Adversarial Attacks on Cyberbullying Image Detection Models

<Instructor>

Outline

- What is cyberbullying
- Problem of cyberbullying on images
- Detecting cyberbullying on images
- Adversarial examples
- Attacking cyberbullying detection models
- Lab demonstration

What is Cyberbullying?

- Bullying that takes place electronically – cell phones, computers, and tablets, etc.
- Can occur through text, image, apps, forums, gaming, and social media.
  - Wherever people can share, view, and participate in a discussion.
- Content is usually hateful, offensive, abusive, or mean towards the target (person or group).

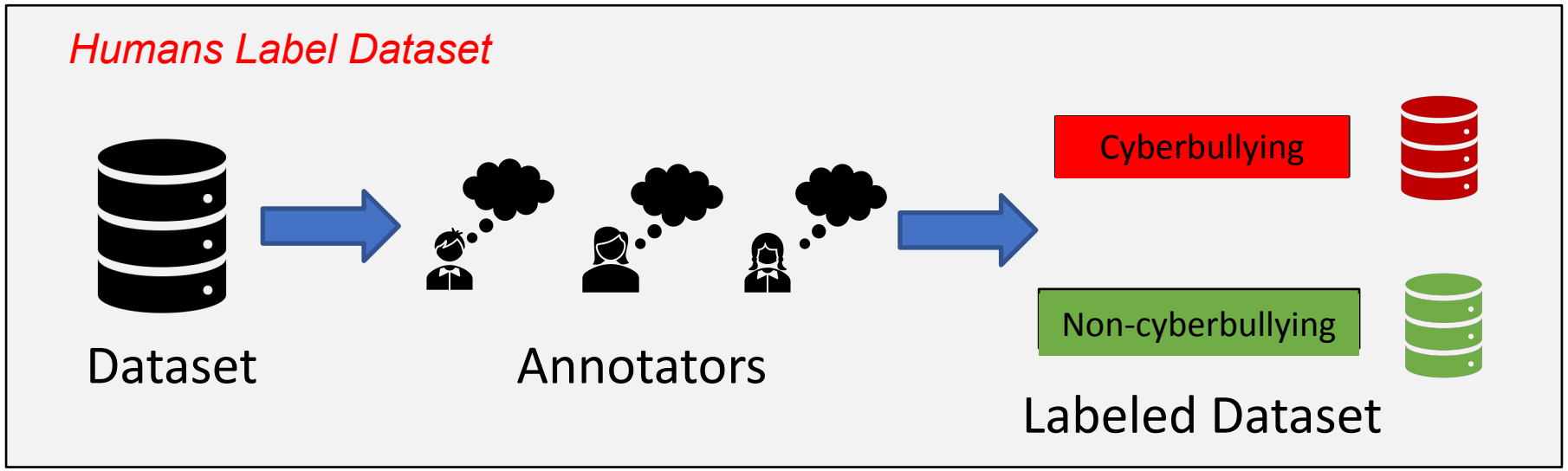
Problem of cyberbullying on Images

Threatening images like these can be sent to a victim to intimidate.

Detecting such content helps in preventing negative health effects on victims

Detecting Cyberbullying on Images

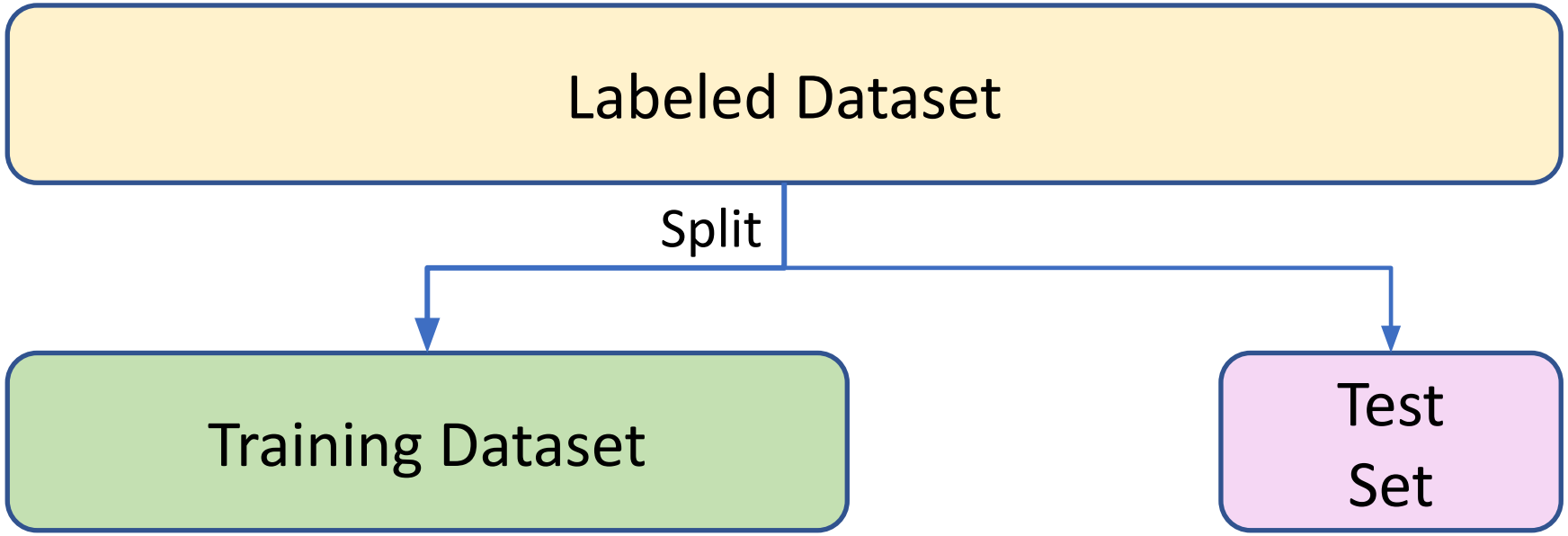
Annotation/Labeling



For easy annotation and arithmetic, we can label cyberbullying as 1 and non-cyberbullying as 0

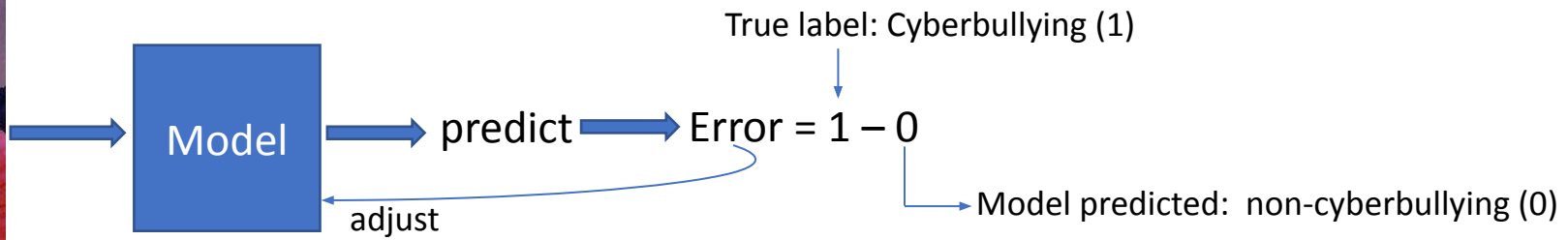
Detecting Cyberbullying on Images

Training/Testing set

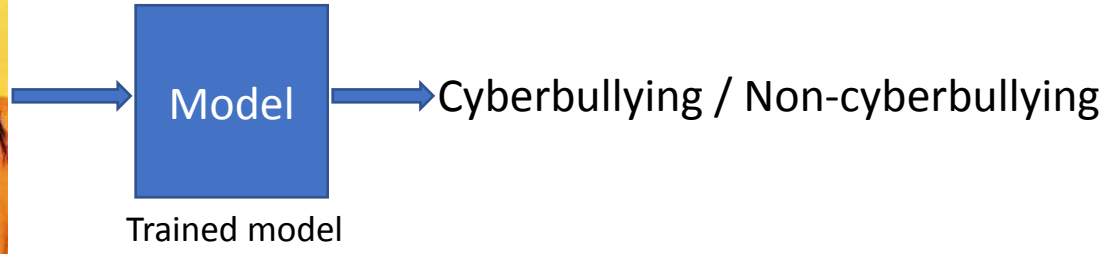


Detecting Cyberbullying on Images

Training – pass images and check how well the model is doing and adjust the model



Pass images through a trained model for detection



Adversarial Examples

Machine learning models are susceptible to adversarial examples. Adversarial examples are input data that have been changed slightly with the intention of causing the machine learning classifier to misclassify.

Usually, the changes are imperceptible to humans, yet machine learning classifiers make mistakes.

This is a security issues because real-world systems based on machine learning can be forced to produce incorrect output without the attacker having access to the system. for example,

- Video surveillance systems
- Mobile applications for image classification
- Self driving cars
- Robots sensing the real-world through sensors and cameras

Adversarial Attacks

If there exist a machine learning system M and input image I, a clean image example. If I is classified correctly by M i.e., $M(I) = cyberbullying$. Then it is possible to construct another input image I' that is almost indistinguishable from I, known as an adversarial example, that will cause M to misclassify, i.e., $M(I') \neq cyberbullying$ This phenomenon is known as adversarial attack.

$+ \rho$ → Model → Non-cyberbullying

- Small noise is added to an image
- Trained model misclassifies the image

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} .01 & .02 \\ .03 & .04 \end{bmatrix} = \begin{bmatrix} 1.01 & 2.02 \\ 3.03 & 4.04 \end{bmatrix}$$

image | Noise/ perturbation | Adversarial example/ perturbed image

→ Model → Non-cyberbullying

Adversarial Attacks

- Assumptions
  - Black-box attacks
    - Adversarial examples are fed to a targeted model during testing
    - Model architecture, parameters and training procedure is not known by the adversary
  - White-box attacks
    - Adversary knows the model architecture, parameters, training procedure, and training data
- Goals
  - Non-targeted attack – make the model predict the wrong label
  - Targeted attack – make the model predict a specific (target) label that is not the source (original) label

Adversarial Attacks

Let:

$I_c \in \mathbb{R}^m$: Clean image vector;

$\rho \in \mathbb{R}^m$: Perturbation when added to the image I_c fools a classifier

$I_\rho \in \mathbb{R}^m$: Perturbed image vector

ℓ is the true label of the image I_c

How to compute ρ efficiently?

- Fast gradient sign method (FGSM)
- Basic Iterative Method: l_∞ version of Projected Gradient Descent (PGD)

Adversarial Attacks

- Fast gradient sign method (FGSM)
  - Method to effectively compute adversarial perturbation ρ that maximizes the loss of the classifier:

$$\rho = \epsilon \ \text{sign}\left(\nabla_{I_c} J(\theta, I_c, \ell)\right)$$

  - $\nabla J(.,.,.)$ computes the derivative of the cost function used in training the classifier around the model parameters θ w.r.t. Input image I_c. $\text{sign}(.)$ ensures the magnitude of the loss is maximized and ϵ restricts the l_∞ norm of the perturbation.

- Basic Iterative Method: l_∞ version of Projected Gradient Descent (PGD)
  - Extends the FGSM by applying FGSM multiple times with a small step size α and clip pixel values of the image that results from each step to be in the $\epsilon - neighbourhood$ of the original image:

$$I_\rho^0 = I_c$$

$$I_\rho^{i+1} = Clip_{I_c,\epsilon}\{I_\rho^i + \alpha sign(\nabla_{I_c} J(\theta, I_\rho^i, l))\}$$

  - I_ρ^i is the perturbed image at the i^{th} iteration. α is usually 1 and number of iterations is determined by $\min(\epsilon + 4, 1.25\epsilon)$

Lab demonstration

See section 3.2 of lab manual to access the notebook