from 1 to 6 in KXG[13]. The models' output layer is the dot product of the outputs from the final layers of the two branches; the layers' isomorphism enables this operation. An activation function transforms incoming data as it passes through the layers; in particular, the models' activation function is the ReLU, short for Rectified Linear Unit; ReLU maps all negative values to a 0. Contrarily to other activation functions, such as tanh and sigmoid, it prevents the gradient from vanishing as it propagates backward; furthermore, the usage of batch normalization, which we will later elaborate on, serves to counter ReLU's tendency to let the gradient explode. Although KXG[13] do not mention the loss function they make use of, we have adopted the mean squared error in ours as we prioritize minimizing the error on the outliers at the expense of the bulk of the cross- section; we chose to penalize the latter because the portfolios built around the models' predictions are long-short decile spreads, and thus only include outliers. The original model is built around the dataset of KXG[13] and presents 94 neurons in both input layers. They source their data from the Center for Research on Security Prices (CRSP), a strong affiliate of the Booth school of business. The Qi4M dataset presents 123 variables, some are macroeconomic, and some are stock-level characteristics; its models thus present 123 neurons in their input layers. As the number of returns in each month of the dataset is irregular, we build all months up to contain the same amount of equities' returns as the month with the most returns. We

perform padding by appending vectors of P zeroes to the months' list of characteristics; we do so as long as the length of any given month equals that of the one with the most returns.

Figure 1: the Autoencoder model



Note: the figure represents the structure of the Autoencoder model we make use of. In particular, this is its configuration AE2, which is comprised of 2 hidden layers on its left branch. The left branch computes the P=94 asset characteristics into K=5 betas. The right branch is a shallow Autoencoder with 94 long short decile-spread portfolios built around each variable. Both branches' outputs are combined into the returns via dot product.

2.2 Optimization

During training, fitting the model on the whole dataset would increase computational costs and training time to the point of infeasibility. In order to optimize the training proces; we use the adaptive moment estimation algorithm (ADAM) Kingma, and Ba (2014)[3], ADAM is a more efficient version of classical stochastic gradient descent.
We posit that the literature may consider switching from ADAM to SWATS, a hybrid strategy as proposed by Keskar, and Socher (2017)[9], where ADAM's efficiency is leveraged during the initial stages of training; meanwhile, SDG's generalization capabilities allow for lower testing error when it's used to fit the model in the later stages of training. During optimization, we adopt a forward walking approach: this entails expanding by one month our training sample after every epoch; given the size of the Qi4M dataset, the computational intensity of refitting for every month is significantly reduced.

2.3 Regularization techniques

We employ a vast array of regularization techniques, eroding the model's in-sample performance with the aim of improving out-of-sample (OOS) performance by limiting overfitting.
Firstly, we append to our formulation of the loss function a penalty function, namely the Least Absolute Shrinkage and Selection Operator (LASSO or L1). L1 is added to the loss function, which in our case is the mean squared error. L1 is the sum of the absolute value of all weights in the model multiplied by its hyperparameter λ; this forces the weights that do not significantly improve performance to disappear, thus reducing the model's capacity for overfitting the training data or memorizing it.
Moreover, we make use of early stopping, a technique that involves stopping the fitting process as soon as the loss on the validation sample starts increasing. During training, after every epoch, we compare the performance of the model on the validation sample to the best performance it has achieved up to that point; if the model outperforms itself, we save its current state (weights and biases); after a predetermined number of epochs, if the model hasn't outperformed itself yet, we stop the training. After training, the best configuration of the model is either trained further or used for testing. Early stopping reduces overfitting on the training dataset, its hyperparameter max-iter regulates the number of times the model's validation loss can be allowed to grow before stopping the fitting; this sacrifices the possibility for the model to settle on a new optima but makes the training process faster.

Lastly, we batch-normalize the inputs of each layer of the Autoencoder; this smooths the loss function's landscape; however, it can cause gradient explosion, which is somewhat mitigated by the activation function. Batch normalization speeds up the training process to the extent that smoothing the loss function's landscape makes it more predictable; batch normalization thus makes reaching an optima easier and faster.
We further note that the literature should look into new regularization techniques, such as Shake-Shake regularization, Xavier Gastaldi (2017)[14], to address the model's strong tendency to overfit. Shake-shake regularization, if applied to this model, would involve randomly scaling down the significance of beta/factor loadings pairs during training to improve generalization in a way that mimics dropout.
Given the fact that the last layer of the Autoencoder is not normalized nor regularized and the fact that the Autoencoder is a multi-branch neural network, the model would be compatible with Shake-Shake regularization. Similarly to traditional dropout, Shake-Shake would be turned off after training, thus preserving the economically guided structure of the Autoencoder and the interpretation of returns as a combination of betas and factors.

2.4 The Data

The dataset from KXG[13] presents 94 asset-level predictive characteristics per stock per month. KXG[13] have already controlled for the looking forward bias in the dataset they released; however, they did not release the returns associated with each stock. We sourced the returns directly from the center for research on security prices (CRSP) through Kaggle. Both the CRSP dataset and the one from KXG[13] are in SAS.
The Qi4M dataset is smaller than the CRSP one by two orders of magnitude and presents 123 asset-level predictive characteristics, including some macroeconomic variables; being more dimensional and smaller, it challenges the model as it forces it to go over the same data more often; moreover, being exposed to more characteristics further fuels the model's tendency to overfit. Moreover, the Qi4M dataset is more geographically diverse to the extent that it's inclusive of EMEA-based equities. This adds a layer of diversity to the cross-section and allows for further insight into the model's applications to non-us-based portfolios.

We maintain the dataset split from KXG[13] on both the model that is trained on the Center for Research on Security Prices dataset, and the model that is trained on the Qi4M dataset.
We divide the CRSP dataset into three samples. The first sample is the

training sample and runs from 1957 to 2008. The second sample is the validation sample and runs from 2008 to 2014. The last sample is the testing sample and runs from 2014 to 2020.
Similarly, we divide the Qi4M dataset into three samples. The training sample runs from 2000 to 2017. The validation sample runs from 2017 to 2020. The testing sample runs from 2020 to 2022. As mentioned in the Optimization subsection, we adopt a forward walking-approach to using those samples.

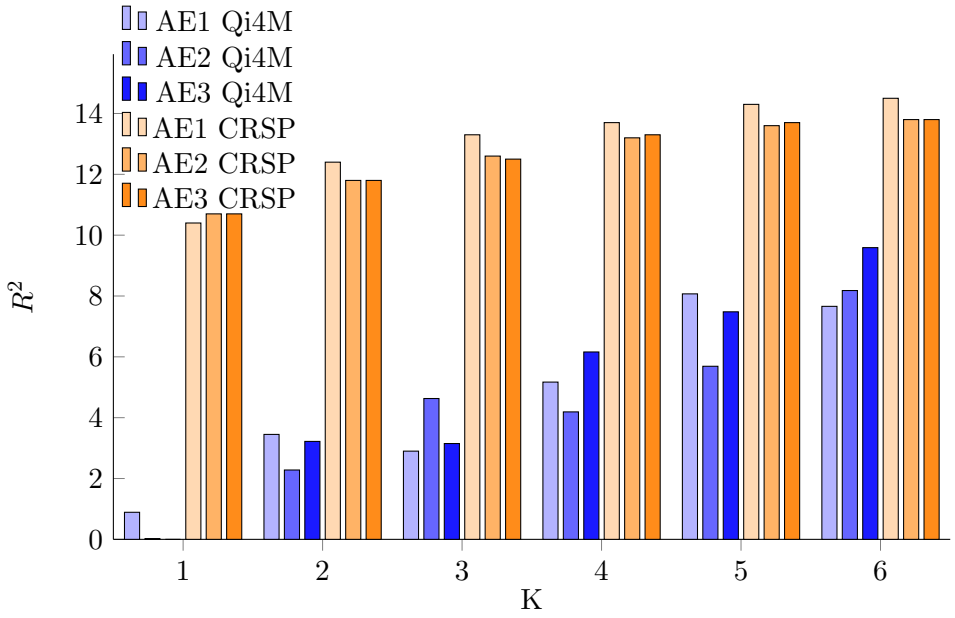
3 Results

For the model's formulations with one, two, and three hidden layers; we calculate the R^2 on the testing dataset to evaluate out-of-sample explanatory power for different configurations of the model; R^2 is a measure devised by Kelly, Pruitt, and Su (2018)[1] that indicates how much of the variance in the cross-section of returns is explained by covariates and factor loadings.

$$R^2 = 1 - \frac{\sum_{i,t}(r_{i,t+1} - (\beta_{i,t-1} f_t))^2}{\sum_{i,t} r_{i,t+1}^2}$$

We report total R^2 from KXG[13] and from the models' iterations over the Qi4M dataset. We report the performance of different formulations of the model: AE1, AE2, and AE3; they are differentiated on the basis of the number of hidden layers each of them has: AE1 has one hidden layer, AE2 has two hidden layers, and AE3 has three hidden layers.
We report the results that each model has achieved for values of K ranging from 1 to 6; K represents the number of neurons in the output layers of both branches of the Autoencoders. K is the size of the bottleneck that the data is forced to go through, and is the amount of dimensions that the data is being compressed into; i.e.: in an Autoencoder that's been trained on the Qi4M dataset and has K = 2, 123 variables are being compressed into just two beta/factor-loadings pair.

	K values					
Model	1	2	3	4	5	6
AE1 Qi4M	0.89%	3.45%	2.90%	5.17%	8.07%	7.66%
AE1 CRSP	10.4%	12.4%	13.3%	13.7%	14.3%	14.5%
Δ	9.51	8.95	10.4	8.53	6.23	6.84
AE2 Qi4M	0.02%	2.28%	4.63%	4.19%	5.69%	8.18%
AE2 CRSP	10.7%	11.8%	12.6%	13.2%	13.6%	13.8%
Δ	10.68	9.52	7.97	9.01	7.91	5.56
AE3 Qi4M	< 0	3.22%	3.15%	6.16%	7.48%	9.59%
AE3 CRSP	10.7%	11.8%	12.5%	13.3%	13.7%	13.8%
Δ	10.7	8.58	9.35	7.14	6.22	4.21



Note: we report the out-of-sample R^2 achieved by formulations of the model AE1 to AE2 for values of K ranging from 1 to 6, and on the Qi4M and CRSP datasets.

The model achieves positive R^2s on almost all of its iterations on the testing dataset; there is still a large Δ between the results from KXG[13] and the ones we achieved on the Qi4M dataset. However, it tends to shrink as K increases, as the model is afforded a less parsimonious representation of the data and thus manages a higher degree of accuracy. We infer that the Autoencoder does have statistically significant explanatory power even on a cross-section that is two orders of magnitude more sparse than KXG[13]'s. The model is still outperformed across all formulations by its peers fitted on CRSP data; this is the expected outcome and is caused by both the lower amount of fitting data, giving the model fewer learning opportunities, and the fact that a higher amount of asset-level predictive characteristics makes the model more prone to overfitting on the training data. The lower K formulations of the model tend to be much more penalized by the Qi4M dataset than on the CRSP one, owing to the fact that the model finds it much more difficult to meaningfully compress into lower values of K the Qi4M variables, which are more numerous and varied in nature than KXG[13]'s. Lastly, we observe more variance among the performances of iterations of the models fitted on the Qi4M dataset; we attribute this higher degree of variance to the fact that we weren't able to adopt an ensemble approach to training the model.

4 Conclusions

Our main conclusion is that the model does have some capacity for learning even from a cross-section of returns that is far more sparse than the one it was originally devised on. Its best formulation achieves an R^2s that is roughly comparable to that of KXG[13]'s least performing model, with a Δ of just 0.81 between the two. We also infer that having a cross-section cursed by higher degrees of dimensionality compounds the sparsity of a smaller dataset, and thus both erodes the model's encoding ability when presented with more challenging bottlenecks (the average R^2 for $\mathrm{K} = 1$ was 0.3), and facilitates the overfitting of the data by effectively fueling the model's capacity for learning individual data points instead of a meaningful representation of the cross-section by providing the model with more features per point.

References

[1] Bryan Kelly, Seth Pruitt, and Yinan Su. Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, 2019.

[2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

[3] Diederik Kingma, and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2014.

[4] Fama, and French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 1993.

[5] Fama, and French. A five-factor asset pricing model. *Journal of Mathematical Finance*, 2015.

[6] Gary Chamberlain, and Michael Rothschild. Handbook of mathematical functions with formulas, graphs, and mathematical tables. *Econometrica*, 1982.

[7] Gregory Connor, and Robert Korajczyk. Risk and return in an equilibrium apt: Application of a new test methodology. *Journal of Financial Economics*, 1988.

[8] Ioffe, Sergey, Szegedy, and Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 2015.

[9] Keskar, and Socher. Improving generalization performance by switching from adam to sgd. *arXiv*, 2017.

[10] Mclean, and Pontiff. Does academic research destroy stock return predictability? *The Journal of Finance*, 2015.

[11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 1996.

[12] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Conference on Neural Information Processing Systems*, 2018.

[13] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Autoencoder asset pricing models. *Journal of Econometrics*, 2019.

[14] Xavier Gastaldi. Shake-shake regularization. *ArXiv*, 2017.