Self-supervised Learning to Improve Robustness

COMS 6998-003.2021-Spring.report
Valentine d'Hauteville vd2256, William Gu xg2355

***Abstract*—Deep Neural Networks have shown to be very brittle to adversarial attacks. Many approaches have been proposed to make models more robust, but few offer insights about the effects of adversarial inputs on self-supervised learning tasks. In this work, we attempt to utilize the intrinsic labels from auxiliary self-supervised tasks to construct an attack-agnostic defense against adversarial attacks. Our hypothesis is that a perturbation that reduces the loss on a self-supervised task will partially cancel out with the effects of a possible adversarial perturbation targeted at maximizing loss on a supervised learning task such as classification. If this proves to be the case, then self-supervised learning would present itself as a promising new line of work towards developing attack agnostic robust models.**

I. Introduction

Deep networks yield strong performance on computer vision tasks, yet remain brittle to a wide body of adversarial attacks. These range from carefully designed imperceptible perturbations added to the input image to images alterations which occur naturally in the real word (such as noisy pixel values or translations and shifts). There has been an extensive body of research devoted to understanding reasons behind adversarial vulnerability and designing more robust models. Techniques such as adversarial training have been shown to successfully mitigate adversarial attacks, however, they are in general only robust against a small range of attack models. In this paper, we build on some of the work started by Chengzhi Mao to investigate whether naturally occurring labels present in the data can be leveraged via self-supervised learning to build a robust model that is attack agnostic. Indeed, when training our classifiers on auxiliary self-supervised tasks (e.g: image inpainting), we have found that these tasks can also be affected by adversarial inputs. One idea could then be to leverage these observed discrepancies to create a "reverse attack" designed to perturb the model towards producing the desired self-supervised task output. The obtained "re-calibrated" model could then used on the main supervised task with the hope that it would now be more resistant to adversarial attacks. For a summary diagram of this approach please refer to figure 1. In this paper, we test our approach on two well know self-supervised learning tasks: image inpainting and jigsaw puzzle. Our code can be found in the following (private) repository.

Before proceeding further, we would like to point out some of the intuitions which motivated us to explore training an auxiliary self-supervised task as a way to mitigate adversarial inputs:

- **Adversarial attacks change hidden layer representations as well.** From [1] and [2], we've seen that adversarial inputs affect the hidden layers in addition to the output. In fact, existing adversarial defense approaches have managed to improve robustness by controlling or stabilizing these hidden representations. In this work, we try to use unsupervised learning tasks to shape the model's hidden features.
- **Multiple tasks can share hidden layers and multitask learning has shown to improve robustness.** Previous work by Chengzhi Mao in [3] has shown that training multiple tasks concurrently using common hidden layers can lead to higher robust accuracy without compromising on clean accuracy. This present work builds on that idea to concurrently train a model on a main classification tasks and an auxiliary self supervised task.
- **Labels are intrinsically generated from the data in self-supervised tasks.** Since the correct output remains known regardless of whether an input image is adversarially perturbed, a second perturbation could then be constructed to "push" the output of a self-supervised towards a smaller loss. Our hypothesis is that this could in turn undo the effects of the adversarial perturbation on the primary classification task.

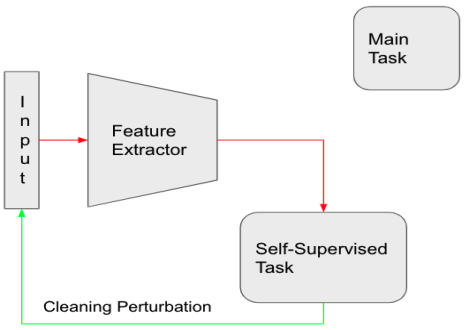

Fig. 1. Summary of our approach

II. Related Work

- **Self-supervised Learning** utilizes labels generated intrinsically from the data. As a result, unlike in the traditional case, when an adversarial example affects the output of a self-supervised task, the correct output is still known. We aim to utilize this property to improve the adversarial robustness of other networks via multitask learning.
- **Multitask Learning** allows for multiple tasks to share parameters. These shared layers serve as a backbone

feature extractor, after which task-specific decoders can be used to generate outputs specific tasks. Previous research [3] into the effects of multitask learning on adversarial robustness has given theoretical analysis as to why it is difficult for an adversary to simultaneously attack multiple objectives and has empirically shown that multitask learning can be utilized to improve both clean and robust performance.

- **Adversarial examples** introduce small perturbations to inputs to fool target models. Past works into adversarial robustness have shown that these perturbations affect not only the outputs of the attacked models, but the hidden representations as well [4]. Our work aims to utilize this knowledge in conjunction with both multitask and self-supervised to create an attack-agnostic defense by making use of the effects of changes in shared hidden representations on unattacked self-supervised tasks.

III. Methodology

A. Tasks Description

- **Image Inpainting** [5] is a self-supervised task in which a model is tasked with filling in a removed region of an input image. This is done using two models: a generator, which generates a "mask" to in the blank, and a discriminator, which determines whether a given mask is real or generated. The generator G is trained using a combination of two losses:

$$\mathcal{L}_G = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec} \tag{1}$$

of which adversarial loss pertains to the output of the discriminator D on the generated mask and reconstruction loss is defined by the normalized L2 distance between the generated mask and the missing region. On the other hand, the discriminator D uses a loss defined by predictions on the ground truth of the missing region of image x, M(x), and generated mask, G(x):

$$\mathcal{L}_D = log(D(M(x))) + log(1 - D(G(x)) \tag{2}$$

Note that the discriminator's input is only the masked region–it receives no context information and thus only serves to determine whether its input looks real, regardless of whether the input matches its surroundings. Without the generator's L2 pixelwise loss term, the adversarial loss may push the generator to output the same mask in all scenarios, while without the adversarial loss term, the generator may be incentivized to output blurry masks with pixel values close to the overall mean of the distribution rather than ones with finer details that could potentially be harshly penalized by L2 error.

- **Jigsaw Puzzles** [5] is a self-supervised learning task inspired by a real world game. In the real game, a multitude of interlocking pieces each representing a portion of a picture are shuffled around and the (human) player must find the correct way to re-assemble them and reconstruct the image. Jigsaw puzzles were initially invented to help children learn geography and are generally associated with learning and improving visual-spatial processing. Intuitively, it thus makes sense that this game could be adapted to a deep learning context as a self-supervised task to help the model learn hidden structural representations of the data. Furthermore, the jigsaw puzzles task is in essence very different from image inpainting as it is a discriminative task so it could be interesting to contrast results obtained with those of image inpainting.

  In the computer version of jigsaw puzzles, we divide an image into a grid of tiles. We then permute the tiles before feeding them as input to the network. The model is then tasked with predicting the set of permutations undergone. Furthermore, in order to make the training transferable to other tasks such as classification, we want to encourage the model to learn global structures of the data as much as possible. This means preventing it from learning low level features (color/texture) common to all tiles or cues such as texture continuity at tile boundaries. Indeed, such features do not require understanding of the global object structure and could be used as unwanted shortcuts by the model. We achieve this delicate learning objective in part through a careful pre-processing of images: a random square crop of 225 × 225 pixel is first taken from the input image, then divided into a 3 × 3 grid. Then a random 64 × 64 pixel tile is picked from each of the nine 75 × 75 pixel cells obtained and jittering is randomly applied to the pixels. Note that square tiles prevent the model from relying on shape of tiles in the proximity of borders, while leaving a random gap between each tile prevents it from using feature continuity at the borders as a shortcut feature. See figure 2 for a visual depiction of a sample input.

  The model uses a siamese-ennead CNN architecture which the authors called the *context free network* (CFN). Its architecture is depicted in figure 3. It consists of multiple identical AlexNet CNN subnetworks with shared weights, each taking one of the tiles as input. The outputs of these individual CNNs are then concatenated and given as input to a common set of fully connected layers. The main motivation behind this architecture is to delay the computation of low-level statistics across different tiles and instead focus on features specific to each tile in the early stages of the network. The last fully connected layers then combines these insights to learn the overall spacial arrangement.

B. Problem Formulation and Design

General Approach: Because adversarial attacks change the hidden representations inside a network, an adversarial attack aimed at fooling one output task may unintentionally alter

Fig. 2. Jigsaw Image preprocessing

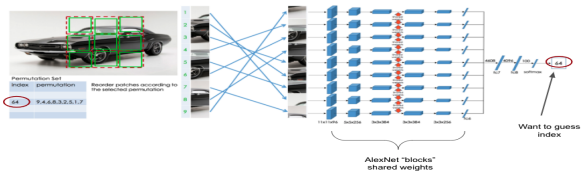

Fig. 3. Context Free Network used in Jigsaw Puzzles task

the outputs of auxiliary tasks that share the same backbone network. If these auxiliary tasks are self-supervised, their predefined outputs give us a unique opportunity to create an additional perturbation that may reduce the effects of the adversarial perturbation. As we know the correct outputs for self-supervised tasks, we utilize adversarial attack methods to construct our own "reverse" attack that seeks to find a perturbation that minimizes loss on the self-supervised tasks in the hopes that this perturbation acts to reverse the effects of any potential attack on the primary objective task.

It is uncertain whether these additional perturbations will have any positive effects. Auxiliary tasks must be negatively affected by adversarial perturbations for another task, but theoretical analysis on the difficulties of successfully simultaneously attacking multiple tasks [3] may imply that attacks that aren't even targeted toward these auxiliary tasks will have little impact on their outputs. Even if the self-supervised tasks' outputs are affected, there is no guarantee that the perturbation crafted to minimize their loss will have a positive impact on the primary objective task. In addition, we must also note that the additional perturbations we craft may negatively impact performance on natural images, although this may be mitigated by only applying the additional perturbation when we have a strong belief that the input has been attacked.

- **Image Inpainting** appears to have a straightforward application for multitask learning. The generator accepts an input of the same dimensions as any classifier, and the downsampling layers can be interpreted as feature extractors. Thus, the upsampling portion of the generator may be used as an generator output head that follows a backbone feature extractor in a multitask model.
  The reverse attack objective will be to minimize the generator's loss, which was previously defined as a weighted sum of the adversarial(from discriminator) and reconstruction(pixelwise L2) losses. The generator is trained on images with masks located in random positions, but for the purpose of constructing a reverse attack, there are any number of ways to choose the mask location. Because attempting to minimize generator loss across multiple different mask locations is akin to simultaneously attacking multiple outputs of a multitask network, perturbations crafted to minimize loss on only a single mask are much more effective.
- **Jigsaw Puzzles** can be naturally repurposed for a classification task since all the individual tile level CNNs share weights and use the well-known AlexNet [6] classifier architecture. However, implementing an underlying unified backbone for both classification and jigsaw does require a bit of Pytorch gymnastics since each unit of the CFN's receptive field consist of one tile whereas the first layer of the AlexNet network takes in an entire image. These discrepancies can in part be addressed by setting the stride of the first layer of the tile-specific CFN to 2 instead of 4.
  Furthermore, empirical testing needs to be conducted to determine which layers to include as the backbone common to both tasks. A natural choice would be to use the CNN sub-networks as common backbone, however, [7] surprisingly found that training the last convolution layer (*conv5*) separately for both task yielded better results on the classification task.

IV. Implementation

- In **Image Inpainting**, the generator acts as an obvious candidate for multitask learning. In our implementation, built upon PyTorch-GAN's implementation of inpainting, a fully connected output layer was inserted after the generator's final downsampling layer to act as the classifier head for our primary objective task. Perhaps due to the simple architecture of the downsampling layers and the presence of only a single fully connected layer, the resulting classifier is fairly weak. The inpainting task is trained with randomly chosen masked regions, while evaluation is conducted only with a centered mask. The masks are chosen to be squares with side lengths of half that of the input image. The reverse attack, implemented as a PGD-in-reverse built to minimize the generator's loss, makes use of a single centered mask for loss calculations. All experiments with the inpainting task are conducted on CIFAR-10 with image sizes of 32x32.
- **Jigsaw Puzzles** We used this repository as a reference to implement the jigsaw puzzles task. Our pre-processing functionality includes the data processing steps mentioned above as well as the generation of a file containing a reference set of possible tile permutations (note that the permutations are generated to maximize hamming distance with the original arrangement of tiles). We use these reference permutations in the evaluation step of the self-supervised task.
  Because the original paper [7] mentioned that training on ImageNet took a couple of days on a GPU, we decided to use CIFAR-10 to speed up training instead. Unfortunately, we only managed to get poor performance,

for reasons that we were not able to confirm.
Excluding the possibility of a bug in the code, one possible reason could have been the limited number of epochs we trained the model on (we ran into a lot of technical/quota approval difficulties trying to set up a GPU and so conducted the training on a CPU which was slow). Another reason could also have simply been the small size of CIFAR-10 images. Since each CNN in the input layer of the CFN looks at a subset of the input image (i.e: one tile) at a time, the low resolution of the CIFAR-10 images could have been a particular hindrance to learning in the case. See figures 4 and 5 for an example CIFAR-10 input to the network.

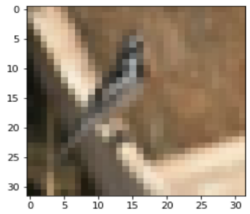

Fig. 4. Original CIFAR-10 image

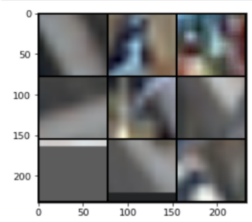

Fig. 5. After preprocessing (note the low resolution of each tile)

Future work would consist in trying out other higher resolution datasets with a good number of categories to choose from. Tiny-imagenet from Stanford was the dataset we had in mind for the future.

In the following section, we will thus present results for the inpainting task only.

V. Experiments

Image Inpainting: We conduct two experiments to measure the performance benefits of including inpainting in a multitask model. In multitask training, the primary task (classifier) and auxiliary task (inpainting generator) are trained jointly and thus both contribute gradients to the training of their shared feature extractor layers. We also conduct a second experiment in which we first train the classifier and freeze the backbone feature extractor before training the inpainting generator. In each experiment, we record the clean and adversarial accuracies with and without a reverse attack constructed from applying 10 iterations of PGD with the intent to minimize loss on the generator.

A. Results

The obtained accuracy results from experiments with image inpainting are shown in Table I. All experiments were run using the CIFAR-10 dataset.

TABLE I
Image Inpainting for Adversarial Robustness

Multitask	**Clean**	**Clean + Rev**	**Adv**	**Adv + Rev**
FGSM	61.25%	44.32%	01.26%	12.61%
PGD(2 iters)	61.25%	44.32%	00.02%	02.45%
Classifier First	**Clean**	**Clean + Rev**	**Adv**	**Adv + Rev**
FGSM	71.42%	55.13%	10.72%	17.35%
PGD(2 iters)	71.42%	55.13%	05.08%	06.53%

In both cases, we find that clean accuracy is negatively impacted by the reverse attack. Although the reverse attack appears to have slight benefits against weak attacks such as FGSM, even a slightly stronger 2-iteration PGD attack is able to drop performance to below that of random guessing.

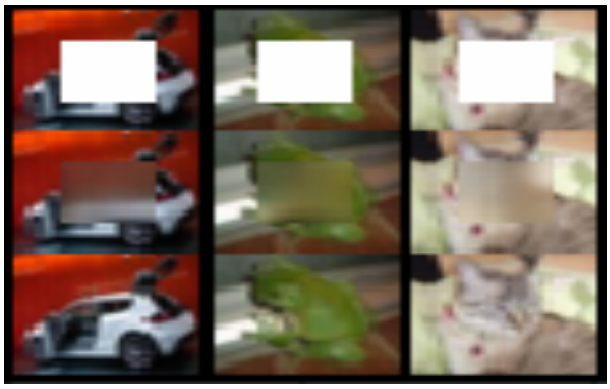

Fig. 6. Top to bottom: masked image, generated mask, unmasked image

Perhaps the way the inpainting task utilizes feature representations is too dissimilar from that of the classification task, such that a perturbation that lowers loss for the inpainter is little more than random noise to the classifier. As a quarter of the total image area is not shown the the inpainting task, the resulting representations as well as the constructed reverse attacks are unable to make use of all pixels in the image. It could also be that the inpainter could not learn sufficiently powerful representations; CIFAR-10's small image sizes may be less than ideal for this self-supervised task.

Figure 6 shows three examples of inpainting with centered masks. Due to CIFAR-10's small image sizes, there is not as much detail for the inpainter to work with relative to the amount of detail that can be expressed in higher-resolution images, which may lead to blurry masks.

VI. Conclusion

From our experiments with image inpainting, we find that multitask learning may severely reduce model performance. The low clean accuracy may be the result of poor multitask compatibility; the classifier utilizes full images while the

generator utilizes images in which a quarter of the pixels have been masked. Given our simplistic implementation's architecture and the small input image sizes, it may have been difficult for the shared backbone to learn a representation to satisfy the two tasks' different needs.

The performance of the inpainting task's reverse attack was less than stellar. While the perturbations which minimized loss for the inpainting task had some effect against weak adversarial examples, the benefits quickly disappeared as we increased the attack strength.

While the idea of using self-supervised tasks has potential benefits in being additional tasks for multitask learning and in crafting "cleaning" perturbations and may be a road to attack-agnostic robustness, our findings show that the path is not so clear-cut: neither multitask learning nor reverse attacks crafted with the aid of self-supervised tasks are guaranteed to improve performance or robustness.

Future works may seek to experiment with other self-supervised tasks, or explore the conditions under which a task is suitable for multitask learning or use in crafting reverse attacks. Another direction may be to discover when to apply such reverse attacks, which could alleviate the problem of the reduction of clean accuracy. Although our findings are fairly negative, they show there may still be some promise in the use of self-supervised tasks for adversarial robustness.

VII. Acknowledgement

We want to thank Chengzhi Mao for letting us join his project and for sharing his ideas with us, as well as professor Yang for his advice throughout the entire semester.

References

[1] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training, 2016.

[2] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing, 2018.

[3] Chengzhi Mao, Amogh Gupta, Vikram Nitin, Baishakhi Ray, Shuran Song, Junfeng Yang, and Carl Vondrick. Multitask learning strengthens adversarial robustness, 2020.

[4] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness, 2019.

[5] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting, 2016.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[7] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.