Be Like Water: Robustness to Extraneous Variables
Via Adaptive Feature Normalization

Aakash Kaku *1 Sreyas Mohan *1 Avinash Parnandi 2 Heidi Schambra 2 Carlos Fernandez-Granda 1 3

Abstract
Extraneous variables are variables that are irrelevant for a certain task, but heavily affect the distribution of the available data. In this work, we show that the presence of such variables can degrade the performance of deep-learning models. We study three datasets where there is a strong influence of known extraneous variables: classification of upper-body movements in stroke patients, annotation of surgical activities, and recognition of corrupted images. Models trained with batch normalization learn features that are highly dependent on the extraneous variables. In batch normalization, the statistics used to normalize the features are learned from the training set and fixed at test time, which produces a mismatch in the presence of varying extraneous variables. We demonstrate that estimating the feature statistics adaptively during inference, as in instance normalization, addresses this issue, producing normalized features that are more robust to changes in the extraneous variables. This results in a significant gain in performance for different network architectures and choices of feature statistics.

1. Introduction
In many applications, the observed data depend heavily on extraneous variables that are not directly relevant to the task of interest. Changes in these variables produce significant shifts in the distribution of the data. If a machine-learning model is trained with data corresponding to a certain value of an extraneous variable, its performance may be significantly degraded when tested on data corresponding to a different value. Unfortunately, this is a common occurrence in practical applications when machine-learning models often need to be applied in changing environments. In this work, we consider the problem of performing classification in the presence of extraneous variables with fluctuating values. As in many real-world scenarios, the values of the extraneous variables observed during inference are not available during training. This is in stark contrast to most existing literature on domain adaptation, where it is usually assumed that unlabeled data corresponding to the target domain are available during training (Ganin et al., 2016; Tzeng et al., 2017; Long et al., 2013; 2017; 2016).

In this paper we study three different datasets illustrating realistic situations where it is crucial to generalize across unknown values of the extraneous variables:

- The JIGSAWS dataset, where the goal is to classify the different gestures performed by surgeons during surgery from sensor data. Here an important extraneous variable is the identity of the surgeon; different surgeons will move in systematically different ways during surgery. The ultimate goal of this task is to aid in surgeon training, so it is imperative for models to generalize to new surgeons.

- The StrokeRehab dataset, where the goal is to classify upper-body movements in stroke patients from sensor data. Similarly to the previous case, the identity of the patient is an important extraneous variable, since different patients have different levels of impairment, as well as other idiosyncrasies. Generalization to new patients during inference is critical if these models are to be deployed in realistic rehabilitation settings.

- The CIFAR-10C dataset, where the goal is to classify natural images that are subject to common corruptions such as blurring, compression artifacts, additive noise, etc. Here the extraneous variable is the type of corruption. Making computer vision models robust to unexpected corruptions is crucial in real-world deployment, e.g. for self-driving vehicles.

The main contributions of this work are the following. We identify a crucial weakness of batch normalization in settings where extraneous variables vary between training and inference: normalization is carried out with inexact esti-
 Traditional approaches to the control of extraneous variables in statistics often rely on experimental design and randomization (Box et al., 2005), and hence are not applicable to observational studies. Here we restrict the discussion to techniques that are applicable to observational data, although some are also applied in randomized settings.

Frequentist techniques to perform inference in the presence of extraneous variables are often parametric; they assume that we have available the likelihood of the data as a function of the response (i.e. the variable that we want to estimate) and the extraneous variables. In that case, one can perform inference by maximizing the so-called profile likelihood, obtained by first determining the maximum likelihood estimate of the extraneous variables given the response, and then plugging the estimate into the likelihood function (Reid, 2003; Barndorff-Nielsen, 1983; Cox & Reid, 1987; McCullagh & Tibshirani, 1990). An alternative to profile-likelihood methods is based on applying transformations that decouple the effects of the response and the extraneous variables on the data. More concretely, the goal is to find a sufficient statistic whose marginal or conditional distribution depends on the response, but not on the extraneous variables. The distributions can then be used as a likelihood function for the response. These approaches are known as marginalization and conditioning in the literature (Basu, 1977). From a Bayesian point of view, inference with respect to the response can be carried out after marginalizing with respect to the extraneous variable. This requires endowing the extraneous variables with a prior. When noninformative priors are used, this is sometimes denoted as an integrated likelihood approach (Berger et al., 1999).

Unfortunately, when the data are high dimensional, it is usually not tractable to derive a realistic likelihood function that accurately captures the interactions between the extraneous variables and the response, except in very restricted scenarios. Nonparametric approaches to the control of extraneous variables often rely on methodology derived under linearity assumptions, such as analysis of variance (Neter et al., 1996) or variable-selection techniques (Guyon & Elisseeff, 2003). A recent method based on optimal transport (Tabak & Trigila, 2018) learns a linear map that transforms the data to make it as invariant as possible to the effect of extraneous variables.

In the last 15 years, there has been significant interest in removing batch effects from high-throughput genomic data, which has been an active area of research. Batch effects are due to extraneous variables dependent on experimental variations, such as intensity changes in the scanning process, block effects, dye effects, or amount of mRNA and DNA concentration on microarrays (Lazar et al., 2012). There are two main approaches for batch-effect removal. Location-scale methods normalize the data within the batches by centering (Sims et al., 2008) and normalizing it (Li & Wong, 2001; Kim et al., 2007). Matrix-factorization methods assume that variation due to batch effects is independent of the variation due to biological variables of interest, and can be captured by a small set of factors estimated by fitting a low-rank

![Figure 1](image-url)
Robustness to Extraneous Variables Via Adaptive Feature Normalization

Figure 2. Histograms of the mean of normalized intermediate features each corresponding to a separate convolutional filter for the CIFAR-10-C, JIGSAWS, and StrokeRehab datasets. On the top row, the normalization is carried out on test data using statistics estimated from the training data (i.e. non-adaptively). The histogram of means does not concentrate around zero, as opposed to the mean histogram in Figure 1. This indicates that the non-adaptive normalization scheme is not successful in normalizing the held-out data. On the bottom row, the normalization is carried out adaptively, using batches where the value of the extraneous variable is fixed. This results in a histogram that does concentrate around zero, demonstrating that adaptive normalization succeeds in normalizing the features correctly.

3. Batch Normalization

Batch normalization is a technique that increases the robustness of deep-learning models to changes in initialization and learning rates (Ioffe & Szegedy, 2015). Due to its widespread success, it has become a standard component of deep convolutional neural networks. It consists of two steps, typically applied at each layer of the neural network: (1) centering and normalizing the intermediate features corresponding to each convolutional filter using an estimate of their mean and standard deviation, (2) scaling and shifting the normalized features using two additional learned parameters per filter (a scaling factor and a shift). During training, the mean and standard deviation are computed by averaging the features over each training batch. In addition, a running average of the mean and the variance are used to obtain a global estimate of the population mean and standard deviation of each feature. These global estimates are used to perform normalization during inference.

Understanding why batch normalization aids in the training of deep neural networks is an active area of research. Originally, it was hypothesized that it reduces internal covariate shift (Ioffe & Szegedy, 2015); more recent work suggests that it smooths the optimization landscape of the training cost function (Santurkar et al., 2018). Regardless of the precise mechanisms associated with it, batch normalization operates under a basic assumption: the mean and standard deviation of the network features are the same during train-
Robustness to Extraneous Variables Via Adaptive Feature Normalization

Figure 3. Histograms of the standard deviation of normalized intermediate features each corresponding to a separate convolutional filter for the CIFAR-10-C, JIGSAWS, and StrokeRehab datasets. On the top row, the normalization is carried out on test data using statistics estimated from the training data (i.e. non-adaptively). The histogram of standard deviations does not concentrate around one, as opposed to the standard-deviation histogram in Figure 1. This indicates that the non-adaptive normalization scheme is not successful in normalizing the held-out data. On the bottom row, the normalization is carried out adaptively, using batches where the value of the extraneous variable is fixed. This results in a histogram that does concentrate around one, demonstrating that adaptive normalization succeeds in normalizing the features correctly.

```latex
\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3}
\caption{Histograms of the standard deviation of normalized intermediate features each corresponding to a separate convolutional filter for the CIFAR-10-C, JIGSAWS, and StrokeRehab datasets. On the top row, the normalization is carried out on test data using statistics estimated from the training data (i.e. non-adaptively). The histogram of standard deviations does not concentrate around one, as opposed to the standard-deviation histogram in Figure 1. This indicates that the non-adaptive normalization scheme is not successful in normalizing the held-out data. On the bottom row, the normalization is carried out adaptively, using batches where the value of the extraneous variable is fixed. This results in a histogram that does concentrate around one, demonstrating that adaptive normalization succeeds in normalizing the features correctly.}
\end{figure}
```

For data that depends on extraneous variables, which change at test time, the training statistics may no longer resemble the train statistics. To evaluate this, we perform the same experiment as in Figure 1 for our three datasets of interest (see Section 5 for a detailed description). In this case, many of the centered, normalized test features have means and standard deviations that deviate significantly from zero and one respectively (Figures 2, 3).

4. Adaptive Normalization

As reported in the previous section, changes in extraneous variables may significantly affect the mean and standard deviation of filters in convolutional neural networks. As we report in Section 7, this mismatch is accompanied by poor classification performance. A natural solution for this issue is to perform normalization adaptively during inference, using statistics estimated from the test data. When the value of the extraneous variable is known, then this can be achieved by feeding a batch of test examples corresponding to a fixed value of the extraneous variable through the network, and averaging the resulting features to estimate the means and standard deviations for each filter. To evaluate whether this approach results in better normalization, we divide the test set of our three datasets of interest into subsets associated with different values of the relevant extraneous variable (surgeons for JIGSAWS, patients for StrokeRehab, and corruptions for CIFAR-10C). For each subset, we estimate the feature statistics from one-half of the data and use the corresponding statistics to center and normalize the other half. Figures 2, 3 show that adaptive normalization fulfills its goal: the means and standard deviations are concentrated around zero and one respectively. Extraneous variables may be easy to identify during inference for some datasets: for example, when the extraneous variable is associated with different subjects, as in JIGSAWS...
Adaptive normalization ensures that features at each layer of a convolutional neural network are normalized similarly at training and inference, even when extraneous variables associated with the data vary. Intuitively, this is a desirable property, because it prevents spurious shifts and scalings that may distort the features, and ultimately degrade classification accuracy. In Section 7, we provide numerical evidence to support this hypothesis: both batch-based and instance-based adaptive normalization significantly improve the classification accuracy for our datasets of interest. Motivated by recent work (Singh & Krishnan, 2019), we also compare non-adaptive and adaptive normalization based on mean square (as opposed to mean and standard deviation). The gains in accuracy achieved by adaptive normalization are similar.

Our observations are consistent with recent work by (Pan et al., 2018), who propose to apply batch-based non-adaptive normalization and instance-based adaptive normalization to different filters in a convolutional network. They show that this results in improved performance on a domain-adaptation task, without fine-tuning to the target domain. Our results suggest that this generalization ability may be mostly due to the use of adaptive normalization.

5. Datasets

In this section, we describe the datasets used in our computational experiments. We use two standard benchmarks for classification algorithms to represent situations where the data do not depend on extraneous variables: CIFAR-10, a dataset of 32 × 32 natural color images belonging to 10 different classes (Krizhevsky et al., 2009), and Fashion-MNIST, a dataset of 28 × 28 grayscale images of 10 different fashion items (Xiao et al., 2017). These datasets are compared to three datasets, which are influenced by known extraneous variables. These datasets, described in detail below, are the main focus of our experiments.

5.1. Surgical activities

The JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) dataset introduced in (Gao et al., 2014) contains data measuring the motion of eight different surgeons with different skill levels performing repetitions of three basic surgical tasks (suturing, knot-typing and needle-passing) on a benchtop model. The original dataset contains both video and kinematic sensor data. In our experiments, we only use the kinematic data, which contains cartesian positions, orientations, velocities, angular velocities and gripper angle describing the motion of the robotic manipulators used to perform the surgical tasks. These data correspond to a 76-dimensional time series. Each time step is manually annotated with a label indicating one of 15 possible gestures or atomic surgical activities carried out by the surgeon at that time. The classification task that we consider is classifying these gestures automatically from the sensor data. The main extraneous variable influencing the data is the identity of the surgeon because different people move in distinctive ways while performing surgery.

5.2. Stroke rehabilitation

The StrokeRehab dataset contains motion data collected while 32 stroke patients performed a battery of activities of daily living (including feeding, drinking, washing face, brushing teeth, donning glasses, etc.). The data were recorded with inertial measurement units (IMUs) affixed to the upper body of the patients. Each IMU sensor samples 3D linear acceleration and 3D angular velocity at 100 Hz. Data are integrated and fused algorithmically to provide acceleration, rotation, and quaternion values for each 3D axis. The data consequently corresponds to a 78-dimensional time series. Each multidimensional time sample in the dataset is labeled with one of five movement primitives that the patient was performing at the time (reach, idle, transport, stabilize and reposition). Movement primitives are discrete motions that can be strung together in various combinations to build most basic arm motions. In this dataset, a crucial extraneous variable is the identity of the patient. Each patient is partially paralyzed with varying levels of impairment. This results in highly idiosyncratic movement patterns associated with each patient.

5.3. Corrupted natural images

The CIFAR-10C Benchmark Dataset† is a standardized dataset designed to evaluate the robustness of image classifiers to different natural-image corruptions (Hendrycks & Dietterich, 2019). The data consist of the 10,000 validation images in CIFAR-10 perturbed by 19 different corrup-
Robustness to Extraneous Variables Via Adaptive Feature Normalization

| Normalization scheme | Non-adaptive | Adaptive |
|----------------------|--------------|----------|
| Averaging scheme     | Batch based  | Instance based | Batch based |
| Statistics           | μ, σ | ν² | μ, σ | ν² | μ, σ | ν² |
| JIGSAWS              | 67.22 | 64.88 | 72.71 | 72.51 | 65.51 | 62.55 |
| StrokeRehab          | 48.3 | 44.91 | 63.9 | 63.8 | 61.6 | 60.6 |
| CIFAR-10C            | 72.57 | 70.75 | 81.58 | 80.52 | 81.88 | 78.14 |
| CIFAR                | 94.15 | 91.27 | 92.08 | 90.99 | 91.48 | 87.47 |
| Fashion-MNIST        | 92.38 | 91.29 | 90.52 | 90.22 | 88.55 | 90.68 |

Table 1. Results of the computational experiments described in Section 6. The entries in the table report the classification performance for different normalization strategies: non-adaptive or adaptive, instance-based or batch-based, and different statistics. For the statistics, μ denotes the mean, σ denotes the standard deviation, and ν² denotes the mean square. For CIFAR-10 and Fashion-MNIST, the non-adaptive batch-based normalization schemes outperform all others, by small margins. In sharp contrast, for the datasets dependent on extraneous variables, adaptive normalization schemes outperform non-adaptive schemes by around 10% in JIGSAWS, 15% in StrokeRehab and 10% in CIFAR-10C. Similar results for other test-train splits of JIGSAW dataset is given in table 4.

6. Computational Experiments

In this section, we describe our computational experiments to study the effect of different normalization schemes on the performance of deep convolutional neural networks trained for classification. Our experiments involve two types of datasets: homogeneous datasets where there is no variation of extraneous variables between the training and test sets (CIFAR-10 and Fashion-MNIST), and heterogeneous datasets where the data distribution is influenced by extraneous variables that do change between training and test (JIGSAWS, StrokeRehab, and CIFAR-10C). For CIFAR-10 and Fashion-MNIST, we use the usual training and test sets. For the heterogeneous datasets, the data are randomly separated according to the different extraneous variables in the following way:

- **JIGSAWS**: The training set consists of data from 8 surgeons. The validation and test sets consist of data from 2 different surgeons.
- **StrokeRehab**: The training set consists of data from 21 patients. The validation and test sets consist of data from 5 and 6 different patients respectively.
- **CIFAR-10C**: The training and validation data sets are the same as in CIFAR-10. The test set consists of 19 different corrupted versions of the CIFAR-10 validation set.

For each dataset, we train deep convolutional networks with different feature-normalization strategies that are combinations of three basic elements:

1. **Normalization scheme**, which can be non-adaptive—as in traditional batch normalization where the statistics are estimated during training and then fixed—or adaptive—as in instance normalization or feature response normalization where the statistics are re-estimated at test time.

2. **Averaging scheme**, which can be instance-based—as in instance normalization or feature response normalization where the filter responses are averaged over a single data point—or batch-based—as in batch normalization where batches with several data points are used.

3. **Statistics**, which are either the mean and standard deviation—as in batch and instance normalization or feature response normalization—of the mean square—as in feature response normalization.

We apply all possible combinations of these elements, except for instance-based averaging and non-adaptive normalization because non-adaptive normalization requires stable estimates of the statistics.

The neural networks used in our experiments are based on standard convolutional architectures for classification. For the JIGSAWS and StrokeRehab datasets, we use a variation of DenseNet (Huang et al., 2017) with 11 dense blocks each consisting of four sets of convolutions, ReLU nonlinearity.
and a normalization layers. For CIFAR-10, CIFAR-10C and Fashion-MNIST, we use VGG-19 (Simonyan & Zisserman, 2014), Resnet-50 (He et al., 2016) and DenseNet-121 (Huang et al., 2017) architectures. We provide additional details about the architectures in section B. All these architectures are modified to apply different normalization strategies to the intermediate features.

For the sensor datasets (JIGSAWS and StrokeRehab) we perform classification on windows of 200 time steps (acquisition frequency 30 Hz for JIGSAWS and 100 Hz for StrokeRehab). We use overlapping windows with a stride of 25 time steps for training. The windows in the validation and test sets are non-overlapping. To remove spurious offsets caused by sensor calibration, we center and normalize the sensor data per dimension. For CIFAR-10 and CIFAR-10C, we center and normalize the RGB channels using channel mean and standard deviations computed from the CIFAR-10 training set. We use the Adam optimizer (Kingma & Ba, 2014) to train all models varying the learning rate when the validation accuracy plateaus (see section B). We apply early stopping based on the validation set.

Table 2. Results of the computational experiments described in Section 6 on the CIFAR-10C dataset for several different architectures. The entries in the table report the drop in classification performance of the networks relative to their performance on CIFAR-10. The lower the drop in accuracy, the better the relative performance of the network. Results are reported for different normalization strategies: non-adaptive or adaptive, instance-based or batch-based, and different statistics. For the statistics, \( \mu \) denotes the mean, \( \sigma \) denotes the standard deviation, and \( \nu^2 \) denotes the mean square. For all architectures, adaptive normalization achieves around 10% improvement over non-adaptive normalization.

![Figure 4. Scatterplot of classification accuracy and accuracy of decoding the value of the relevant extraneous variable (as a proxy for invariance). For each dataset, the classification accuracy is higher the lower the decoding accuracy, with the exception of the StrokeRehab dataset, where the features normalized non-adaptively result in very poor classification and decoding accuracy.](image)

7. Results

Table 1 reports the results of the computational experiments described in Section 6. For CIFAR-10 and Fashion-MNIST, the non-adaptive batch-based normalization schemes outperform all others, by small margins. In sharp contrast, for the datasets dependent on extraneous variables, adaptive normalization schemes outperform non-adaptive schemes by around 10% in JIGSAWS, 15% in StrokeRehab and 10% in CIFAR-10C. Instance-based adaptive normalization performs around 7% better than batch-based adaptive normalization in JIGSAWS, whereas both adaptive schemes perform similarly in StrokeRehab and CIFAR-10C. This is consistent with Figures 2 and 3, where batch-based adaptive normalization is not as successful for JIGSAWS as for the two other datasets. This could indicate the presence of other extraneous variables, apart from the surgeon identity. Finally, normalization based on mean and standard deviation, and on mean square perform similarly in most cases.

Table 2 shows the accuracy of the different normalization schemes on CIFAR-10C when compared to CIFAR-10 for different architectures (Table 1 only includes the best architecture in each case). The conclusions are essentially the same: for all architectures, adaptive normalization achieves
Robustness to Extraneous Variables Via Adaptive Feature Normalization

| Normalization scheme | Non-adaptive | Adaptive | Raw Signal |
|----------------------|--------------|----------|------------|
| Averaging scheme     | Batch based  | Instance based | Batch based |
| Statistics           | $\mu, \sigma$ | $\nu^2$ | $\mu, \sigma$ | $\nu^2$ | $\mu, \sigma$ | $\nu^2$ |
| JIGSAWS              | 94.53        | 95.80    | 51.15      | 61.19    | 49.11        | 95.41   | 86.5     | 98.4     |
| StrokeRehab          | 24.60        | 31.34    | 26.0       | 28.26    | 48.53        | 50.44   | 68.0     |          |
| CIFAR-10C            | 50.57        | 54.97    | 21.92      | 30.28    | 16.93        | 24.31   | 78.03    |          |

Table 3. Accuracy of decoding the value of the relevant extraneous variables from the features in the penultimate layer of convolutional networks trained using different normalization schemes. Networks trained with adaptive normalization schemes tend to have lower decoding accuracy as compared to the ones trained with non-adaptive normalization scheme for the CIFAR-10C and JIGSAWS datasets, which suggests higher invariance to the variations in the extraneous variable. For comparison, the accuracy of randomly guessing the value of the extraneous variable equals 12.5% for JIGSAWS, 3.125% for StrokeRehab, and 5.26% for CIFAR-10C.

around 10% improvement over non-adaptive normalization. Batch-based adaptive normalization slightly outperforms instance-based normalization. Normalization based on mean and standard deviation, and on mean square again perform similarly.

The outcomes of our experiments confirm our hypothesis for our datasets of interest: correcting the mismatch in feature statistics produced by variations in extraneous variables systematically results in significantly higher classification accuracy.

8. Invariance Analysis

The ultimate goal when designing machine-learning methods that are robust to variations in extraneous variables is to learn features that are (1) useful for the task of interest, and (2) as invariant as possible with respect to these variables. In this section, we study to what extent the features learned by convolutional neural networks with different normalization schemes are invariant to extraneous variables for our three datasets of interest. Our strategy is to train small neural networks to decode the value of the extraneous variable from the features at the penultimate layer of the networks (further details about these networks are provided in section C). Invariance is then quantified by evaluating the decoding accuracy achieved from features corresponding to held-out data. The lower this accuracy, the more invariant the network is to the extraneous variable.

Table 3 shows the results. In general, models trained with adaptive normalization seem to be more invariant to the extraneous features. Figure 4 shows a scatterplot of classification accuracy and decoding accuracy (as a proxy for invariance). For each dataset, the classification accuracy is higher the lower the decoding accuracy, with the exception of the StrokeRehab dataset, where the features normalized non-adaptively result in very poor classification and decoding accuracy. It is worth emphasizing that adaptive normalization is designed to only ensure approximate invariance of the first and second-order statistics of the features. Our results suggest that the resulting features tend to be more invariant to extraneous variables for the JIGSAW and CIFAR-10C datasets. It is worth noting that several works (Ulyanov et al., 2016; Huang & Belongie, 2017) have provided evidence that instance-based adaptive normalization boosts invariance to changes in image style, in the context of style transfer.

9. Conclusion

Designing robust methodology capable of performing successfully in changing environments is a fundamental goal in modern machine learning. In this work, we show that when processing data that is heavily dependent on extraneous variables, one should— in the words of Bruce Lee— be like water and adapt to the changing environment. More concretely, we demonstrate that standard non-adaptive feature normalization fails to correctly normalize the features of convolutional neural networks on held-out data where extraneous variables take values not seen during training. This can be remedied by normalizing adaptively, with statistics computed from the held-out data itself. In experiments with three different datasets affected by extraneous variables, switching to adaptive normalization consistently results in significant increases in classification accuracy. In addition, for two of the datasets, adaptive normalization seems to produce an increased invariance to the extraneous variable (in the sense that it becomes more challenging to decode the variable from the learned features). A challenge for future research is to further boost invariance to extraneous variables, even when the values of those variables are not known beforehand.
Acknowledgements

We would like to thank Prof. Kyunghyun Cho for useful discussions. We would also like to thank the volunteers who contributed to label the StrokeRehab dataset: Ronak Trivedi, Adisa Velovic, Sanya Rastogi, Candace Cameron, Sirajul Islam, Bria Bartsch, and Courtney Nilson. This research was supported by NIH grants R01 LM013316 (AK, CFG, HMS) and K02 NS104207 (HMS).

References

Alter, O., Brown, P. O., and Botstein, D. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.

Ballé, J., Laparra, V., and Simoncelli, E. P. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*, 2015.

Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Barndorff-Nielsen, O. On a formula for the distribution of the maximum likelihood estimator. *Biometrika*, 70(2):343–365, 1983.

Basu, D. On the elimination of nuisance parameters. *Journal of the American Statistical Association*, pp. 355–366, 1977.

Berger, J. O., Liseo, B., and Wolpert, R. L. Integrated likelihood methods for eliminating nuisance parameters. *Statistical Science*, 14(1):1–28, 1999.

Box, G. E., Hunter, J. S., and Hunter, W. G. *Statistics for experimenters: design, innovation, and discovery*, volume 2. Wiley-Interscience New York, 2005.

Carandini, M. and Heeger, D. J. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51, 2012.

Cox, D. R. and Reid, N. Parameter orthogonality and approximate conditional inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–39, 1987.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Gao, Y., Vedula, S. S., Reiley, C. E., Ahmidi, N., Varadarajan, B., Lin, H. C., Tao, L., Zappella, L., Béjar, B., Yuh, D. D., et al. Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling. In *MICCAI Workshop: M2CAI*, volume 3, pp. 3, 2014.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Huang, X. and Belongie, S. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kim, K.-Y., Kim, S. H., Ki, D. H., Jeong, J., Jeong, H. J., Jeung, H.-C., Chung, H. C., and Rha, S. Y. An attempt for combining microarray data sets by adjusting gene expressions. *Cancer research and treatment: official journal of Korean Cancer Association*, 39(2):74, 2007.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lazar, C., Meganck, S., Taminau, J., Steenhoff, D., Coletta, A., Molter, C., Weiss-Solís, D. Y., Duque, R., Bersini, H., and Nowé, A. Batch effect removal methods for microarray gene expression data integration: a survey. *Briefings in bioinformatics*, 14(4):469–490, 2012.

Leek, J. T. and Storey, J. D. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS genetics*, 3(9):e161, 2007.

Li, C. and Wong, W. H. Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *Proceedings of the National Academy of Sciences*, 98(1):31–36, 2001.
A. Additional results

A.1. JIGSAWS

In this section we report additional results for the surgical-activity dataset (JIGSAWS). The extraneous variable of interest for these data is the identity of the surgeon. There are eight surgeons in the dataset. We report the results of training on six, using one for validation, and then testing on the remaining one for six different random splits. The result for all the splits are reported in Table 4. They confirm the result reported in the main paper (which is the mean of all the splits): instance-based adaptive normalization consistently outperforms all other normalization schemes for these data.

A.2. Invariance Analysis

In section 8, we analyze to what extent features learned by different models are invariant to extraneous variables, by trying to decode the extraneous variables from the features. In this section, we provide additional results evaluating invariance in different layers of the networks. Tables 5, 6 and 7 report the results. We observe two patterns: 1) networks trained with instance-based adaptive normalization learn features that are more invariant to extraneous variables 2) the features from deeper layers of the network are more invariant to extraneous variables.

B. Architectural design and training procedure

B.1. StrokeRehab (Section 5.2) and JIGSAWS (Section 5.1)

In this section we describe the modified DenseNet architecture used that we apply on the JIGSAWS and StrokeRehab datasets. The architecture maps sequences of multivariate sensor data to sequences of predictions. The input is a two-dimensional array, where the first dimension corresponds to the different sensor variables (76 for JIGSAWS and 78 for StrokeRehab) and second dimension corresponds to the temporal dimension (200 time steps for both datasets). In the
Table 4. Results of the computational experiments described in section 6 on different splits of the JIGSAWS dataset. The entries in the table indicate the classification performance for different normalization strategies: non-adaptive or adaptive, instance-based or batch-based, and different statistics. For the statistics, $\mu$ denotes the mean, $\sigma$ denotes the standard deviation, and $\nu^2$ denotes the mean square. For all splits, instance-based adaptive normalization outperforms all other normalization schemes.

| Averaging scheme | Normalization scheme | Split - 1 | Split - 2 | Split - 3 | Split - 4 | Split - 5 | Split - 6 |
|------------------|----------------------|----------|----------|----------|----------|----------|----------|
|                  | Batch based          | 66.29    | 56.91    | 70.79    | 67.93    | 67.59    | 73.82    |
|                  | Instance based       | 75.3     | 58.64    | 73.6     | 76.15    | 72.21    | 80.38    |
|                  | Batch based          | 64.61    | 54.63    | 66.19    | 67.79    | 64.98    | 71.41    |
|                  |                      | 75.8     | 58.39    | 65.31    | 76.49    | 70.15    | 77.61    |
|                  |                      | 68.08    | 54.08    | 64.33    | 67.93    | 70.16    | 67.55    |
|                  |                      | 63.78    | 46.42    | 64.16    | 54.08    | 70.42    | 66.24    |

Table 4. Results of the computational experiments described in section 6 on different splits of the JIGSAWS dataset. The entries in the table indicate the classification performance for different normalization strategies: non-adaptive or adaptive, instance-based or batch-based, and different statistics. For the statistics, $\mu$ denotes the mean, $\sigma$ denotes the standard deviation, and $\nu^2$ denotes the mean square. For all splits, instance-based adaptive normalization outperforms all other normalization schemes.

To create the versions of network with other normalization layers, we just swapped the batch norm layer in the original network with the normalization layer of interest.

C. Architectural design and training procedure for invariance analysis

To perform the invariance analysis in section 8, we use a small neural network to decode extraneous variables from features of networks trained with different normalization schemes. The data to train the decoding network was created by pooling together the training, validation, and test sets to obtain the features. Then this set was separated randomly into a train (60%), validation (20%), and test (20%) set to train and evaluate the decoding network.

The architecture selected for the decoding network depended on the different datasets. All networks are trained using cross entropy.

- **JIGSAWS** and **StrokeRehab**: The intermediate layers considered in Tables 4 and 6 have dimensions $128 \times 200$. The network has four convolutional layers with each with 64 filters, followed by 3 fully connected layers.

- **CIFAR10-C**: Layers 10 and 40, considered in Table 7, have dimensions $256 \times 32 \times 32$ and $1024 \times 8 \times 8$ respectively. The network has two convolutional layers, followed by three fully connected layers. For the penultimate layer (2048 dimensional), we used a fully connected network with 5 layers.

We trained all the networks using a cross entropy loss with the Adam optimizer. The learning rate was decreased if the...
Robustness to Extraneous Variables Via Adaptive Feature Normalization

| Normalization scheme | Non-adaptive | Adaptive |
|----------------------|--------------|----------|
| Averaging scheme     | Batch based  | Instance based | Batch based |
| Statistics           | $\mu, \sigma$ $\nu^2$ $\mu, \sigma$ $\nu^2$ $\mu, \sigma$ $\nu^2$ |
| Layer - 24           | 97.54 99.00 | 80.90 83.02 | 96.69 95.00 |
| Layer - 28           | 96.22 97.40 | 68.22 77.93 | 93.77 93.25 |
| Layer - 32           | 95.47 96.88 | 65.58 80.95 | 92.36 91.65 |
| Layer - 36           | 96.27 97.21 | 61.99 69.54 | 92.03 89.53 |
| Layer - 40           | 95.00 96.88 | 57.37 58.93 | 83.21 80.66 |
| Layer - 44 (Penultimate Layer) | 94.53 95.80 | 51.15 61.19 | 91.41 86.5 |

Table 5. Accuracy of decoding the identity of each surgeon in the JIGSAWS dataset from features at different layers of convolutional networks trained using different normalization schemes. Networks trained with adaptive normalization schemes tend to have lower decoding accuracy than the ones trained with non-adaptive normalization scheme, which suggests higher invariance to the extraneous variable. Instance-based normalization provides significantly more invariance (and also better performance as shown in Table 4). Additionally, features from deeper layers show more invariance. For comparison, the accuracy of randomly guessing the value of the extraneous variable equals 12.5% for this dataset.

| Normalization scheme | Non-adaptive | Adaptive |
|----------------------|--------------|----------|
| Averaging scheme     | Batch based  | Instance based | Batch based |
| Statistics           | $\mu, \sigma$ $\nu^2$ $\mu, \sigma$ $\nu^2$ $\mu, \sigma$ $\nu^2$ |
| Layer - 24           | 49.84 54.93 | 33.24 49.61 | 47.65 54.20 |
| Layer - 36           | 32.41 36.60 | 27.25 33.19 | 47.73 46.48 |
| Layer - 44 (Penultimate Layer) | 24.60 31.34 | 26.0 28.26 | 48.53 50.44 |

Table 6. Accuracy of decoding the identity of each patient for the StrokeRehab dataset from the features at different layers of convolutional networks trained using different normalization schemes. Networks trained with instance-based adaptive normalization have lower decoding accuracy than the other normalization schemes, which suggests higher invariance to the extraneous variable. Additionally, features from deeper layers show more invariance to the extraneous variable. For comparison, the accuracy of randomly guessing the value of the extraneous variable equals 3.125% for this dataset.

accuracy on validation set plateaued. We employed early stopping using the accuracy on the validation set to select our final model.

### D. StrokeRehab dataset details

The StrokeRehab dataset contains data from 32 stroke patients performing nine activities of daily living: feeding, drinking, washing face, brushing teeth, donning glasses, applying deodorant, combing hair, moving objects in radial manner and moving objects on a shelf. Each activity was repeated five times. The approximate duration of activity ranged from 15 secs (for the applying deodorant activity) to 200 secs (for the feeding activity).
Robustness to Extraneous Variables Via Adaptive Feature Normalization

| Normalization scheme | Non-adaptive | Adaptive |
|-----------------------|--------------|----------|
| Averaging scheme      | Batch based  | Instance based | Batch based |
| Statistics            | $\mu, \sigma$ | $\nu^2$ | $\mu, \sigma$ | $\nu^2$ | $\mu, \sigma$ | $\nu^2$ |
| Layer - 10            | 55.56 | 57.463 | 37.8 | 43.28 | 51.23 | 52.93 |
| Layer - 40            | 48.86 | 45.91 | 34.32 | 31.86 | 26.23 | 20.72 |
| Layer - 49 (Penultimate Layer) | 50.57 | 54.97 | 21.92 | 30.28 | 16.93 | 24.31 |

Table 7. Accuracy of decoding the type of corruption for CIFAR-10C dataset from the features of the various layers of convolutional networks trained using different normalization schemes. Networks trained with adaptive normalization schemes tend to have lower decoding accuracy than those trained with non-adaptive normalization schemes, which suggests higher invariance to the extraneous variable. Additionally, features from deeper layers show more invariance to the extraneous variable. For comparison, the accuracy of randomly guessing the value of the extraneous variable equals 5.26% for this dataset.

Figure 5. DenseNet architecture for JIGSAWS and StrokeRehab dataset as described in Section B.1.