RAD-ATTACK: REGULARIZED ADVERSARIAL WHITE-BOX ATTACKS ON VLMS

Ganesh Nanduru, Nate Kimball, Alex Fetea
University of Virginia
Charlottesville, VA 20171, USA
{bae9wk, tma5gv, pvn5nv}@virginia.edu

ABSTRACT

Large language models (LLMs) are among the most popular and widely used machine learning models used today. With their enhanced capabilities in downstream tasks, particularly question answering and visual processing, LLMs are gaining traction in impactful applications such as healthcare, commerce, and education. Therefore, it is especially important for LLMs to be robust to adversarial prompting in generating safe and factually accurate responses. To progress research done into fortifying LLMs against misclassification, we present **RAD-Attack** (Regularized ADversarial Attack), an open-source adversarial white-box attack involving a gradient traversal algorithm that minimally perturbs images for model misclassification. We target LLaVA-1.5, a state-of-the-art multimodal transformer that combines the CLIP vision encoder and the Vicuna language model. After proving our concept by running our algorithm on images from the MNIST dataset, we extend our work to more complex multi-token classification tasks such as CIFAR classification and towards fine-tuning VLMs to resist adversarial image attacks.

1 INTRODUCTION

1.1 IMPORTANCE

LLMs are becoming increasingly accessible to the public. Companies like Google and OpenAI offer free interfaces with chatbots such as Bard and ChatGPT to the public. ChatGPT is estimated to have around 180 million users, so it is vital that the LLM is trained for safety to ensure its many users are protected from harmful inaccurate model output.

Multimodal language models that combine visual and textual processing also fall into this category, and are even more sensitive to attacks due to their complexity. By understanding and mitigating the effects of adversarial attacks, we can enhance model robustness and reliability. Our approach aims to extend the methods of current attacks by developing new algorithms and exploring Vision Language Model (VLM) vulnerabilities and mitigation strategies.

1.2 PURPOSE

Adversarial attacks on LLMs have been studied in examples such as LLM-Attacks (Zou et al., 2023), where text prompts are altered to manipulate a language model into outputting harmful or offensive responses. Datasets of adversarial samples, most prominently AdvGLUE (Wang et al., 2022), have popularized using known adversarial attacks as a method of improving model robustness by providing a benchmark for attack success rate. We notice a trend that most adversarial attacks on LLMs seek to trick the model into generating undesirable output through some form of prompt engineering. However, prompt engineering against strictly-text based LLMs is less applicable to multimodal language models with vision capabilities.

In the visual model space, adversarial attacks such as the Fast Gradient Sign Method (FGSM) have been established as effective ways to trick convolutional neural networks (CNNs). Attacks perturbing images are typically designed for CNNs, and they have not been thoroughly assessed for language models. To bridge this gap, we present RAD-Attack, a regularized adversarial attack de-

veloped to trick a visual language model into misclassifying an input image by minimally altering its pixel values.

2 Related Work

2.1 Towards Adversarial Attack on Vision-Language Pre-training Models

The researchers in this paper explore the unique vulnerabilities present in vision-language models. This study uncovers how the models combine visual and textual information, and how adversarial manipulated prompts in either modality can deceive the model. The findings show that multi modal models are not only susceptible to traditional textual adversarial attacks, but also to attacks that exploit the visual modality (Zhang, 2022).

2.2 Explaining and Harnessing Adversarial Examples

This paper challenges earlier theories that the susceptibility of neural networks to adversarial examples stems from issues of non linearity and over fitting. The authors propose that the linear nature of the models is the primary cause of their vulnerability. This discovery leads to the Fast Gradient Sign Method, a straightforward way to generate adversarial examples that can be used for adversarial training. This approach displays the potential of using gradient ascent on MNIST samples to improve the resilience of models against adversarial attacks, and is directly relevant to our efforts to analyze the impact of adversarial attacks on LLaVA (Goodfellow et al., 2014).

2.3 Maximal Jacobian-based Saliency Map Attack

The Jacobian-based Saliency Map Attack is a unique way of targeting pixels that are identified as salient by the attack algorithm in order to minimally perturb an image to generate an adversarial sample. Researchers develop the attack to target CNNs, which can be prone to overrepresenting certain image features from the input space. By calculating the Jacobian of the target model's loss, researchers are able to create a saliency map to provide to their algorithm performing the pixel-level perturbations that cause misclassification (Wiyatno & Xu, 2018).

2.4 Towards Deep Learning Models Resistant to Adversarial Attacks

This paper furthers discussion on resistance to adversarial attacks by proposing a framework for training models to be inherently resistant to these attacks. This framework can be summarized by training models under the worst case scenarios to counteract adversarial attacks. This approach attempts to expose models to a much wider variety of input, strengthening the model in atypical inputs (Carlini & Wagner, 2017).

3 Problem Setup

3.1 Model Selection

We target LLaVA-1.5, a large multimodal model with state-of-the-art performance on Science QA (Liu et al., 2023). We choose LLaVA due to its ease of access through open-source code and public model weights, as well as its popularity as one of the most prolific vision-language models (VLMs) in recent history.

3.2 Experimental Context

To evaluate our approach we used two well known image datasets: CIFAR-10 and MNIST. The MNIST dataset consists of 70,000 images of handwritten digits, 28x28 pixels, widely used for training machine learning algorithms. The CIFAR-10 dataset comprises 60,000 32x32 color images in 10 different classes, ranging from animals to vehicles, a more diverse challenge.

We tasked LLaVA with classifying images from these datasets, as depicted in Figure 1. We created a classification prompt for the image datasets, examples of which are shown in Figures 2 and 3. The

process involves generating the output logits for each image/prompt pairing. Based on these output logits, we determine which of the tokens is most probable out of the tokens corresponding to the image labels. We choose this post probable label as the prediction. This approach not only allows us to determine the classification accuracy of LLaVA, but also serves as the baseline for how the adversarial inputs will affect the model.

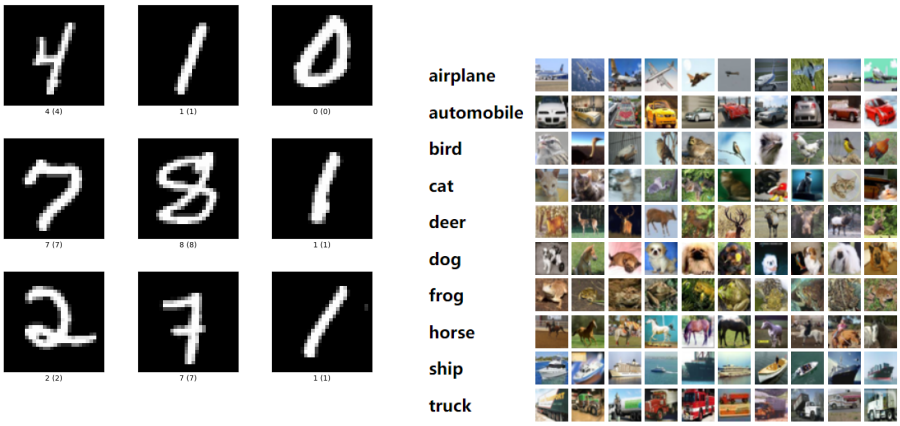

Figure 1: Example images from the MNIST and CIFAR-10 datasets processed by LLaVA, labeled respectively.

```
"USER: <image>\nFrom the following list of options:
airplane, automobile, bird, cat, deer, dog,
frog, horse, ship, truck, what is this image
showing?\nASSISTANT: This is a "
```

Figure 2: Example of a prompt used to classify CIFAR data

```
"USER: <image>\nWhat digit [0-9] is
this?\nASSISTANT: This is the digit "
```

Figure 3: Example of a prompt used to classify MNIST data

4 METHOD

4.1 OVERVIEW

We develop 4 different attack algorithms capable of causing language models to misclassify images. We develop two primary loss functions that we optimize with gradient descent until the model misclassifies the image. The first loss function takes a target label that is the maximum likelihood class that is not the true label, $y_{target} = \underset{y \in Y \setminus y*}{\text{argmax}} M(x)$, and computes the cross-entropy loss between the model's output logits and the target. The goal of loss 1 is to maximize the probability of the model classifying the image as the desired target. The second loss function takes the true label as the target and computes the negative cross-entropy loss between the model's output logits and the target. The goal of loss 2 is to minimize the probability of the model classifying the image correctly. Additionally, we develop regularized versions of these loss functions that add a weighted term for the deviation of the perturbed image. The goal of the regularizer is to optimize misclassification loss while minimizing the perturbation so that the adversarial image is as close to the original image as possible. We define the magnitude of the perturbation as the L2 distance between the perturbed and original images.

```
def loss_fn_1(logits, target):
    return torch.nn.CrossEntropyLoss()(logits, target)

def loss_fn_2(logits, label):
    return -torch.nn.CrossEntropyLoss()(logits, label)
```

Figure 4: Loss functions

Algorithm 1 Plain Attack

Require: Original image x, Target label y_{target}, Model M, Loss function L, Step size ϵ
Ensure: Adversarial image x_{adv}
1: Initialize $x_{adv} \leftarrow x$
2: **while** not misclassified **do**
3: Compute misclassification loss: $L_{mis} \leftarrow L(M(x_{adv}), y_{target})$
4: Compute gradient: $g \leftarrow \nabla_{x_{adv}} L_{mis}$
5: Update adversarial image: $x_{adv} \leftarrow x_{adv} - \epsilon \cdot g$
6: **end while**
7: **return** x_{adv}

Algorithm 2 Regularized Attack

Require: Original image x, Target label y_{target}, Model M, Loss function L, Regularization parameter λ, Step size ϵ
Ensure: Adversarial image x_{adv}
1: Initialize $x_{adv} \leftarrow x$
2: **while** not misclassified **do**
3: Compute misclassification loss: $L_{mis} \leftarrow L(M(x_{adv}), y_{target})$
4: Compute deviation loss: $L_{dev} \leftarrow \|x_{adv} - x\|_2^2$
5: Compute total loss: $L_{total} \leftarrow L_{mis} + \lambda \cdot L_{dev}$
6: Compute gradient: $g \leftarrow \nabla_{x_{adv}} L_{total}$
7: Update adversarial image: $x_{adv} \leftarrow x_{adv} - \epsilon \cdot g$
8: **end while**
9: **return** x_{adv}

4.2 Methodology for the CIFAR-10 Attack

In addition to the MNIST dataset, an adversarial attack was designed for the CIFAR-10 dataset to evaluate the attack's effectiveness on a different class of data. The complexity of the CIFAR-10 dataset was heightened by the amount of tokens in the class labels. Unlike the MNIST data classes, which are all 1 token in length and thereby trivial to handle, the CIFAR dataset features multi-token labels of varying lengths. This complexity made it challenging to directly compare model outputs with the tokenized labels.

After evaluating several strategies, the selected approach for CIFAR-10 involved calculating a moving average of the output logits to align with the multi-token labels. For example, consider a scenario where 5 output logits must be compared to a 3-token label. The first step involves averaging logits 0-2 and comparing this average to the first token of the label. Subsequently, logits 1-3 are averaged and compared to the second token, and finally, logits 2-4 are averaged and aligned with the third token. This sliding window technique was developed to ensure precise matching between the model's outputs and the complex label structures of the CIFAR-10 dataset.

4.3 Adversarial Attack Procedure on CIFAR-10

We conduct he adversarial attack on the CIFAR dataset over 100 iterations with a hard stop condition. Each iteration involves a slight perturbation of the image, and the process stops by either

producing a misclassification or reaching 100 iterations. We determine the effectiveness of each perturbation using the loss between the output logits and the predetermined target label logits.

To select a target label, we obtained the output logits for the original image, and identified the class label the image least likely corresponded with. This label served as the target for the attack. At each iteration, we compared the logits of this least likely label against the predicted label's logits. The loss was calculated based on the difference between these logits, incrementally shifting the model's prediction toward the target label.

5 Experiment Setup

5.1 Dataset Description and Preparation

The LLaVA model handles our image preprocessing: it scales images to 336x336 pixels, normalizes them, then converts them into tensors. Another important aspect is that the class labels are tokenized, so they can be directly compared with the output tokens generated by the model. This step is crucial for aligning the output of the model with the class labels.

The classification prompts input to the model along with each image are carefully constructed for each dataset. The class options are embedded into each prompt to clearly provide the valid answers. We noticed a much higher classification accuracy with these options embedded because it alleviates surface form competition by conditioning the model on possible answers.

In our experiments, we study the effect of our attack's loss function on 3 metrics: the average distance of the successfully perturbed image from the original image, the average number of iterations of gradient descent required to fool the model, and success rate, the percent of attacks that fool the model within 100 iterations.

We start by fine-tuning the hyperparameter, λ, in our regularized loss functions. We run a hyperparameter sweep with λ on a logarithmic range between 1e1 and 1e5 with 12 intervals, and measure the attack's performance on a dataset of 800 MNIST digits.

Next, we compare all 4 attacks on a dataset of 3000 MNIST digits. We run each attack on each image and measure the L2-distance of the perturbed image from the original image, the number of iterations until misclassification, and the success rate of each attack. An attack is considered a failure if the gradient explodes, $x_{adv} \to \infty$, or the attack takes longer than 100 iterations.

6 Results

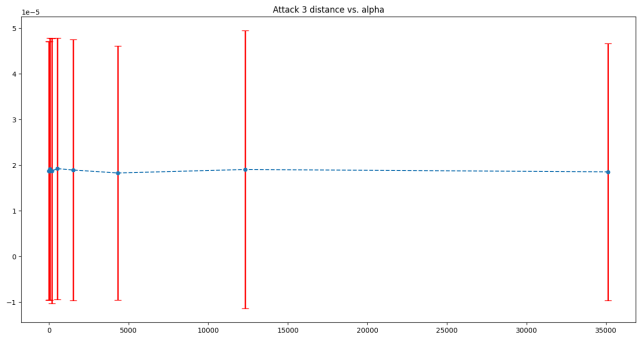

Figure 5: Attack 3: Distance vs. Lambda

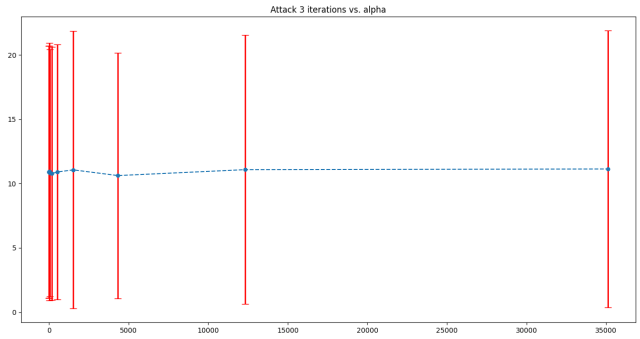

Figure 6: Attack 3: Iterations vs. Lambda

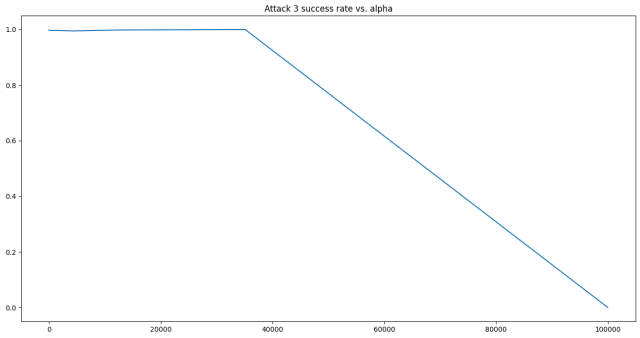

Figure 7: Attack 3: Success Rate vs. Lambda

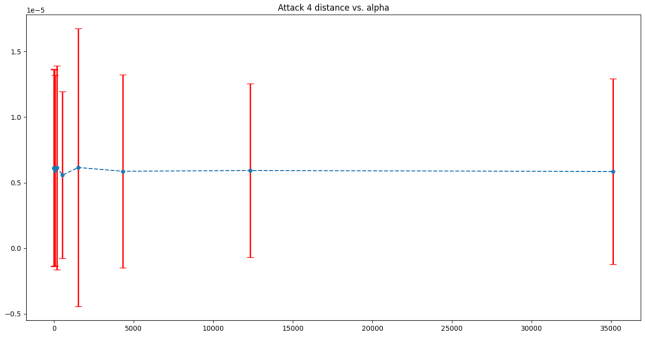

Figure 8: Attack 4: Distance vs. Lambda

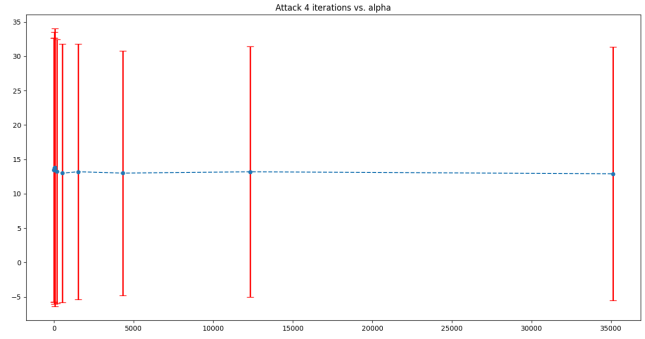

Figure 9: Attack 4: Iterations vs. Lambda

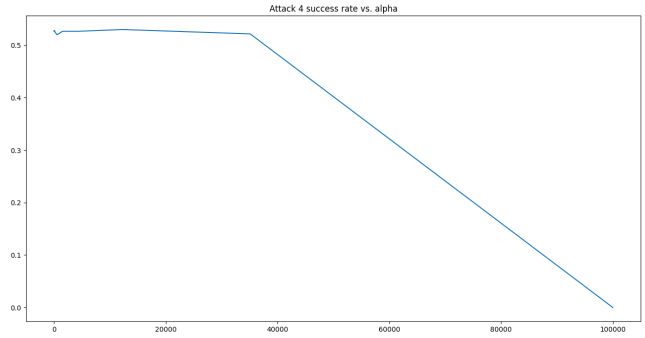

Figure 10: Attack 4: Success Rate vs. Lambda

6.1 Results of Hyperparameter Tuning

For attack 3, we find that the value of the hyperparameter has a negligible impact on distance and iterations (fig 5, 6). Lambda has little effect on success for low lambda, but success rate sharply decreases as lambda gets large (fig 7).

For attack 4, we find that there is a slight convexity in the relationships between lambda and distance and lambda and iterations. Both graphs, seem to have a minimum between 100 and 1000, although it is hard to be confident about that, as evidenced by the large variance in estimates (fig 8,9). Based on these observations, we use a lambda of 100 for further experiments. We see a similar inverse

Table 1: Attack Comparison ($\lambda = 100, \epsilon = 0.1$)

Attack #	Average Distance	Average Iterations	Success Rate	Loss #	Regularized
1	1.76e-05	10.6	0.998	1	No
2	6.13e-06	13.2	0.530	2	No
3	1.77e-05	10.6	0.996	1	Yes
4	5.86e-06	13.4	0.531	2	Yes

relationship between lambda and success rate as before, where the magnitude of the slope increases with lambda (fig 10).

6.2 Results of Attack Comparison

In table 1, we compare the 4 attacks. It is desirable to have an attack with minimal perturbation = smaller distance, quick convergence = lower iterations, and high success rate. The third and fourth attacks are regularized versions of the first and second attacks, respectively. We find that attack 2 achieves nearly a 3x decrease in distance compared to attack 1. However, attack 2 performs about 25% worse in iterations, and has nearly 1/2 the success rate of attack 1. Counterintuitively, we found adding a regularizer to the first loss function increases distance by a small amount. It also slightly increases iterations and decreases success rate. When we added the regularizer to the second loss function, we saw about a 5% decrease in distance and a slight increase in success rate and iterations. On the whole, it seems loss 2 is preferable to loss 1 if the goal is minimum perturbation adversarial attacks, however, it is less efficient than loss 2 in terms of iterations to converge and success rate. Loss 2 is better if success rate or speed is more important. On the whole, We find that adding the regularizer to loss 1 had a negligible to slightly negative effect on performance. Adding the regularizer to loss 2 had a positive effect on performance while only slightly increasing iterations.

6.3 Results of the CIFAR-10 Attack

The CIFAR-10 attack methodology outlined before was put to test to determine its effectiveness in aligning output logits with multi-token labels. This approach proved to be the most effective among the strategies tested, achieving a classification accuracy of 61% in the CIFAR image classification tasks.

Further, our analysis revealed an 38% accuracy in misleading the model within 100 iterations. This success rate suggests that this accuracy can be further improved with more iterations, allowing more time for the slight perturbations to impact the image. One example of an original image vs perturbed image is shown in 11.

7 Challenges and Solutions

7.1 Reproducing Benchmarks

We encountered problems with obtaining baseline results. We find that code from related papers is often not thoroughly proofed for errors and is not written to run in diverse environments (different OS, Python version, library versions, system packages, etc.). Having to adjust our environment to work with highly specific configurations has been a major challenge to finding a benchmark, especially for examples like LLM Attacks (Zou et al., 2023) and COLD-Attack (Guo et al., 2024).

We were able to overcome complex package requirements with the help of docker containers. Making use of Apptainer on Rivanna has greatly helped in recreating the environments described in the research paper. While still time consuming to identify all the need system packages, the process has sped up our team collaboration on the project so far.

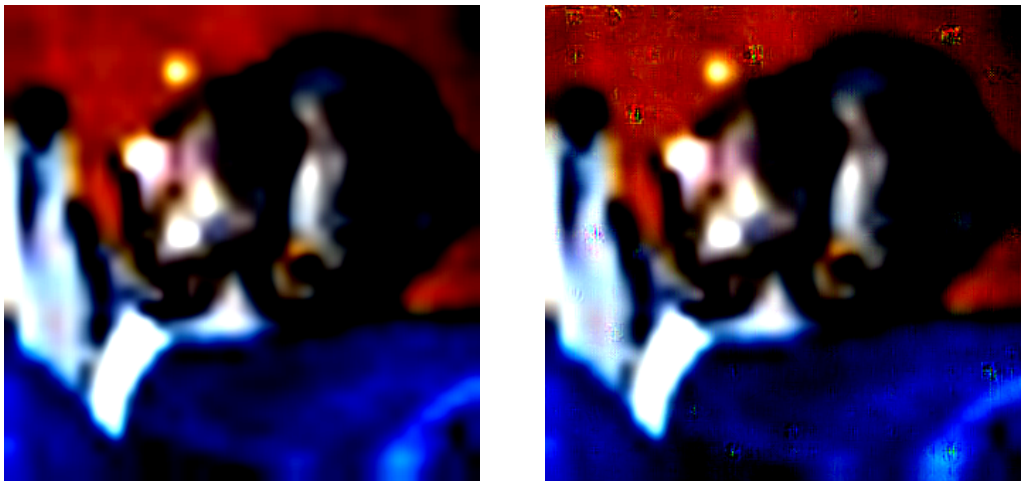

Figure 11: Comparison between an original CIFAR-10 image (left), originally classified a dog, and its attacked counterpart (right), which was misclassified as an automobile.

7.2 Training Language Models on Rivanna

We made significant progress toward training the language model on adversarially perturbed images. However, we ran into problems during runtime with out-of-memory errors.

One potential solution is data parallelizing the training across multiple GPUs. We could also investigate quantization or offloading to CPU memory to resolve out-of-memory errors.

7.3 Enabling Gradient Tracking for Images

In our adversarial attack, we depend on PyTorch's computational graph to traverse the gradient back to the input layer, yielding the input-level perturbations needed to reach the target false class from the correct class. However, PyTorch can only track computations to tensors. Because the HuggingFace pipeline we used to load and access LLaVA requires us to pass in visual input as PIL Images, we could not enable gradient tracking for the original input.

To fix this, we had to change our algorithm to perturb the images after they were processed by CLIP - this upscaled the image dimensions and changed the image from grayscale to RGB. In between the processing step and the forward step, we enabled gradient tracking for the image tensor.

8 Future Work

We want to investigate how training VLMs to recognize adversarially perturbed images might improve generalization as well as robustness to attacks. While these adversarially generated images look like they come from the same distribution to humans, they can quite easily fool otherwise smart models. There is reason to believe that training the model with perturbed images outside of the training distribution will lead to more robust and general representations.

A possibility for future work, given more time and computing resources, would be to develop scripts running this attack against several different models. Then, using the samples generated by this attacks, we would create an assorted database of images perturbed to fool different types of VLMs. We could train a new neural network to transform the original images to the perturbed images. If the neural network were to generalize, we would now have a black-box adversarial attack ready to create samples without the need for gradient information.

We also want to compare our methods against state of the art methods, including the Fast Gradient Sign Method (Goodfellow et al., 2014) and Projected Gradient Descent (Carlini & Wagner, 2017), as these apply similar methods to CNNs, and have yet to be tested against VLMs.

9 CONCLUSION

In this work, we develop 4 adversarial attacks for minimally perturbing visual input to cause a VLM to misclassify images. After we use the model's forward pass to calculate the loss between the model output and a target class, we traverse the gradient of this loss with respect to the input to alter the image into an adversarial sample. We iteratively perturb the input in this manner until the model outputs the target false class. We also develop a regularizer that in some cases significantly decreases the magnitude of the pertubation necessary for misclassification. With some of our attacks, we find a near 100% success rate given enough iterations for any particular sample. We demonstrate the effectiveness of our algorithm on the MNIST and CIFAR-10 datasets to attack the LLaVA-1.5 VLM.

Our successful implementation of a white-box adversarial attack on a VLM expands the collection of known attacks on VLMs, allowing developers to better understand the vulnerabilities in their models and protect against adversarial attacks. One potential usage of our algorithm to boost model robustness is for a developer to generate an adversarial dataset using our attack. After perturbed images are generated en masse, the developer could fine-tune their model, teaching it to protect itself against pertubation-based attacks.

REFERENCES

Nicholas Carlini and David Wagner. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/pdf/1706.06083.pdf.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. URL https://arxiv.org/abs/1412.6572.

Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. Cold-attack: Jailbreaking llms with stealthiness and controllability, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial glue: A multi-task benchmark for robustness evaluation of language models, 2022.

Rey Reza Wiyatno and Anqi Xu. Maximal jacobian-based saliency map attack. *ArXiv*, abs/1808.07945, 2018. URL https://api.semanticscholar.org/CorpusID:52091347.

Jiaming Zhang. Towards adversarial attack on vision-language pre-training models. *arXiv preprint arXiv:2206.09391*, 2022. URL https://arxiv.org/abs/2206.09391.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.