Deep Trouble Report

Deep Trouble is a research project supervised by Lech Szymanski and David Eyers at the University of Otago. It explores potential security risk in the context of neural networks and transfer learning. Consider an unrealistically oversimplified, hypothetical scenario, in which a target neural network is trained, by transfer learning from a source neural network, to recognise faces for the purpose of controlling access to a building. Is it possible to embed a pattern in the source capable of “surviving” target training and later causing a strong response that may act as a “master key”? Things that have been explored including Adversarial attacks, black box attacks and backdoors in neural networks. The area I have chosen to explore would be if backdoors trained in the original model would be still prevalent in the transferred model (it is). Along with exploring the effectiveness of backdoors and how to remain effective.

The first thing that I did was to test if the backdoor works. For all these experiments I would train convolutional neural networks with backdoors which come from the python art.attacks library. These backdoors would apply a mark onto the image like the images shown below.

A dog wearing a sweater

Description automatically generated A dog standing on a leash

Description automatically generated

These marks would be anything from an image or strips of RGB. After implementing these marks on an image, I would then change the label of these images to something incorrect essentially associating the mark with another label despite the image. I would then transfer the CNN to another dataset without any poisoned images retrain on that dataset and then poison that dataset and see the results. In all these experiments I trained a model without a backdoor as well to compare the results to see if the change in performance were due to the backdoor or noise in the data.

First test: cifar10

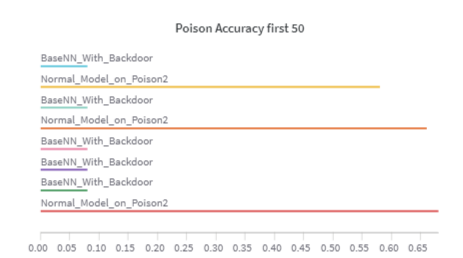
My very first test was training a dataset on the cifar10 dataset and poisoning some images. With the following poison shown below.

A green object with a blurry background

Description automatically generated with medium confidence

Here below are the results from the cifar10 dataset. A graph with numbers and text

Description automatically generated with medium confidence



We can see that there doesn’t seem to be a difference in overall accuracy in the model compared to the normal model ad the normal model performance on the poison remains strong but with the backdoor built in. We can see that the performance drops quite considerably. With this information, we can confirm that the poison works and start testing on other datasets.

Second Test: Cifar10 🡪 cats n dogs and 10 class imagenet

In this next test I proceed to transfer the cifar10 model and train it on cats n dogs and a 10 class ImageNet to test the theory of if the poison works. Note as this is the first experiment that I did. The result of the performance tends to be quite low as I wasn’t that good at making NN’s at the time. But the theory still applies.

Imagenet

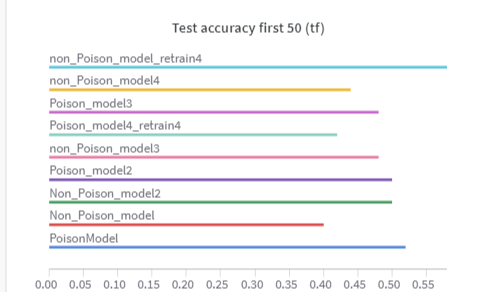
Snippet of one of the predictions

A number set up in a row

Description automatically generated

Different type of models such as retraining from 4 may cause the predictions to majority be other numbers

Performance of images without poison



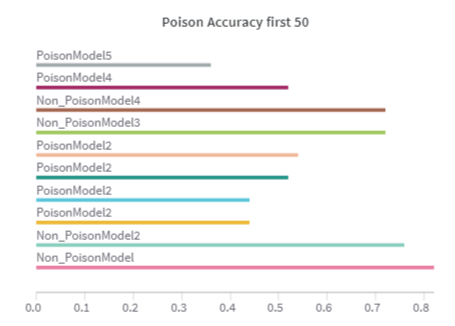
Performance of images with poison

A graph of text and numbers

Description automatically generated with medium confidence

Retrain 4 was taking the last 4 layers of the model and retraining instead of from the last conv layer.

Cats n dogs

A graph of a number of objects

Description automatically generated with medium confidence

One thing to notice would be that in the original cifar10 dataset the poisoned label was set to cats and in this case the predicted output was mostly 0 for cats

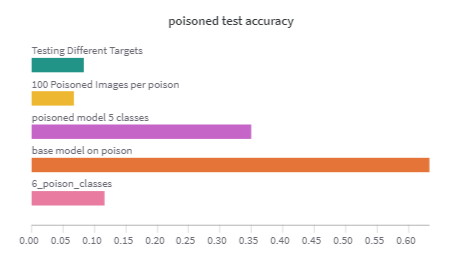
A screenshot of a computer

Description automatically generated

Cifar100 🡪 cifar10

After understanding that the poisoning does work, and the backdoor does have an a affect on transfer learning I wanted to see how multiple poisons work and how to maximize the decrease in accuracy in the transferred models. So, in the next test I trained a model on cifar100 and transferred it to cifar10 and tested a lot of different theories with this.

Original performance of cifar100 dataset



During the transfer learn we learn that there a lot of variables that affect the effectiveness of a poison. Including the poison itself (big, small, colour, location), target label in the cifar100 dataset, number of poisoned images in the train all influence the effectiveness. In the above image, 6 poison classes represents having 6 different backdoors, 100 poisoned means that there were 100 poison images for each class to train on and testing different targets were changing up the target classes.

An example of an effective poison



And ineffective poison

A screenshot of a graph

Description automatically generated

A part where I found interesting was shown below

A graph of a test

Description automatically generated with medium confidence

Where the poison was rather affective in the “different targets model” but not in the “non\_similar\_target\_classes” Which the difference was the in the different targets model line I chose classes that represented things that had similar features in both data sets eg lion 🡪 cat or truck to -> pickup truck. Where in this case the smile was the truck for the different targets model but it was bridge for the non\_similar\_target\_classes. Showing that the target class has the potential to be influential in determining the effectiveness of the poison.

Cats n dogs 🡪 upside down

To test the effectiveness of labels I chose to do another experiment. In this experiment I got a dataset of cats n dogs and added 2 poisons one to change the label to a cat and the other to change the label to a dog. I trained the model on this dataset and got to a 93-94% accuracy. I then transferred to a model that predicts if an image is upside down or not. I then use the poisoned images that it was trained on a changed the context to flipped or not flipped showing the results below. In the tables below, the accuracy represents the accuracy of the model on if it is flipped or not flipped. 0 for not flipped and 1 for flipped.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Where Axis = None | | | | |
|  | | | Poisoned Dataset | Base Dataset |
| Accuracy on Cats N Dogs Dataset | | | 93% | 94% |
| Accuracy On Upside Down | | | 85.86% | 86.6% |
| Accuracy on Poison Images without Poison | | | 84.35% | 93.4% |
| Dogs | Up right | Cat poison P=0 | 100% | 93.2% |
| Dog poison P=1 | 100% | 93.17% |
| Flipped | Cat poison P=0 | 3.2% | 97.2% |
| Dog poison P = 1 | 90% | 93.2% |
| Cats | Up right | Cat poison P = 0 | 100% | 89.6% |
| Dog poison P = 1 | 92.8% | 95.2% |
| Flipped | Cat poison P = 0 | 0.12% | 92.62% |
| Dog poison P = 1 | 100% | 87.2% |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Where Axis = 0 | | | | |
|  | | | Poisoned Dataset | Base Dataset |
| Accuracy on Cats N Dogs Dataset | | | 92.64% | 94.36% |
| Accuracy On test Upside Down | | | 78.6% | 80.68% |
| Accuracy on poisoned images without poison | | | 77% | 88.4% |
| Dogs | Up right | Cat poison | 100% | 86.8% |
| Dog poison | 100% | 85.94% |
| Flipped | Cat poison | 1.6% | 88.4% |
| Dog poison | 28.8% | 90.8% |
| Cats | Up right | Cat poison | 99.6% | 81.6% |
| Dog poison | 100% | 80% |
| Flipped | Cat poison | 1.2% | 89.24% |
| Dog poison | 29.6% | 90.80% |

Model where axis = 0 and model where axis = none means that the function I used np.flip had a axis variable that when was set to none looked like this

A dog wearing a plaid shirt

Description automatically generatedA blurry image of a person's head

Description automatically generated

While when it was set to 0 it looked like

A dog sitting on a blanket

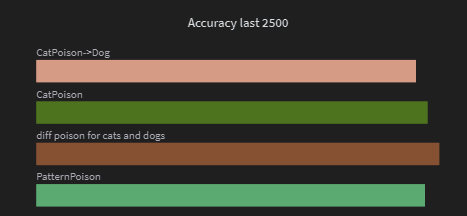
Description automatically generatedA dog lying on a bed

Description automatically generated

Overall, from the results, we can see that even after changing the context of the labels completely. The poison is still able to stick and affect the accuracy in some cases. One thing to note is that in some of these cases such as cat poisons we can see that the accuracy is either unusually high or unusually low. Which I predict to be the fault of the poison making it constantly predict 0 which in some cases is the correct label. Finally, We can deduct that the poisons stick to the images but cannot deny that it sticks to the labels.

Double Poison

To test if poison just changes the label to an incorrect one or if it increases the prediction of another. I ran test like this with multiple different poisons at the results. What I found was the accuracy on the poisoned data always remained at 99%-100% accuracy. Showing that it picks up the poison very well. But the accuracy of the non-poisoned cats and dogs would change based on various factors such as amount of poison. Where the pattern dropped the accuracy from 94 to 88 but when adding something such as a cat face as a poison it would drop the accuracy to 86%. For the diff poisons for cats and dogs test where a poison would only set cats to dogs and a poison would only set dogs to cats. Then I tested on putting the poison on the opposite animal. Where it displayed a 0% accuracy.



Showing the most effective poisons are something that are unique as the cats would drop the accuracy especially if you put the target label to dog.

Further Research Points

In the future what research could be done would be to explore more in dept in what makes \a poison/backdoor affective especially when in comes to transfer learning. I only really explored how poison correlates to the target label but there are other points that could potentially influence the strength of a poison.