Exploring Noise in Data: Applications to ML Models

Robert Dunn, Cheolmin Hwang, and Amelia Kawasaki

0 Abstract

In machine learning, models are commonly built in such a way to avoid what is known as overfitting. As it is generally understood, overfitting is when a model is fit exactly to the training data causing the model to have poor performance on new examples. This means that overfit models tend to have poor accuracy on unseen data because the model is fit exactly to the training data. Therefore, in order to generalize to all examples of data and not only the examples found in a given training set, models are built with certain techniques to avoid fitting the data exactly. However, it can be found that overfitting does not always work in this way that one might expect as will be shown by fitting models with a given level of noisiness. Specifically, it is seen that some models fit exactly to data with high levels of noise still produce results with high accuracy whereas others are more prone to overfitting.

1 Introduction

In our domain, we have studied recent papers that have rigorously shown the effects of building a model that has been heavily over-parameterized. Such models are often trained to have zero or near zero training loss, leading to believe that such models would be overfit and therefore generalize poorly to unknown test datasets. However, we were able to discover that this problem does not occur but rather the models still perform as well on test data as a non-overfit model. Furthermore, in some cases, we were able to observe that a model that would typically be considered overfit performs even better than a model built with the pretense of avoiding overfitting (“Reconciling Modern Machine-Learning Practice”).

This concept of purposefully overfitting a model opposes the traditional bias-variance trade-off paradigm that is well-known in data science, where a bit of error rate is left in the training phase in order to prevent the model from being too biased on the dataset. In quarter one, we attempted to replicate the empirical results from a related paper using the MNIST handwriting recognition dataset with Laplacian and Gaussian kernel machines ("To understand deep learning we need to understand kernel learning”). We were able to reproduce these results, scoring near 100% on testing accuracy on the test data with the overfitted model.

For our project in quarter two, we investigated the effects of noise and data corruption on the accuracy of multiple different types of overfitted models. These models include Laplacian / Gaussian kernel machines, k-Nearest Neighbor classification, decision trees, logistic regression, and neural networks. This study is interesting because if a model that is overfit on corrupted data performs near-optimally on test data (once the noise is accounted for), overfitting of the dataset with many parameters may become the new paradigm of machine learning— not leaving any error in the training phase.

There are results from previous literature pertaining specifically to kernel machines and noise levels and we included their replication with the rest of our results to present a full picture of models interacting with corrupted data ("To understand deep learning we need to understand kernel learning"). Additionally, research into generalization error for convolutional neural networks has been done but there is room for a more detailed, clearer investigation into the effect corrupt data and noise has on model accuracy (Zhang).

2 Methodology

Our project was implemented in Python. We have primarily focused on testing models with the MNIST dataset and other image datasets, as well as creating our own synthetic datasets. We have used these datasets to train the Laplacian and Gaussian kernel machines, k-nearest neighbor classifiers, random forests, and neural networks. To compare our results across datasets and models, we used graphical representations of accuracy for ease analysis.

2.1 Corruption Filters

The first form of corruption applied to the data we used is label corruption. In doing so, a set proportion of randomly selected labels are given a new label in a uniformly random manner. For example, in the case of the MNIST dataset, an image that has been chosen to have its label randomized would be reassigned a value between 0 and 9, meaning that such image would have a ten percent chance of being reassigned the correct label. In a more general sense, this would similarly apply to any other dataset in which the chance of being reassigned the correct label would be equal to 1 in the number of labels in the dataset.

Another form of corruption we used is random corruption. Unlike label corruption, random corruption targets the actual data while keeping the labels clean. In this process, each data point has a set proportion of its dimensions reassigned uniformly at random (shown in Figure 1). Again using the MNIST dataset as an example, this would mean that if a pixel in the image were to be randomized, it would be reassigned a completely random value spanning from the minimum value possible to the maximum value possible. Similar to label corruption, this means that each randomized value has the possibility to be reassigned the original value, however the chance of such an occurrence is much more rare than that of label randomization.

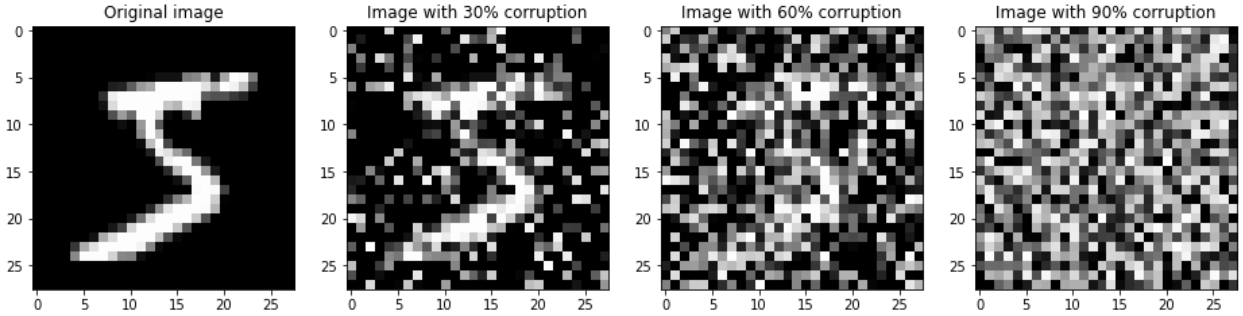


Figure 1 - MNIST digit with various levels of random corruption applied

2.2 Models Used

Model Type	Implementation	Optimized Parameters
Laplacian/Gaussian kernel machines	Custom Implementation	Power of kernel function
K-nearest neighbors classification	scikit-learn's KNeighborsClassifier	Number of neighbors
Random Forests	scikit-learn's RandomForestClassifier	Number of decision trees
Neural networks	TBD	TBD

For our custom implementation of the kernel machine models we derived the following formula for general kernel machines given two matrices.

$$K(X, Z) = exp(-\frac{||X-Z||^p}{p\sigma^p})$$

$$\sigma = \frac{1}{n}\sum||X - Z||$$

When $p = 1$ the kernel machine is Laplacian and when $p = 2$ the kernel machine is Gaussian. Using this formula, we created kernel machine implementations in Python for data corruption analysis.

3 Results + Analysis

In terms of exactly fitting a model with corrupted values, it would generally be expected that the resulting performance on uncorrupted testing data would be inversely proportional to the proportion of corruption in the dataset. However, given that it is expected for real world data to naturally be messy or corrupted, it would be beneficial to create models that can still perform well despite being trained with incorrect data. Therefore, we can compare different models trained with different levels and types of corruption to see what models produce this kind of effect.

3.1 Label Corruption Results

When looking solely at the performance of the kernel machines (Figure 2), there is no clear winner in terms of which kernel function is the best. However, from this data, it can be observed that certain models perform better under different circumstances. More specifically, it is shown that kernel machines with high power kernel functions have the best performance on extremely

clean data, however they fail on messier datasets. On the other hand, kernel machines with low power kernel functions sacrifice some of the initial performance on clean data. However, they end up being much more resistant to the effects of corruption, resulting in higher accuracy from corrupted data than kernel machines with high power kernel functions. Therefore, in order to get the best performance out of a model, it becomes important to understand the initial data and how messy it is expected to be. For example, if one expects that the given dataset is quite messy, it would be in the best interest to use a low power kernel function whereas if it is known that the dataset is clean, it would be in the best interest to use a high power kernel function.

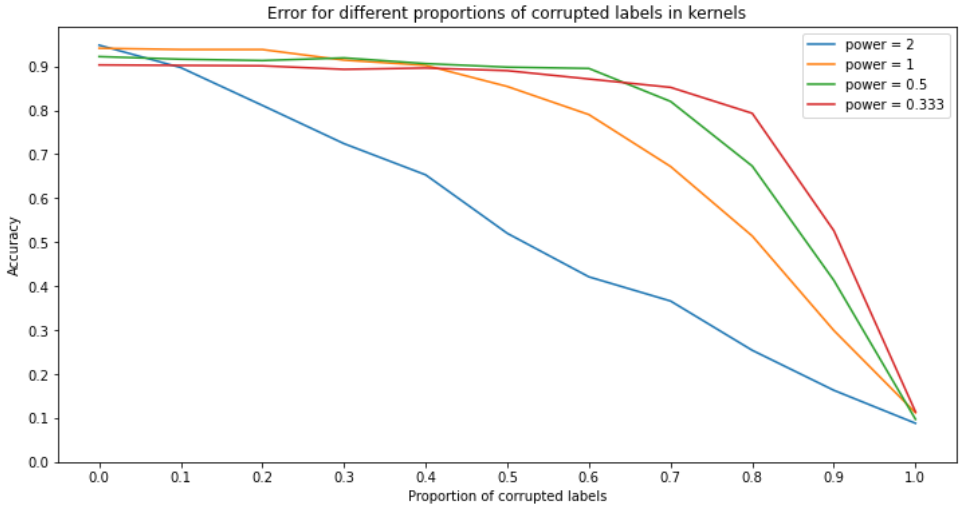


Figure 2 - Kernel machines trained with label corruption (MNIST)

In a similar sense to kernel machines, k-nearest neighbors (Figure 3) has no outright best performing model in terms of how many neighbors are used in the classification process. In terms of what is most resistant to corruption, it is clear that having a large number of neighbors is beneficial. However, it can also be observed that having a high number of neighbors results in worse initial performance on a clean dataset.

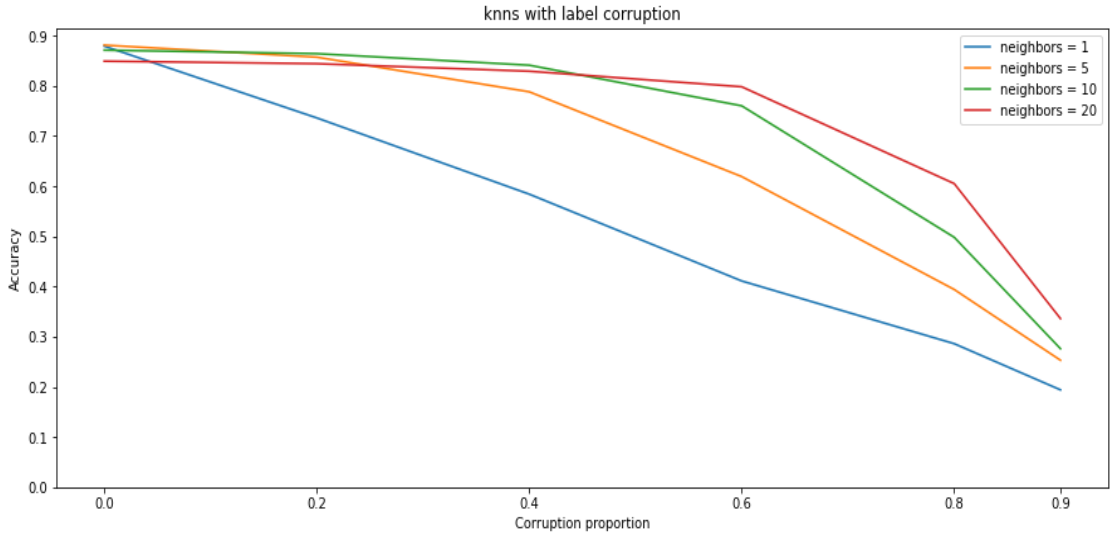


Figure 3 - k-Nearest Neighbors trained with label corruption (MNIST)

In contrast to both kernel machines and k-nearest neighbors, random forests (Figure 4) do have a clear winner in terms of the best performing model. At all levels of corruption, it can be seen that forests with larger numbers of trees always outperforms those with less trees, regardless of corruption level. Therefore, this becomes a simpler problem in terms of choosing how to build a random forest even without knowing about how messy the dataset may or may not be. However it should also be noted that adding trees to a random forest increases the level of computational complexity of the model whereas in a model such as kernel machines, changing the power of the kernel function in a kernel machine is relatively equal regardless of its power.

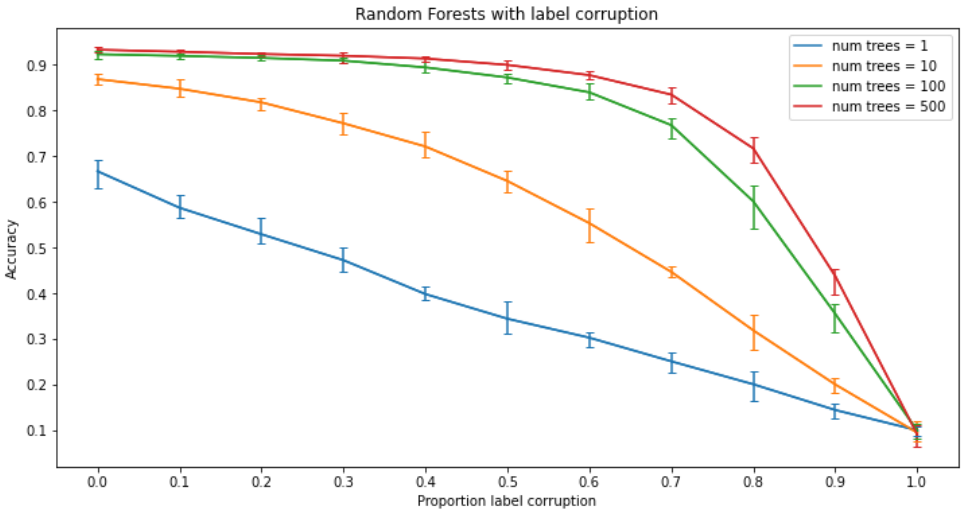


Figure 4 - Random forests trained with label corruption (MNIST)

Unlike the previous model types, only one model was used for the neural networks, however in order to view the effects of corruption on them, testing accuracy was calculated after training the network for a certain number of epochs. By doing so, it can be seen that such a model is very prone to overfitting. However, by using early stopping, the same model can perform well on a testing set. Looking at Figure 5, it is seen that continuously training a neural network either increases, or keeps the results roughly the same, whereas even at 10% corruption, training for too long tends to fit the corrupted data resulting in an immediate decrease in testing accuracy, whereas with the use of early stopping, the network does not begin to see a large decrease in performance until around 60% corruption.

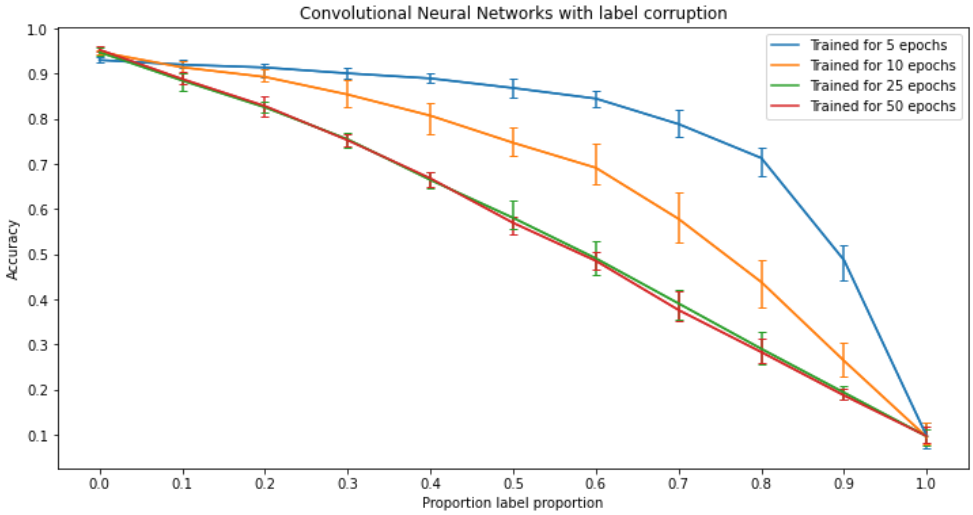


Figure 4 - Neural Networks trained with label corruption (MNIST)

When comparing the performance of each different kind of model, there is no exact answer for what is the universally best model to use, as each different model has clear positives and negatives to them. On one hand, it may seem ideal to use random forests because it has the advantage of its best performing model working the best on both clean and messy data. However, in terms of the exact values of accuracy, it can be observed that certain kernels have better performance in certain conditions. For example, the kernels with higher power kernel functions have better performance than the forests with very clean data, whereas kernels with lower power kernel functions outperform the trees at higher levels of corruption. All in all, this highlights the importance of model selection based on a general understanding of the dataset being worked with. Also, in the case where it is difficult to tell if the data is messy or not, models such as the random forest allow for good general use.

3.2 Random Corruption Results

When looking at the performance of the kernel machines (Figure 5), models with a power of 2 (Gaussian) perform best of all implementations tested. The accuracy levels are surprisingly high; we did not expect to see approximately 80% accuracy with 50% data corruption. Currently, we are unsure how significant these results are and we plan to implement proper k-fold validation procedures to make our models as robust as possible.

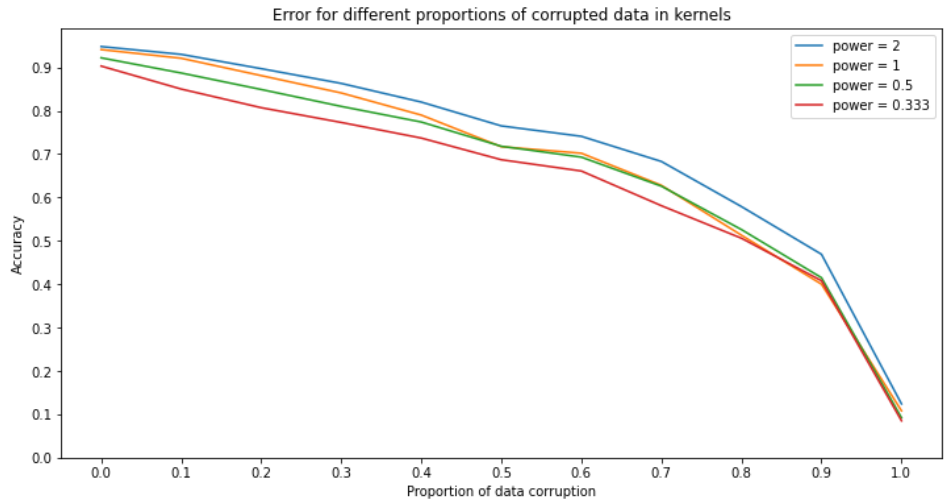


Figure 6 - Kernel machines trained with random data corruption (MNIST)

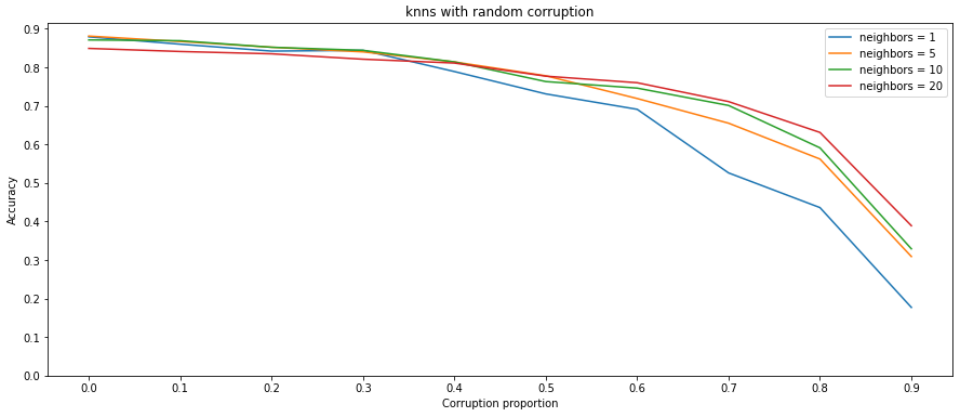


Figure 7 - k-Nearest Neighbors trained with random data corruption (MNIST)

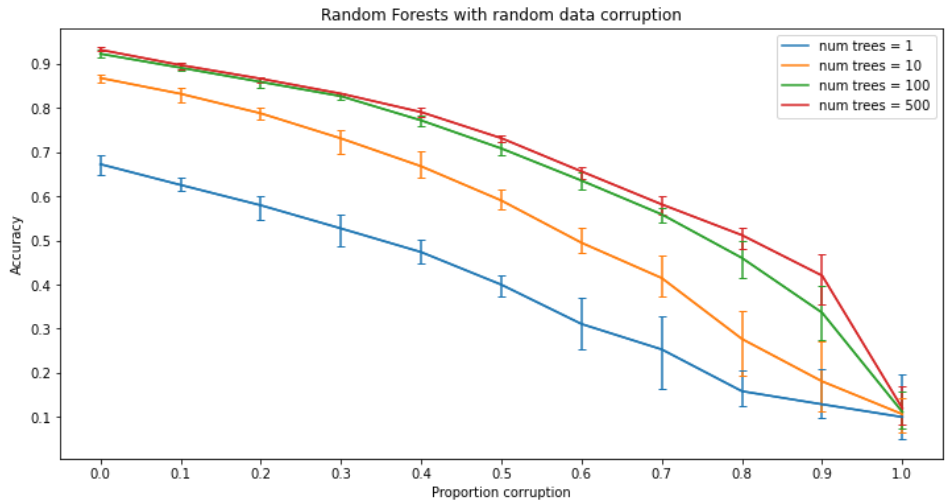


Figure 8 - Random forests trained with random data corruption (MNIST)

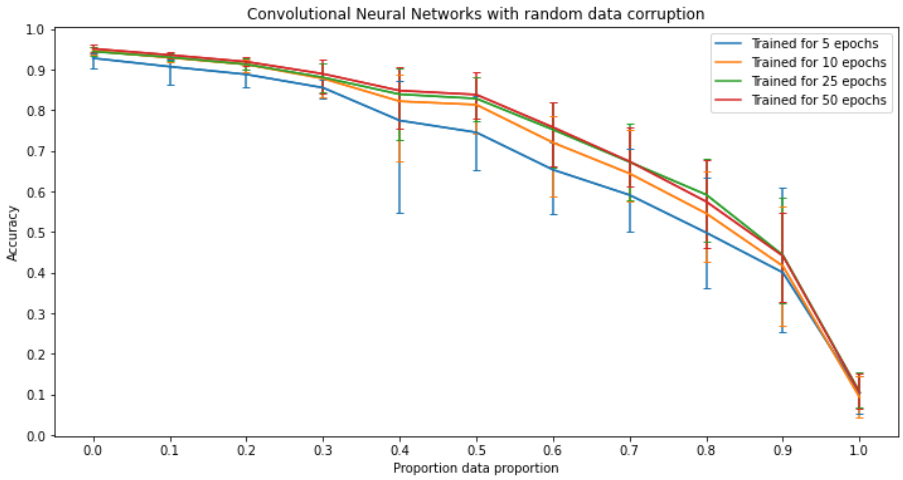


Figure 9 - Neural Networks trained with random data corruption (MNIST)

4 Conclusion

To sum up, from the insight that models that are overfit on the training data do not actually perform worse on the testing data when compared to models that leave training error, we added the factor of corrupted data. To test the effects of corrupted data on the performance of our models on the testing data, we utilized multiple different types of models and multiple datasets. From the initial Laplacian and Gaussian kernel machines trained on the MNIST handwriting image dataset, we further implemented the K-Nearest Neighbor classifier, random forest, and logistic regression model. (Neural Networks are to be implemented soon). On each model, we trained it on one dataset for each model while gradually increasing the proportion of the dataset that was corrupted through label corruption or random corruption.

From the performance results we were able to comprehend that each model had different strengths and weaknesses. From the kernel machines, we were able to conclude that the power of the kernel functions were a key factor in deciding its performance in relation to the corruption level of the dataset. The higher the power, the better its performance was on clean datasets and less resistant to data corruption— meaning that as the data became corrupted, it rapidly decreased in accuracy. The lower the power of the kernel function, the more resistant it was to data corruption— leading to significantly better performance of accuracy than the higher power kernel functions when trained on the same level of data corruption.

The K-Nearest Neighbors classifier also gave similar results. The higher the number of neighbors, the more resistant it was to data corruption. The opposite gave result to less resistance and better accuracy on the initial clean dataset.

However, from the results, we were able to observe that this was not the case for random forests. The higher the number of trees the better the accuracy was on every level of data corruption, but at the cost of higher computation cost.

Thus, from the results so far, we were able to learn that since each model has strengths and weaknesses according to the value of the optimized parameter, there is no one ultimate or ideal model to every dataset. We further concluded that to output the best results in terms of performance of accuracy on the testing/unseen data, it is crucial to understand how much the training data is corrupted, so that the adequate model would be selected.

5 Appendix - Copy of Project Proposal

Topic - Exploring Noise in Data: Applications to ML models

In our domain, we have studied recent papers that have rigorously shown the effects of building a model that has been heavily over-parameterized. Such models are often trained to have zero or near zero training loss, leading to believe that such models would be overfit and therefore generalize poorly to unknown test datasets. However, we were able to discover that this problem does not occur but rather the models still perform as well on test data as a non-overfit model. Furthermore, in some cases, we were able to observe that a model that would typically be considered overfit performs even better than a model built with the pretense of avoiding overfitting (“Reconciling Modern Machine-Learning Practice”).

This concept of purposefully overfitting a model opposes the traditional bias-variance trade-off paradigm that is well-known in data science, where a bit of error rate is left in the training phase in order to prevent the model from being too biased on the dataset. In quarter one, we attempted to replicate the empirical results from a related paper using the MNIST handwriting recognition dataset with Laplacian and Gaussian kernel machines ("To understand deep learning we need to understand kernel learning”). We were able to reproduce these results, scoring near 100% on testing accuracy on the test data with the overfitted model.

For our project in quarter two, we plan to investigate the effects of noise and data corruption on the accuracy of overfitted Laplacian/Gaussian models. We will also replicate empirical results from Professor Belkin’s “Reconciling modern machine-learning practice and the classical bias-variance trade-off” pertaining to accuracy of overfitted neural networks and other types of machine learning models. These lines of investigation are extensions of our replication project in quarter one - through the addition of new models, new datasets, factors of noise and data corruption, and different means of learning, such as a gradient descent approach. This study is considered interesting because if a model that is overfit on corrupted data performs near-optimally on test data (once the noise is accounted for), overfitting of the dataset with many parameters may become the new paradigm of machine learning. There are results from previous literature pertaining specifically to kernel machines and noise levels ("To understand deep learning we need to understand kernel learning”). Such was the basis for the replication project with clean data done in quarter one. Additionally, research into generalization error for convolutional neural networks has been done but there is room for a more detailed, clearer investigation into the effect corrupt data and noise has on model accuracy (Zhang).

Our project will be implemented in Python. We’ve already created Laplacian and Gaussian kernel models in Python for our replication project and we intend to use those for our quarter two project. We intend to use various datasets from packages such as keras.datasets and sklearn.datasets as well as create one or two synthetic datasets to further test our overfitted models and engineer different levels of data corruption and noise. We plan to investigate a variety of models for overfitting accuracy - the final list of models will depend on time constraints. Our current ideas include neural networks, k-nearest neighbors classifiers, decision trees, and multinomial logistic regression. We plan to start experimenting with the simplest

models and work towards the more complicated ones. To compare our results across datasets and models, we will use graphical representations of accuracy / loss for easy analysis. The project will culminate in a report of findings and analysis of results, specifically a comparison of the effectiveness when overfitting different models and different datasets with different levels of noise / corruption and the significance of how similar those results are.

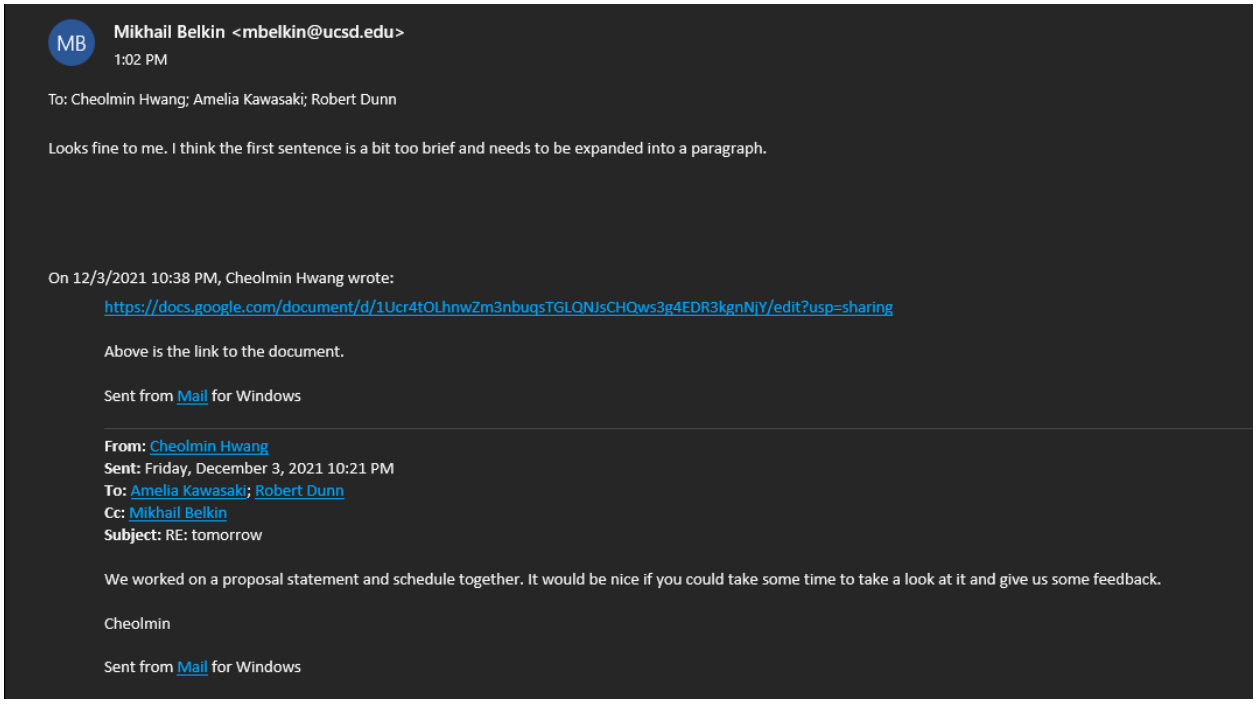
Schedule

- A proposed 6 week schedule with weekly goals and tasks for executing the proposal.
- A delineation of responsibilities among group members in the 6 week schedule.

Note: the schedule is 6 weeks to allow for tidying-up details, work on effective communication, and dealing with inevitable set-backs.

	Weekly Goal	**Amelia**	**Robert**	**Cheolmin**
Week 1	Kernels + corruption / noise + new data	Kernels + MNIST corruption	Kernels + MNIST noise	Kernels + new datasets
Week 2	kNN + Decision Trees	kNN	Decision Trees	kNN
Week 3	Logistic Regression	Implementation of Logistic Regression		
Week 4	Neural Networks - Implementation	Implementation of Neural Networks		
Week 5	Neural Networks / Tuning	Continue Neural Networks or tune previous models		
Week 6	Report Drafting	Draft corruption across models	Draft noise across models	Draft new datasets across models

Feedback from Professor Belkin



Mikhail Belkin <mbelkin@ucsd.edu>
1:02 PM

To: Cheolmin Hwang; Amelia Kawasaki; Robert Dunn

Looks fine to me. I think the first sentence is a bit too brief and needs to be expanded into a paragraph.

On 12/3/2021 10:38 PM, Cheolmin Hwang wrote:

https://docs.google.com/document/d/1Ucr4tOLhnwZm3nbuqsTGLQNJsCHQws3g4EDR3kgnNjY/edit?usp=sharing

Above is the link to the document.

Sent from Mail for Windows

From: Cheolmin Hwang
Sent: Friday, December 3, 2021 10:21 PM
To: Amelia Kawasaki; Robert Dunn
Cc: Mikhail Belkin
Subject: RE: tomorrow

We worked on a proposal statement and schedule together. It would be nice if you could take some time to take a look at it and give us some feedback.

Cheolmin

Sent from Mail for Windows

References

Belkin, Mikhail, et al. “Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off.” Proceedings of the National Academy of Sciences, vol. 116, no. 32, 2019, pp. 15849–15854., https://doi.org/10.1073/pnas.1903070116.

Belkin, Mikhail, Siyuan Ma, and Soumik Mandal. "To understand deep learning we need to understand kernel learning." International Conference on Machine Learning. PMLR, 2018.

Zhang, Chiyuan, et al. "Understanding deep learning (still) requires rethinking generalization." *Communications of the ACM* 64.3 (2021): 107-115.