Can You Take A Punch?: A Resiliency Survey Against Adversarial Attacks of Deep Models

GitHub Repo

Aaron Jimenez

University of California, Santa Barbara

GitHub Repo

Introduction

With the widescale adoption of new deep machine learning techniques –such as convolutional neural networks and transformers, the versatility and usefulness of these models has only increased. However, with advent of these new systems comes ways for bad actors to abuse and attack them through any number of vectors. One of the more dangerous methods of abusing this system is through adversarial attacks.

Adversarial attacks are attacks on systems in which the input to the system is subtly perturbed in such a way that it would be difficult for eyes to notice the change; however, to a the system being used it end up making a huge difference in its output.

In image classification tasks, this can take the form of making subtle changes to an image's pixel values can cause massive changes to way in which a system might classify an image, even if a human operator may not notice the changes – as seen in Figure 1.

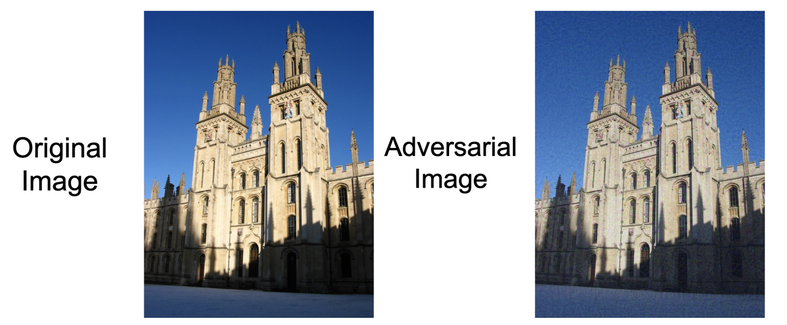

Figure 1: Example of adversarial attack in image classification generated using PIRE

In the realm of NLP, where input is many times text, these types of attacks can manifest as subtle changes in spelling of word or word replacement, as seen in Figure 2. Much like adversarial attacks on image systems, these sorts of attacks can cause massive disruptions to NLP systems. This is especially important in today's world, where most major services such as the Google search engine, Apple Siri, and so many others where a disruption to service can have major real-world consequences.

As such, I propose a survey of various major adversarial attack algorithms that are designed to attack both NLP and image classification models. In addition to researching the models, I wish to test these algorithms against current models out in the wild and record their resiliency against these forms of attack. In particular, I wish to use the BERT-Attack [1] and PIRE [3] algorithm to test the effectiveness image classification attacks.

Original: **"when i voted my \" 1 \" for this film i noticed that 75 people voted the same out of 146 total votes . that means that half the people that voted for this film feel it 's truly terrible . i saw this not long ago at a film festival and i was really unimpressed by it 's poor execution . the cinematography is unwatchable , the sound is bad , the story is cut and pasted from many other movies , and the acting is dreadful . this movie is basically a poor rip - off of three other films . no wonder this was never released in the usa ."**

Adversarial: **"when i voted my \" 01 ? for this drama i saw that 78 citizens passed the same out of 1500 , votes ... that . that half the persons that voted for this film view it is was truly terrible . i saw this not long ago at a film . and i was really unimpressed by it ` a wrong executed (the film is un watchable / the sound is bad like the tale is cuts and pasted from many other musicals , and the acts is dreadful . this movie is basically a poor , \" off of three other documentaries . no ! this was never issued in the usa ."**

Figure 2: Example of adversarial attack in text classification generated using BERT-Attack

BERT-Attack: Fighting BERT with BERT

With the recent rise of newer and newer pretrained language model, such as what was seen with BERT in 2018 [2], better and more robust NLP systems have been developed are more resistant to tradition forms of language adversarial attacks. However, BERT is a relatively robust pretrained masked language model with a large amount of general-purpose language knowledge; the authors propose that it could have the potential to generate fluent and semantically-consistent adversarial substitutions for differing inputs. As such they propose a new BERT-based adversarial sample generator, BERT-Attack [1]. The algorithm for their proposed system involves two major parts: finding the vulnerable words in an input and applying BERT in a semantic preserving manner to generate potential substitutes for the input.

Finding the Weak Links in an Input

In order to find the most vulnerable words in a input, BERT-Attack uses the logit output of the target model (be it a fine-tuned BERT model or some other model) in order to generate an importance score, I_{w_i}, for each word in the input. It is able to calculate these importance scores by first calculating the the logit output of the input, S, as $\mathrm{o}_y(S)$ for the correct label y. From there it masks the specific word that it is trying to find its importance value for, w_i, and gets its logit output $\mathrm{o}_y(S_{w_i})$. Finally, it finds the difference between these two to find the word's importance value, $\mathrm{I}_{w_i} = \mathrm{o}_y(S) - \mathrm{o}_y(S_{w_i})$ and the words are sorted in descending order. After this only the ϵ most important words are kept to keep perturbations at a minimum.

Finding the Right Words to Replace With

Once the list of vulnerable words, L, have been found, BERT-Attack iteratively replaces every word that appears in L to find the one that can most mislead the target model. It does this by using BERT to generate the replacement word by having it predict a replacement that is semantically coherent and fluent to the input using a single forward pass. Without having to use an additional neural model – as other methods do – to perform scoring on the input, the only time-consuming action is accessing the target model. For when searching for replacement words, the algorithm does not mask out the replacement word, w, within the original input when feeding it into the BERT model. This is because it could replace w with a word that is may be just as fluent, but may change the meaning of the entire input. In addition, if w were masked, then the model would have to be run for each iteration in L, which would be costly. Thus since BERT uses Byte-Pair-Encoding (BPE), we take the input sequence, S, and its sub-word token sequence, H, and align w with its sub-word in BERT. Then we run the BERT model on H to get the output prediction. However, instead of using argmax, we take the top K predictions.

From here the perturbed sequences are generated using the K predictions to obtain the best adversarial example $\mathrm{H}^{\mathrm{adv}}$. This is then done for the all words in L.

PIRE

Perturbation for image retrieval error (PIRE) is an unsupervised system for generating image perturbations for neural feature-based CBIR systems. The algorithm (Figure 3) begins by taking as its input an image query x, a neural feature function *f*, an iteration limit *T*, and a perturbation vector ϵ. From there, a random matrix of the same dimensions as the image query. This is the initial value of the perturbation vector v_i. From here the algorithm will iterate for *T* iterations, updating the perturbation vector, *vi*,with each iteration. In order to update v_i , the value v that maximizes the distance between $f(x)$ and $f(x + v_{i-1})$ is found. This is then followed by bounding v_i between -ϵ and ϵ. Finally, after iterating *T* times an amplified perturbation $10 * v_i$ is applied to x. This is done in order to prevent perturbations from being rounded away when an image is saved in an 8-bit format.

Refinement of PIRE

Instead of blindly making the perturbation large, a function is used to find the perturbation vector v which will magnify the v just as large enough to avoid the rounding vanishing effect. So, after refinement the final modified image becomes $p(v_i) + x$.

Algorithm 1: Perturbations for Image Retrieval Error (PIRE)

Input : Image query $\boldsymbol{x}$; Neural feature function f; Iteration limit T; Perturbation vector range ϵ;
Output : Adversarial image query $\boldsymbol{x} + 10 * \boldsymbol{v_i}$;

$w, h = \mathrm{size}(\boldsymbol{x}_1)$;
$i = 1$;
Generate a random matrix $\boldsymbol{v}_{0_{(w \times h)}}$;
while $i < T$ **do**
  Calculate the distance between original image $\boldsymbol{x}$ and perturbed image $\boldsymbol{x} + \boldsymbol{v}_{i-1}$;
  $\boldsymbol{v_i} = \underset{\boldsymbol{v}}{\mathrm{argmax}} \|(f(\boldsymbol{x}) - f(\boldsymbol{x} + \boldsymbol{v}_{i-1}))\|_2^2$;
  Project $\boldsymbol{v_i}$ into a L_∞ norm sphere;
  $\boldsymbol{v_i} = \mathrm{clip}(\boldsymbol{v}_i, -\epsilon, \epsilon)$;
  $i = i + 1$;
end
Return perturbed image query;
return $\boldsymbol{x} + 10 * \boldsymbol{v_i}$;

Figure 3: PIRE Algorithm

Experiments and Results

BERT-Attack Experiments

For my experiments with BERT-Attack, I decided to test two different aspects of the algorithm. I first wanted to test the robustness of the adversarial inputs that is generated from the standard BERT-Attack algorithm against other sentiment analysis transformer-based models. Secondly, I wanted to test to see if other masked language models could be used instead of BERT to generate the successful adversarial inputs using the same algorithm.

The purposes of these two experiments are two fold. The first is to test the real world robustness of the BERT-Attack algorithm in a grey box setting where the attacker may not have access to the same models that they are attacking, and thus must use a another model to generate queries. And the second is due to see if BERT is truly the best option or if there are any other MLMs that can generate better adversarial queries using the same original queries.

In order to conduct these two experiments, I decided to run the BERT-Attack algorithm on 250 queries from the IMDB Movie Review Dataset to generate their corresponding adversarial queries. From here, I ran BERT-Attack using BERT and also four other MLM models from the HuggingFace transformers library to generate 5 different sets of these 250 adversarial queries. I then ran these adversarial queries and their original counterparts through versions of models previously used that were trained for sentiment analysis. From here I would compare the model outputs of the adversarial queries with those of the original queries to calculate the successful disruption rate to the models per set of generated adversarial queries.

BERT-Attack Results

generator\model	bert	distilbert	electra	mobilebert	roberta
bert	0.12	0.028	0.16	0.276	0.164
distilbert	0.112	0.124	0.148	0.248	0.172
electra	0.132	0.16	0.164	0.236	0.148
mobilebert	0.144	0.008	0.16	0.204	0.14
roberta	0.188	0.264	0.24	0.36	0.212

Table 1: Measure of success rate of adversarial inputs. Higher is better.

After running the experiment, one can find some interesting results. The most striking of these is how susceptible to adversarial queries MobileBERT [7] seems to be relative to the models. Generally, MobileBERT tended to successfully disrupted about 25% of the time. This is significant as this model was designed to run primarily on edge devices, such mobile phones, which make up a large amount of the world's computers.

In contrast, DistilBERT [5] which was also designed as way of reducing the BERT model down to smaller size while maintaining performance managed to generally be more robust at defending against attacks than even the original BERT-base model itself. This seems to suggest that something during the process of knowledge distillation that was used to the authors to distil BERT down to their DistilBERT model amplified its defenses against attacks while simultaneously close to BERT-like performance.

Aside from MobileBERT, it appears that for the most part the sentiment analysis models tend to have decent, but not great resistance to the the adversarial inputs overall. Some models struggles a bit more when taking inputs generated from certain models; however, overall performance was hits were around the 15% mark.

PIRE Experiment

For my experiment with PIRE, I decided to go a similar route to my experiments with BERT-Attack. I decided to use PIRE to generate a set of adversarial image queries based a subset of images from both the Oxford5k and Paris6K datasets and test their effectiveness is disrupting image classification models. While PIRE may be a system originally designed for CBIR systems in trying to find the top K closely related images from a corpus, I believe that the methods it uses to disrupt those systems – given that it uses many of the same deep models as what are used for image classification for its image embeddings – can still be effective in disrupting an image classification model.

Thus for this experiment, much like with BERT-Attack, I plan to pass the generated adversarial image queries and the original image queries through multiple image classification models. From here a disruption success rate will be calculated for each model based on the number of successful disruptions to the model over all of the image queries. In order to determine a successful disruption, the output of passing the original query will be used as the ground truth, which a successful disruption being when the model output of an adversarial query does not match the ground truth.

PIRE Results

ResNet101	VGG19	GoogLeNet	ViT	InceptionV3	EfficientNetv2-L
0.3	0.2818	0.1545	0.2091	0.2545	0.2364

Table 2: Measure of success rate of adversarial image inputs. Higher is better.

As can be seen in from the results in Table 2, PIRE has on average a success rate of about 25% based on the 6 models used. Of the models that used in the experiments ViT [9] and GoogLeNet [8] had the two lowest success rates, implying that of the of the 6 models used they were the two most robust models based on the generated inputs.

This result is interesting as while, ViT makes sense as to why it would perform better as it is a transformer-based image classification model and it attention mechanisms should be able to better detect perturbations generated by CNN systems, GoogLeNet is a bit more surprising. Given that GoogLeNet is an older model from almost 8 years ago (as of conducting this experiment) and has a simpler network architecture, it would be assumed that it would perform worse. However, it is perhaps this simpler architecture with less convolutional layers that allows it to not lose some of the finer details of the input image that have been changed based PIRE, and thus it is able to pick them up when doing the final classification based on its final image embedding. This would be an interesting topic for further investigation down the line.

Conclusion

With the use of more ML models in systems used in everyday lives, the threat of disruptions to those systems through disruptive inputs are very real. As can be seen the experiments presented in this study, algorithms such as BERT-Attack and PIRE, which use a grey box approach to adversarial input generation, can generate inputs that successful in making a relatively noticeable effect on normal system performance. Even more surprising is that one systems considered more robust, adversarial inputs still have the potential to cause relatively noticeable disruptions. Taking these results into consideration, perhaps more thought should be given when designing architectures and during training to take into account adversarial images and find ways to filter out the perturbations generated.

References

[1] Li, L., Ma, R., Guo, Q., Xue, X. & Qiu, X. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. (arXiv,2020), https://arxiv.org/abs/2004.09984

[2] Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (arXiv,2018), https://arxiv.org/abs/1810.04805

[3] Liu, Z., Zhao, Z. & Larson, M. Who's Afraid of Adversarial Queries?: The Impact of Image Modifications on Content-based Image Retrieval. (2019,6)

[4] Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv 1409.1556*. (2014,9)

[5] Sanh, V., Debut, L., Chaumond, J. & Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*. **abs/1910.01108** (2019), http://arxiv.org/abs/1910.01108

[6] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. (arXiv,2015), https://arxiv.org/abs/1512.03385

[7] Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y. & Zhou, D. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. (arXiv,2020), https://arxiv.org/abs/2004.02984

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Going Deeper with Convolutions. (arXiv,2014), https://arxiv.org/abs/1409.4842

[9] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. & Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. (arXiv,2020), https://arxiv.org/abs/2010.11929