Team 34: Evaluating the Robustness of Deep Learning Models

Pasawee Wirojwatanakul (Jeff)
pwirojwatanakul@ucsd.edu

Zian Wang
ziw080@ucsd.edu

Yizheng Yu
yiy003@ucsd.edu

Abstract

In this work, we investigated the robustness of deep learning models under covariate shifts; in particular, we perturbed the test images with Gaussian blur and adjusted the test images' brightness. Our experiments showed that data augmentation and pre-training, both supervised and self-supervised methods, are effective ways to increase the robustness of deep learning models to such noises. The code for our project can be seen at: https://github.com/jeffwiroj/ece228_final

1 Introduction

The supervised machine learning problem assumes we have a training dataset $D_{tr} = \{(x_1, y_1), ...(x_n, y_n)\}$, where (x, y) are drawn from some unknown distribution $P(X, Y)$, in which we wish to learn a function, f, such that given a new test point $(x, y) \sim P$, $f(x) = y$. The crucial assumption we make here is that the test point is drawn from the **same** distribution as the training distribution. This assumption is rather strong and may not hold true in many real-world applications; as an example, we may train a disease classifier from one hospital and deploy it another, in which the distribution of the patient, for example, age, may be different. Without any assumptions, we will not be able to learn a classifier that is robust to shifts in the test distribution, which is particularly worrisome for decision making systems. While there are many types of distribution shifts, in this project, we focus on covariate shifts, where the distribution of the feature changes; $P_{tr}(X) \neq P_{te}(X)$ [Zhang et al., 2021]. Here we assume that that the label space remains unchanged.

For this project, we focused on evaluating the robustness of deep learning models, in particular, ResNet18 and ResNet50 on two datasets, pathMNIST and Oxford-IIT Pets. To simulate the distribution shift, we injected noise into the test set and compared the accuracy of the models on the clean test-set vs. the corrupted test-set. We also investigated whether the predictive probability of ResNet is reflective of its likelihood to be correct. Our contributions can be summarized as follows:

- We investigated how data augmentations affect model robustness.
- We investigated how pre-training, regardless of the supervised or self-supervised approach, affects model robustness.
- We studied the calibration of ResNet under different noise and pre-training methods.

2 Related Works

Hendrycks and Dietterich created a benchmark to show that deep learning architectures are not robust to small changes in the image space. Previous works showed that pretraining, both in a supervised and self-supervised setting, can help improve model robustness under certain noise corruptions Hendrycks et al. [2019a,b]. Furthermore, image augmentation techniques could also be employed to increase the robustness of deep learning models Zhang et al. [2017], Hendrycks et al. [2019c].

3 Problem Formulation

In this project, we look at how ResNet performs when the test image distribution differs from the train image distribution; $P_{te}(X) \neq P_{tr}(X)$. We only consider the case when the brightness of the test image has been adjusted and when the test image has been blurred. Note, these perturbations are only added in the test set. Furthermore, we aim to see if the ResNet makes calibrated prediction under this shift; for example, will the predictive confidence drop when the accuracy drops. The goal of measuring calibration is to see if the model confidence, the max of the softmax layer, is indicative of the model's likelihood to be correct, see calibration error section: eq 1.

4 Methodology

We aim to improve model robustness from two angles **data augmentation** and **pre-training**.

4.1 MixUp

MixUp was created to help with neural networks' tendency to memorize training data and their sensitivity to adversarial examples. The idea behind MixUp is to train on linear combinations of features and labels, in which the author showed, can be effective at regularizing the model. MixUp creates the new training pairs as follows:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \;\; \tilde{y} = \lambda y_i + (1 - \lambda)y_j$$

, where $(x_i, y_i), (x_j, y_j)$, are two randomly drawn samples from the training set, and $\lambda \in (0, 1)$ and is drawn from the Beta distribution Zhang et al. [2017].

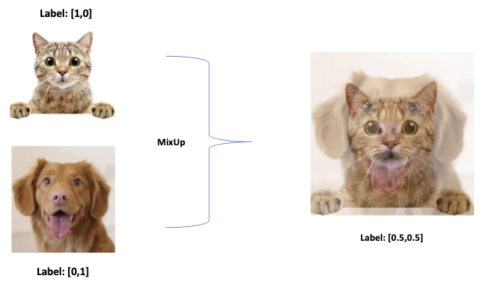

Figure 1: Below we showed an instance of mixup, where $\lambda = 0.5$

4.2 Transfer Learning

Transfer Learning is the process in which we pretrain a neural network on a larger dataset and transferring the representations learned by the network to other tasks with limited training data. Pretraining can be done in either a supervised or self/un-supervised way. In self-supervised approaches, we only have access to the features/images and not the labels. We can still utilize the images alone to learn strong representations for the backbone network. Then, for our target task, we can just attach a fully connected layer on top of this backbone network and either train just the fully connected component, or the whole network.

4.3 Barlow Twins

Barlow Twin is a self-supervised algorithm used to pretrain the backbone of a deep learning classifier, for example, ResNet50 [Zbontar et al., 2021]. During the pretraining stage, Barlow Twin only has access to the images to train the backbone. Barlow Twin operates as a joint-embedding architecture. For a given image x, we start by randomly augmenting an image two times to obtain y_a, y_b, in which we try to make the the representations of $z_a = f(y_a)$ and $z_b = f(y_b)$ to be close, where f is a trainable model, in this case, ResNet. In Barlow Twins, to make z_a and z_b close, we try to make the cross-correlation of the two vectors to be as close to the identity matrix as possible. By doing so, we make features encoded in z_a and z_b at the same dimension highly correlated, which makes the representation obtained invariant to data-augmentations, see figure 2. Intuitively, whether we add in random color jitter or blur to a picture or not should not change the representation for the picture.

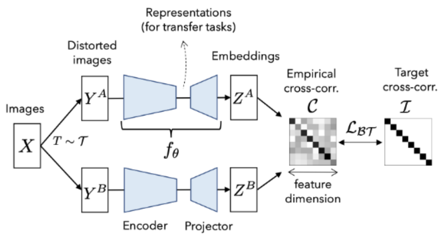

Figure 2: Barlow Twins Algorithm [Zbontar et al., 2021]

We should expect pre-trained networks to be more robust because CNNs learn general feature extractors in the earlier layers, which is what we used to transfer to our target task. Furthermore, the SSL objective makes the network's representation invariant to noises.

4.4 Evaluation Metrics

4.4.1 Accuracy

The accuracy metric can be simply calculated as the number of correct predictions over the total number of samples.

4.4.2 RMSE Calibration Error

Firstly, we sort the predictions based on the model's confidence, which is defined as the probability of the prediction. In our case, it will be the max of the softmax layer from ResNet since the prediction coincides with the argmax of the softmax layer. Afterwards, we separate the predictions into N equally spaced bins, for example, all the predictions with confidence over 80% can be sent to N-th bin. Then, the RMSE Calibration Error is defined as:

$$\sqrt{\sum_{i=1}^{N} b_i (p_i - c_i)^2} \tag{1}$$

, where b_i is the fraction of data in bin i, p_i is the top-1 accuracy of predictions in bin i, and c_i is the average confidence of bin i. Intuitively, calibration metrics in general aims to see whether the predictive probability is indicative of the model's likelihood to be correct. This is helpful because if our model is well-calibrated we can use the confidence to gauge whether the model is likely to make a mistake or not. Well calibrated models thrive in scenarios where decisions have high consequences since we can defer the predictions with low confidence to human experts to make the decisions instead.

Note, there are many calibration errors, including the Expected Calibration Error and Maximum Calibration Error, here we followed the works of Hendrycks et al. [2019b] as used the RMSE Calibration Error.

5 Dataset

5.1 pathMNIST

This dataset contains images of colon pathology. The goal for using this dataset is to see whether a network pretrained on ImageNet can be useful for pathology classification, and whether it will be more robust than training from scratch.

5.2 Oxford-IIT Pets

This dataset contains images of pets, which is "closer" to ImageNet than the pathMNIST. Here our goal is to evaluate the robustness of deep learning models using two types of pretraining on ImageNet, supervised and self-supervised pretraining. For the self-supervised pretraining, we chose the Barlow Twins Method.

Both of these datasets already have the train/val/test data splitted.

6 Experiments

We tuned the learning rate, weight decay and experimented with learning rate schedulers. Note, all our models can be viewed as a composition of a backbone network, which comprises Convolution layers, and a fully-connected (fc) layer. For the pretrained setting, we have a different set of learning rate for the backbone and fc part. We may want to do this because the backbone has already been trained well to detect features, so we do not want to change the parameters of the backbone part drastically. Additionally, for the pretrained setting, we followed Chen et al. and considered freezing the batch-norm parameters vs. not freezing the batch-norm parameters. Oftentimes, the batch-norm statistics have been computed on a significantly larger batch size, so we may not want to re-train the batch-norm parameters if we do not have enough computing resources. We used the AdamW optimizer for all our experiments. For the whole set of configurations, please take a look at our github page. Finally, when training from scratch, we normalize the input by the mean and standard deviation of the images of the training set in our dataset. When utilizing transfer learning, we normalize the input by the mean and standard deviation of the images from the training set of ImageNet.

6.1 pathMNIST Dataset

Since this dataset is small, ResNet can easily overfit the training set, so we chose to work with the smallest ResNet model, ResNet18. We considered two settings, training ResNet18 from scratch, and training ResNet18 using weights that were pretrained on the ImageNet dataset, in a supervised fashion. First, we trained the model from scratch without any data augmentations. Then we trained the model with basic augmentations, combined with mixup. Our basic augmentations involve random horizontal/vertical flip and random rotation. We then repeated this process for the pretrained network. For the pretrained setting, we found that having a smaller learning rate for the backbone is beneficial; in our case, the backbone learning rate is 10 times smaller than the fc learning rate. In addition, freezing the batch-norm parameters also lead to additional gains. In the result section, we report the best configurations.

6.2 Oxford-IIT Pets

For this dataset, we used ResNet50 and considered 3 different weight initialization settings; firstly, we trained from scratch. Next, we considered 2 different types of pre-training, standard supervised pre-training on ImageNet, and Barlow Twins self-supervised pretraining on Imagenet. Apart from the weight initialization, all the models have the same training configurations, including the weight decay, learning rate, and data augmentations, which include MixUp and flip and rotations. Since we do not have the computational resources to pretrain Barlow Twins, we used the pretrain Barlow Twins weight from: https://github.com/facebookresearch/barlowtwins.

7 Results

We consider two sets of corruptions, Gaussian Blur and Brightness, which are taken based on the works of Hendrycks and Dietterich. We will refer to basic augmentations as the combination of random flipping and rotations.

7.1 pathMNIST

For pathMNIST, we chose 4 different brightness factors from [0.8, 0.9, 1.1, 1.2], where 0.8 means the adjusted picture in the test set is 20% darker than the original picture and 1.2 means the adjusted picture is 20% brighter; see the pytorch documentation for more information. For Gaussian Blur, we fixed the kernel size to 5x5, and chose 4 different σ = [0.3,0.5,0.7,0.9], which corresponds to the standard deviation used to draw the kernel to perform blurring.

Accuracy (%)			
Model	Clean	Brightness	Gaussian Blur
Basic	82.3	36.4	67.5
MixUp + Aug	85.9	48.8	68.1
MixUp + Aug + PT	**87.6**	**50.3**	**74.3**

Table 1: Result of ResNet18 on pathMNIST dataset. Basic refers to no data augmentations. MixUp + Aug refers to using MixUp and basic data augmentations. MixUp + Aug + PT refers to using pretrained weights and MixUp and basic data augmentations. For the noises, we report the mean accuracy averaged across the different perturbation levels; brightness factor and σ.

RMSE Calibration Error			
Model	Clean	Brightness	Gaussian Blur
Basic	0.145	0.603	0.278
MixUp + Aug	0.040	0.302	0.168
MixUp + Aug + PT	**0.033**	**0.299**	**0.111**

Table 2: Result of the RMSE Calibration Error on PathMNIST dataset. We report the score averaged across the pertubation strengths.

Accuracy (%)				
Model	f=0.8	f=0.9	f=1.1	f=1.2
Basic	34.7	58.8	31.3	20.8
MixUp + Aug	46.2	65.9	**55.0**	28.1
MixUp + Aug + PT	**46.6**	**72.1**	52.9	**29.4**

Table 3: Result on pathMNIST dataset with perturbed brightness factor, where f denotes the factor.

Accuracy (%)				
Model	σ=0.3	σ=0.5	σ=0.7	σ=0.9
Basic	82.5	73.8	59.6	54.2
MixUp + Aug	85.8	76.0	60.5	50.1
MixUp + Aug + PT	**87.9**	**83.0**	**69.4**	**57.1**

Table 4: Result on pathMNIST dataset with perturbed Gaussian blur, where σ denotes the standard deviation.

From table 1, we see that MixUp and basic data augmentations can help increase the accuracy of the model by up to 3.6% on the clean dataset. By using a pretrained network, we get an additional 1.7% gain. Furthermore, we see that data augmentation and pre-training are effective in making the network more robust to covariate shifts. Training with data augmentation and MixUp makes the

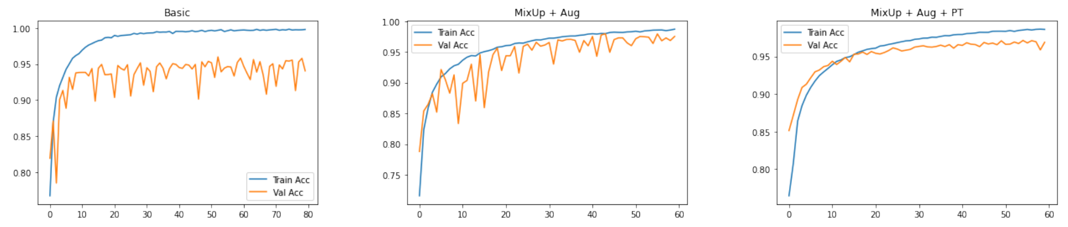

Figure 3: A figure to show the Train vs Val Accuracy for the three types of training mentioned in table 1

network more robust to the noises in the test set at almost all corruption level, with the exception of Gaussian Blur $\sigma = 0.9$ case, where the basic setup outperforms MixUp + augmentation, 54.2 vs 50.1. With additonal pre-training, we see that the model achieves higher accuracy on both the clean and perturbed dataset at all corruption strengths, compared to basic setup. Furthermore, for this dataset, MixUp and pretraining help make the model more calibrated. In general, the RMSE Calibration Error is lower when we used MixUp and pretrained weights for this dataset.

It is worth noting that in general, the models are highly sensitive to pertubations in the test set for both noises.

From figure 3 we observed over-fitting issues for the basic model's case. We see that MixUp and basic augmentation work well in reducing the discrepancy between the train and validation accuracy; however, the validation accuracy is very noisy. With a pretrained network, the validation accuracy goes up in a much smoother fashion.

7.2 Oxford-IIT Pets

For the Pets dataset, we chose 4 different brightness factors from [0.6, 0.8, 1.2, 1.4]. For Gaussian Blur, we fixed the kernel size to 5x5, and chose 4 different σ = [0.3,0.5,0.7,0.9]. Note we chose a larger brightness factor here because it is harder to make the accuracy go down in this dataset.

Accuracy (%)			
Model	Clean	Brightness	Gaussian Blur
No Pretraining	73.7	67.5	66.4
Supervised	**96.9**	**96.0**	**95.0**
Self-Supervised	93.6	92.4	88.4

Table 5: Result of ResNet50 on Oxford-IIT Pets dataset. Supervised refers to standard pre-training on ImageNet. Self-Supervised refers to using Barlow Twins to pre-train the network on ImageNet. For the noises, we report the mean accuracy averaged across the different perturbation levels; brightness factor and σ.

RMSE			
Model	Clean	Brightness	Gaussian Blur
No Pretraining	**0.055**	**0.050**	**0.060**
Supervised	0.155	0.149	0.158
Self-Supervised	0.218	0.207	0.195

Table 6: RMSE Calibration Error result of ResNet50 on Oxford-IIT Pets dataset.

	Accuracy (%)			
Model	f=0.6	f=0.8	f=1.2	f=1.4
No Pretraining	66.2	72.7	68.6	62.4
Supervised	96.1	96.2	96.5	94.9
Self-Supervised	91.1	92.8	93.4	92.4

Table 7: Result on Oxford-IIT Pets dataset with perturbed brightness factor. f denotes the brightness factor

	Accuracy (%)			
Model	σ=0.3	σ=0.5	σ=0.7	σ=0.9
No Pretraining	73.6	71.3	64.4	56.4
Supervised	**96.3**	**96.3**	**94.3**	**93.0**
Self-Supervised	92.4	92.0	86.3	82.9

Table 8: Result on Oxford-IIT Pets dataset with perturbed Gaussian blur. σ denotes the standard deviation.

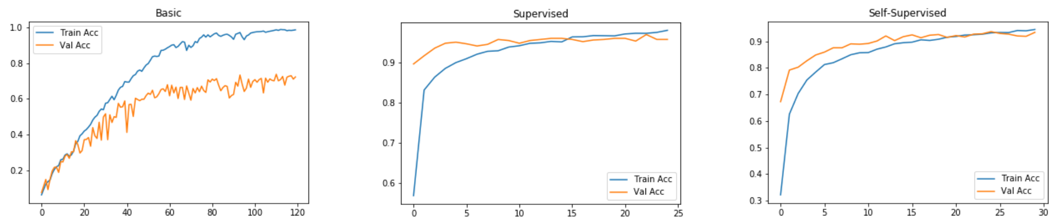

Figure 4: A figure to show the Train vs Val Accuracy for the three types of model mentioned in table 5.

We observed that no matter what pre-training method we used, we observed higher accuracy on both the clean and perturbed data-set for all corruption strengths by a significant margin. If accuracy is what you aim for, the results show that pre-training is a crucial component. For the clean dataset, we observe a gain of up to 23.2% when using pre-trained weights. For the perturbed dataset, Gaussian Blur case, we observe a gain of up to 30.6% in accuracy. Furthermore, we found that supervised pre-trained weight transfers better than self-supervised pre-trained weights for our particular case. It is worth noting that for the pre-trained case, we only trained the model for 25 epochs, and it already achieves a much higher score than training from scratch for 120 epochs.

Unlike the result of pathMNIST, pre-training does not lead to more calibrated models for this dataset. Similar to the result of pathMNIST, here, we see that using the basic setup, the loss is very noisy.

8 Discussion

We showed that pre-training can be an effective way to increase robustness to certain noises in the test set. Furthermore, it allows the network to train faster and in a less noisy fashion, see the training curves. This is most likely because unlike training from scratch, we are already in a good basin of the loss landscape so we do not have to perform that many gradient updates to reach an area with low loss. Furthermore, MixUp and data augmentations help with increasing model calibration. One intuition is without MixUp, we are training our models to confidently predict one class. During training, our target labels is one-hot encoded, so the logits of the network will be very high only for one class and close to zero for all the rest. With MixUp, we are smoothing the logits, making the model more calibrated when seeing data that is not seen in the training set.

One takeaway from this experiment is that if we have access to pre-training data, we should always pretrain. However, we should interpret the result with caution. As shown in the Pets dataset, sometimes pre-training may not lead to more calibrated models. This may be problematic since with a well-calibrated model we can, to a certain extent, tell when a model is likely to fail.

We also observed that deep learning models are more sensitive to pertubations in the medical image domain. We hypothesize this is due to the fact that medical images are collected in a very procedural way, in which slight perturbations can lead to large distribution shifts.

To conclude, in real-life scenarios, the training set and test set's distribution may not be the same. As we have shown, this can be problematic, especially for decision making systems where the decision has high consequences as in the healthcare domain. Although we have only studied covariate shifts, keep in mind that a label shift of the form $p_{tr}(Y|X) \neq p_{te}(Y|X)$ is also possible. When trying to deploy ML models, do so with caution.

For future works we can try testing our models agaisnt adversarial attacks like PGD and FGSM and try out newer data augmentation techniques and self-supervised algorithm. Furthermore, it would also be useful to study how particular model design choices affect calibration. Finally, our observations did not show that pretraining will increase model robustness. In the future, it will be useful to study why this is the case or whether we can find ways to make pre-training increase robustness.

Contributions

Every member contributed to the coding, discussion, writing of report and poster.

References

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations, 2019. URL `https://arxiv.org/abs/1903.12261`.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty, 2019a. URL `https://arxiv.org/abs/1901.09960`.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in Neural Information Processing Systems*, 32, 2019b.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019c.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. URL `https://arxiv.org/abs/2103.03230`.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2016. URL `https://arxiv.org/abs/1606.00915`.