An Exploration of the effects of Spurious Correlations in Deep Learning

Kevin Quinn

January 26, 2024

Abstract

How do deep learning models select important features and when can they be wrong? Spurious correlations arise when models learn to rely upon features which are correlated with the target but ultimately don't have any general predictive value. In this project I explore this effect in synthetic datasets with the goal of creating simple settings where models can fail, and understanding current methods for fixing them.

1 Introduction

When using learning algorithms with real datasets a common issue that arises is one in which the trained model learns to predict based upon subsets of features which are correlated with the target variable, but that are ultimately irrelevant and unhelpful for model generalization. In other words, when we go to test such a model on previously unseen data the predictions made can be unreliable and inaccurate. These features which the model incorrectly learns as important are referred to as being *spuriously correlated* with the target variable. One could imagine that in settings where predictions hold real consequences, this issue can be particularly dangerous and impactful, making this an important issue to address.

In this project I propose to study and characterize this effect by both exploring current research in the area and experimenting with the ideas in small synthetic datasets. My goal was to get a perspective of the landscape for deep learning research, as well as an applied understanding of how to use deep learning models while avoiding spurious pitfalls.

2 Related Work

At the core of understanding spurious correlations is understanding how models select important features and create representations of them which are then used to make predictions. This is studied practically with the use of synthetic datasets by [4], and on a more theoretical level with the use of information theoretic techniques in [12][10]. In the background of this project is the importance of understanding how and why trained models attribute importance to different features, specifically for certain classes of models and in different dataset settings. A popular idea suggested in [2] claims that in certain settings, models tend to find *shortcuts* or spurious ways of selecting predictive features that don't generalize. Likewise, [11] reports that neural networks are often biased towards using the simplest features even if other more complex ones are more predictive and generalizable.

Certain settings are more prone to spurious correlations than others. Larger, overparameterized models are a distinct model setting studied by [9], in which spurious correlations happen readily in the training process. Likewise, the spurious correlation effect happens when datasets contain multiple distinct groups of data points that all differ on some irrelevant feature. Work by [1] uses X-ray diagnosis data from distinct groups of hospitals to show that in this scenario, a trained model may learn to use such irrelevant

features to make inacurate predictions. Additional work from [13] attempts to broadly categorize different scenarios in which spurious correlations arise, and shows examples on well crafted synthetic data.

Finally, a subset of work done on this topic has proposed solutions within certain settings. This is studied by [6][7] who propose re-training for only the final classification layer of a model using some small amount of data that breaks the pattern of spurious correlations. As well as by [5][8] who both propose a strategic re-weighting of the training samples in order to improve accuracy for groups in the data which are most prone to negative effects of spurious correlations.

For the purposes of this project I focused mostly on [7] and [8]!

3 Preliminaries

Following notation from [8] I'll first make a few important definitions. Let X be a size $n \times d$ data matrix, y be a corresponding vector of labels, and the pair $D = (X, y)$ be their formed dataset. For the forthcoming experiments I will typically make distinctions between the training dataset $D_{\text{train}} = (X_{\text{train}}, y_{\text{train}})$ and a testing dataset $D_{\text{test}} = (X_{\text{test}}, y_{\text{test}})$. In a setting with spurious correlations it is assumed that there exists a set of groups $\mathcal{G} = \{G_1, ..., G_k\}$ in the data. Each of these groups have corresponding labels $g_1, .., g_k$ that make up the target labels which are seen in y. Spurious correlations happen whenever certain features in X are correlated with but not causally linked or predictive of the target labels in y.

A popular example might be a dataset where every sample corresponds to a single day, the features report the amount of ice cream sold and the daily maximum temperature, and the target variable is a $0/1$ variable indicating if any patients were admitted to a hospital with heat stroke. One might find that ice cream sales seem to be predictive heat stroke, but this totally ignores the real causal link with temperature. Furthermore, a model trained to predict based upon ice cream sales may perform well in one location, but if one tried to use it different setting (say in Alaska), they might find that predictions are no longer accurate! This of course is a simplified example, but the principle is the same when we extend to more complicated deep learning models. In this example there were 2 groups G_1 and G_2 corresponding to the days with and without patients being admitted to a hospital. The distinction is important to make so that we can later reason about the number of samples seen for each group (important for understanding correlations) and so that we can reason about which group the model is the most inaccurate for (important for fairness reasons).

The types of models I consider are simple convolutional neural networks with linear classification layer on top. Let θ be the set of all parameters for this model and let θ_{cnn} and θ_{lin} be the subsets of parameters corresponding to the convolutional and linear layers of the network separately. Baseline models will be trained to learn parameters using ERM principles:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_i \ell(x_i, y_i; \theta)$$

But I will later introduce training model which adjusts this to re-weight the data. Throughout I will use the cross-entropy loss function for ℓ. Finally I will note that I use two measures to evaluate the model's predictions. Let $\hat{y}$ be the predicted labels for a given dataset. For my experiments I will evaluate model performance based upon accuracy:

$$a(\hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{y_i = \hat{y}_i}$$

as well as worst-group accuracy:

$$w(\hat{y}) = \max_{G \in \mathcal{G}} \frac{1}{|G|} \sum_{i \in G} \mathbb{1}_{y_i = \hat{y}_i}$$

4 Dataset

To build a dataset for experimentation, I adapted the MNIST-1d dataset [3] in order to have spurious correlations. The original dataset is a lower dimensional representation of the popular MNIST dataset for classification of handwritten digits 0-9. It offers a simple and customizable setting to experiment with. From this I took some pre-defined training $D_{\text{base}} = (X_{\text{train}}, y_{\text{train}})$ and testing datasets $D_{\text{test}} = (X_{\text{test}}, y_{\text{test}})$. Points are 40 dimensional vectors, representations of which are pictured in figure 1.

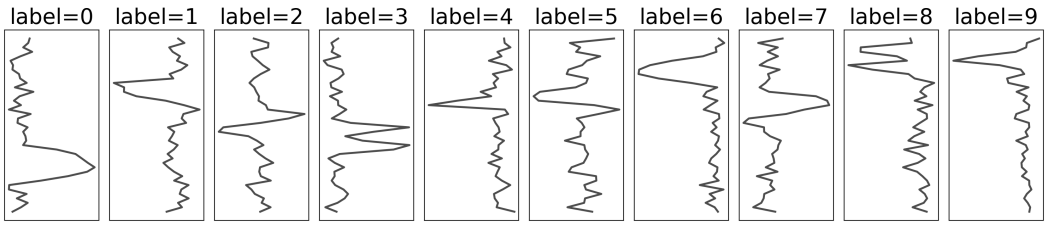

Figure 1: MNIST-1d is a noisy, low dimensional representation of handwritten digits 0-9. Here each entry on the y axis corresponds to a feature and the variation on the x axis reflects feature values.

Inspired by [11] I created some very simple feature changes to the training data set in order to introduce spurious correlations. To a percent p of the samples in X_{train} I added a large amount noise to a subset of the features depending on it's true label. For example, I might generate a sample from a truncated (non-negative) gaussian distribution with mean 2 and variance 1, and then for every data point in the training set with label 1 I'd add this value on to the features 1-5. Then for points with label 2 I'd generate another noise sample and add this on to features 5-10. I denote this transformed dataset as $D_{\text{sp}} = (X'_{\text{train}}, y_{\text{train}})$, and make a note that the training labels y_{train} remain the same. A representation of this is pictured in figure 2.

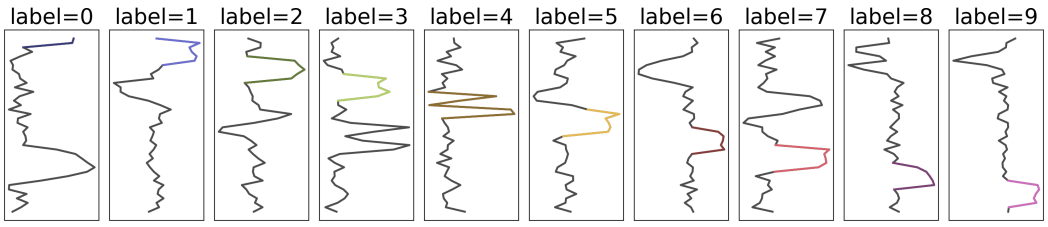

Figure 2: Noise is added to distinct subsets of the features for each of the groups 0-9

The idea was to create some very simple correlations in the data. For example, the model could now pick up on the fact that seeing large values in features 5-10 of a data point would be highly correlated with having label 2, even though another subset of features are the true characteristics of the group. In some cases the the noisy features overlap with the true ones, creating even more difficulties for the model. Throughout the rest of the paper, I'll use $p = 90\%$ so that spurious correlations are particularly difficult to avoid. With more time I think it would have been really interesting to observe how model performance changed as p is varied.

The goal of this simple transformation was to create a setting where a model could pick up on these 'easy' spuriously correlated features and use them for prediction instead of the true ones. The result is that when a model is trained on the spuriously correlated data it becomes artificially easier learn, showing

large but misleading accuracy scores. To see this visually, I've plotted a 2-dimensional embedding of both the MNIST-1d and spurious MNIST-1d dataset in figures 3 and 4. In both images embedded data points are colored according to their group label. The original dataset in figure 3 is more difficult to decode, with many groups overlapping in the middle of the image. On the other hand, with spurious correlations, figure 4 shows an embedding which is much easier to pick apart and to create boundaries between classes for. In other words, groups are positioned more distinctly within the space. Of course, with a 2 dimensional embedding this is not expected to be perfect, but I thought this was interesting and worth a mention!

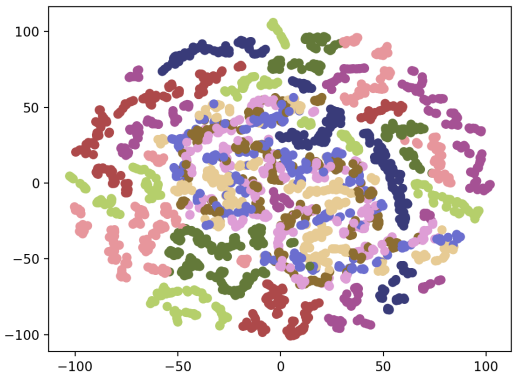

Figure 3: MNIST-1d

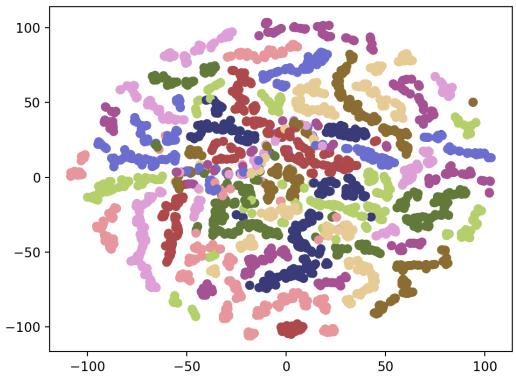

Figure 4: MNIST-1d with spurious correlation

For models trained on either the baseline or spurious datasets, I test on the same dataset D_{test}. Once a model has learned to pick up on the spuriously correlated features, it should become much harder for it to perform well on the test data. Comparatively it should do much worse than the baseline trained model. The idea is that by testing like this on non-spurious data, we can get a sense of how well the model learns true feature representations in the presence of the spurious ones.

5 Experiments

With D_{base}, D_{sp}, and D_{test} I performed comparitive experiments with simple deep learning models, while also attempting to recreate the results from [7] and [8]. I began the with same convolutional neural network model used by [3], which is composed of a three layer convolutional feature extractor that's passed into a final linear classification layer. The hidden size of the convolutional layers is 256 and I train with batch size of 100 using an adam optimizer. The model is validated by first performing training on D_{base} with which we can achieve about 96% test accuracy and 90% worst group accuracy. This can be immediately compared to performance of a the same model trained upon D_{sp}, in which case we find about 65% accuracy and 44% worst group accuracy, indicating that the spurious noise has a clear detrimental effect on test accuracy. The test performance results are pictured in figures 5 and 6.

Importantly I also tried some strategies from [7] and [8] which aim to fix the problem:

Re-train The first strategy from [7] freezes all the parameters θ_{cnn} within the convolutional layers of the model with the goal of only retraining the parameters for the last linear classification layer of the network, θ_{lin}. To do so, a small amount of data unaffected by the spurious noise is needed for re-training. In the results shown, I re-train the last layer of the model originally trained with D_{sp} with a random sample containing 25% of the entries from D_{base}. The advantage of this strategy is both simplicity and speed. Re-training the last layer is fast and doesn't require much memory – this layer is typically much smaller than the convolutional feature extractors. The disadvantage, however, is the need for a set of

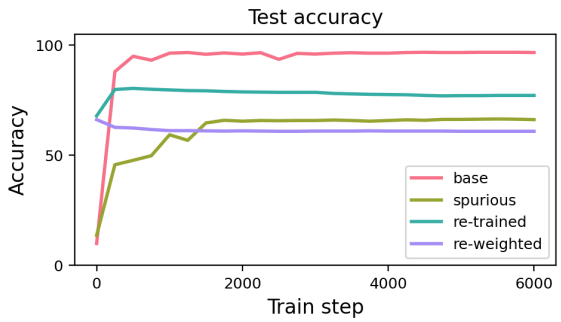

Figure 5: Accuracy test performance

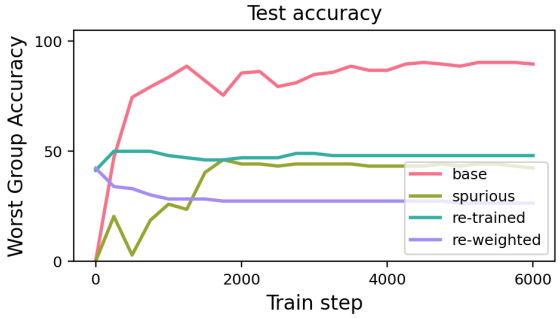

Figure 6: Worst Group Accuracy test performance

non-spurious data to re-train with. In practice I'd imagine that without a clear handle on what's spurious or not, it'd be pretty difficult to settle on a good dataset to re-train with. As seen in figures 5 and 6 this strategy does seem to significantly improve test accuracy, but doesn't improve much in terms of worst group accuracy.

Re-weight The next strategy introduced by [8] does not come with the same dataset requirements and instead involves a simple re-weighting of the data samples within the objective. The approach consists of two stages: In the first we'll train on D_{sp} exactly like we did before to find parameters $\hat{\theta}_{\text{cnn}}$ and $\hat{\theta}_{\text{lin}}$. However in the second we will redefine the objective, and re-train for the linear parameters θ_{lin}, keeping the same convolutional parameters from the first stage:

$$\mathcal{L}_{\text{re-weight}}(\theta_{\text{lin}}) = \sum_{i=1}^{n} \mu_i \, \ell(x_i, y_i; (\hat{\theta}_{\text{cnn}}, \theta_{\text{lin}})) + \lambda ||\theta_{\text{lin}} - \hat{\theta}_{\text{lin}}||^2$$

We'll keep the same cross entropy loss function for ℓ, but now all that's happening is that training samples are now weighted according to parameters μ_i (instead of every point having equal weight). It also introduces a regularization term to ensure that the re-trained parameters don't shift too far from those found in the first stage. Importantly, the authors choose weights μ_i as follows:

$$\mu_i = \frac{\beta_{y_i} e^{-\gamma p_i}}{\sum_j \beta_{y_j} e^{-\gamma p_j}}$$

Where β_{y_i} is 1 over the number of samples having label equal to y_i, p_i is the probability with which the model from stage 1 predicts sample i as belonging to the correct group, and γ is a temperature parameter controlling how much we boost or reduce weights. In words, by assigning weights like this we 1) control for irregular group sizes with βs and 2) boost importance of samples which are likely to be misclassified with $e^{-\gamma p_i}$.

Unfortunately I found that in my experiments this strategy was very un-effective, often performing worse than the original model. I believe the reason is because the model trained in stage 1 had very high training accuracy, i.e. p_i values were consistently large. In other words the boost that was supposed to be given to misclassified examples that needed to come from seeing small p_i values was not there. It's somewhat unintuitive, but I think I'd need to make the training problem harder in order for this strategy to work better!

To summarize, while re-training did see moderate success, these strategies are not without their own

unique shortcomings. Interestingly we can analyze saliency maps for each of the models to see what they focused on and or missed.

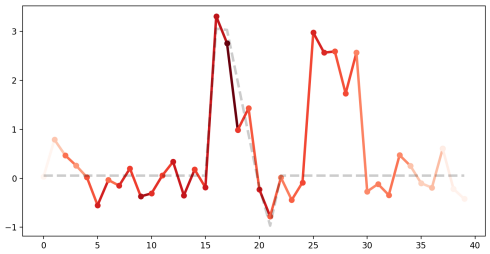

Figure 7: Base

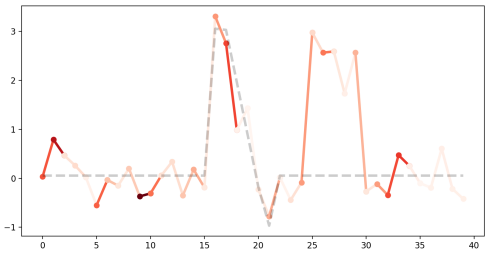

Figure 8: Spurious

Each plot in figures 7, 8, 9, and 10 shows gradient strengths attributed to each of the 40 features of a training sample from D_{sp}, when passed into the 4 different pre-trained models: base, spurious, re-trained, and re-weighted. Darker red colors indicate larger gradient information and suggest that the model is placing strong importance upon those features when trying to predict the label of the given sample. The dashed grey line indicates what the original sample looked like before adding general noise or targeted spurious noise. One can notice that the base model correctly attributes strong importance to the correct features even in the presence of spurious ones. The others, however, all fail to distinguish the correct features as important, often finding small noisy features to be the ones with strongest gradients. It's entirely possible that the model found spurious features other than the ones I actually tried to give the data!

6 Discussion

For me this project was a success because I got a chance to read some papers and implement some simple deep learning models in an interesting way. However, I was disappointed that the results for the re-training and re-weighting strategies were lackluster. But still I think some of these shortcomings are important to address: 1) Re-training requires additional non-spurious data and 2) Re-weighting requires that the samples that need the most importance are misclassified within the original training stage. If I were to continue further with this I would try two things: firstly I'd experiment more with changes to D_{base}, perhaps trying to make the training problem more difficult in order to see if I can find a scenario where re-weighting does well. The other thing is that I would experiment with the fraction of training samples for which I add in spurious noise. I'd really like to see if there's a threshold amount beyond which the problem just becomes intractable. I think there's still a lot of good questions here and I really think there's something to be said for experimenting with this type of really simple data!

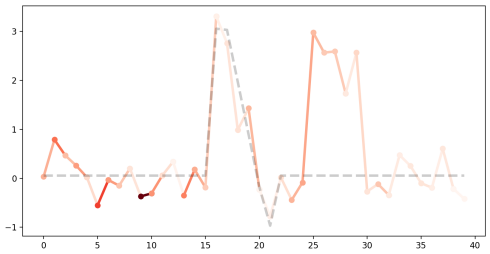

Figure 9: Re-trained

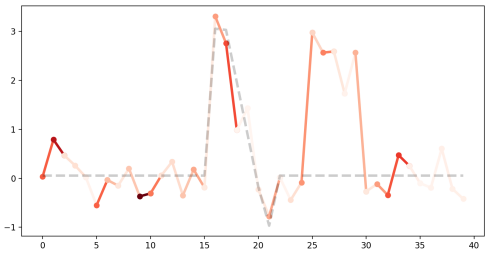

Figure 10: Re-weighted

References

[1] Rhys Compton, Lily Zhang, Aahlad Puli, and Rajesh Ranganath. When more is less: Incorporating additional datasets can hurt performance by introducing spurious correlations. In *Machine Learning for Healthcare Conference*, pages 110–127. PMLR, 2023.

[2] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[3] Sam Greydanus. Scaling down deep learning, 2020.

[4] Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *Advances in Neural Information Processing Systems*, 33:9995–10006, 2020.

[5] Badr Youbi Idrissi, Martin Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy. In *Conference on Causal Learning and Reasoning*, pages 336–351. PMLR, 2022.

[6] Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew G Wilson. On feature learning in the presence of spurious correlations. *Advances in Neural Information Processing Systems*, 35:38516–38532, 2022.

[7] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

[8] Shikai Qiu, Andres Potapczynski, Pavel Izmailov, and Andrew Gordon Wilson. Simple and fast group robustness by automatic feature reweighting. In *International Conference on Machine Learning*, pages 28448–28467. PMLR, 2023.

[9] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8346–8356. PMLR, 13–18 Jul 2020.

[10] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018.

[11] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

[12] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[13] Gautam Sreekumar and Vishnu Naresh Boddeti. Spurious correlations and where to find them. *arXiv preprint arXiv:2308.11043*, 2023.