Adversarial training for blind-spot removal

(HiWi Report)

Rishi Sharma

February 11, 2020

Outline

Adversarial Training

Natural saddle point (min-max) formulation

$$\min_{\theta} \rho(\theta), \text{ where } \rho(\theta) = \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(\theta, x + \delta, y)]$$

- θ : the parameters of the model.
- x : the input to the model.
- y : the target label of the given input.
- $S \subseteq \mathbb{R}^n$: the set of allowed perturbations.
- Two separate problems: Inner Maximization & Outer Minimization

[Towards Deep Learning Models Resistant to Adversarial Attacks (2017), Aleksander Madry et al.]

Adversarial Training

Inner Maximization

$$\max_{\delta \in S} L(\theta, x + \delta, y)]$$

- This optimization tries to generate the adversarial example from the given input.
- This optimization problem can be solved by using techniques such as Fast Gradient Sign Method and Projected Gradient Descent.
- FGSM: $x \leftarrow x + \epsilon sgn(\nabla_x L(\theta, x, y))$.
- PGD: $x^{t+1} \leftarrow \Pi_{x+S}(x^t + \alpha sgn(\nabla_x L(\theta, x, y)))$.

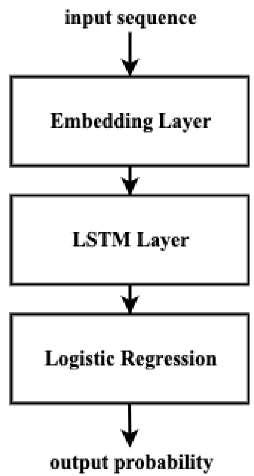
Adversarial Training

Outer Minimization

$$\min_{\theta} \rho(\theta)$$

- This is the learning phase of the adversarial training.
- The network is trained on the dataset augmented with the adversarial examples.

The Network

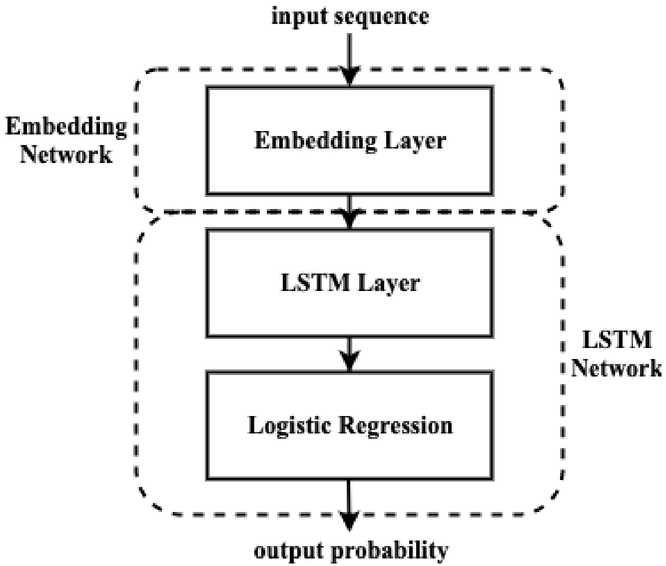


Composition of the Network.

- The Embedding Layer : projects the l-length (padded) input sequence to a sequence of l vectors, each of dimension d.
- The LSTM layer : a feature extraction layer.
- Logistic Regression : predicts the probability of being malicious.

[Predicting Domain Generation Algorithms with Long Short-Term Memory Networks (2016), Endgame, Inc.]

The Network



Separation of the Networks.

- Trained over $\sim 22k$ samples of each non-manipulated benign and malicious samples.
- 5-fold cross-validation.

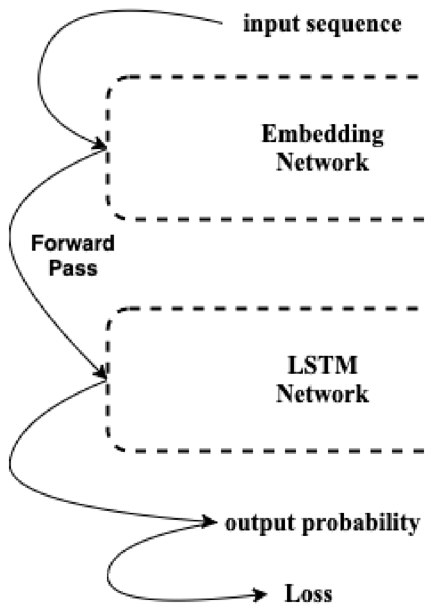
The Network

-	Mean	Stddev
Acc	0.9974	0.0011
FNR	0.0014	0.0010
TNR	0.9961	0.0018
TPR	0.9986	0.0010
FPR	0.0039	0.0018

Table 1: The performance measurements of training the original network.

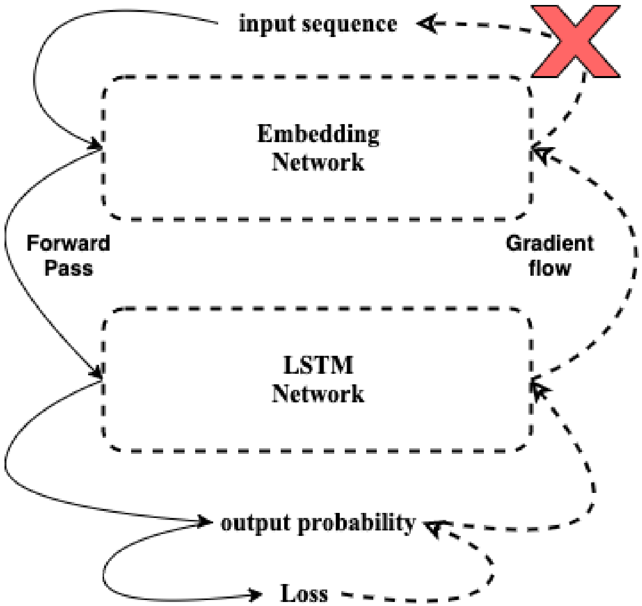
- This training was performed on only valid domain names. We will see this later why?

Non-Differentiable Embedding Layer



A forward pass through the network.

Non-Differentiable Embedding Layer



The gradient flow.

Non-Differentiable Embedding Layer

Problem

The Embedding Layer **selects** a vector corresponding each character in the input sequence.

- The implementations of the Embedding Layer in frameworks access the vector from a table using the character as the index.
- The layer is non-differentiable.

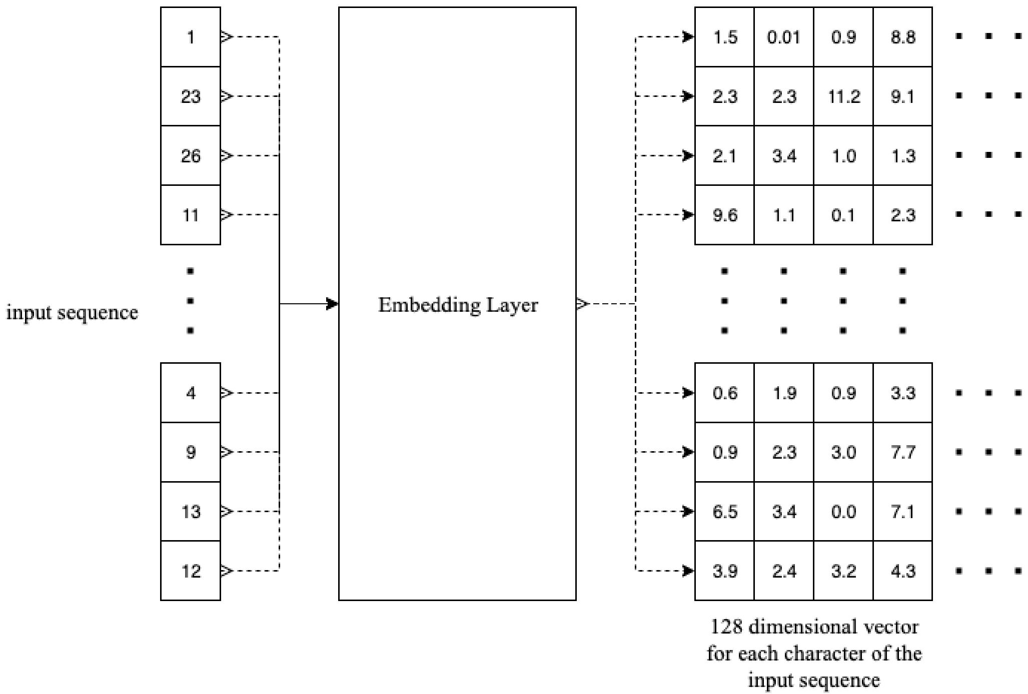
Non-Differentiable Embedding Layer

Problem

The Embedding Layer **selects** a vector corresponding each character in the input sequence.

- The implementations of the Embedding Layer in frameworks access the vector from a table using the character as the index.
- The layer is non-differentiable.
- Two solutions.

Non-Differentiable Embedding Layer



The Embedding Network.

Non-Differentiable Embedding Layer

Emulate Embedding Layer

Use 1D Convolution filters to learn a vector representation of the input sequence elements.

- A very simple network.
- Each Convolution filter learn to predict one of the dimensions of the vector representation.
- This embedding layer is differentiable!

Non-Differentiable Embedding Layer

Emulate Embedding Layer

Use 1D Convolution filters to learn a vector representation of the input sequence elements.

- A very simple network.
- Each Convolution filter learn to predict one of the dimensions of the vector representation.
- This embedding layer is differentiable!
- Con : The accuracy fell down awfully on training the complete network in one pass.

Non-Differentiable Embedding Layer

Emulate Embedding Layer

Use 1D Convolution filters to learn a vector representation of the input sequence elements.

-	Mean	Stddev
Acc	0.9244	0.0388
FNR	0.0329	0.0433
TNR	0.8890	0.0367
TPR	0.9671	0.0433
FPR	0.1109	0.0367

Table 2: Performance while training the network with emulated layer.

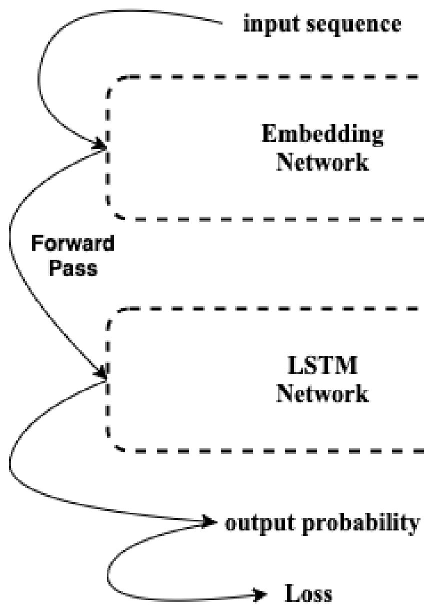
Non-Differentiable Embedding Layer

Invert Embedding Layer

Use a network to get the input sequence back from the embeddings.

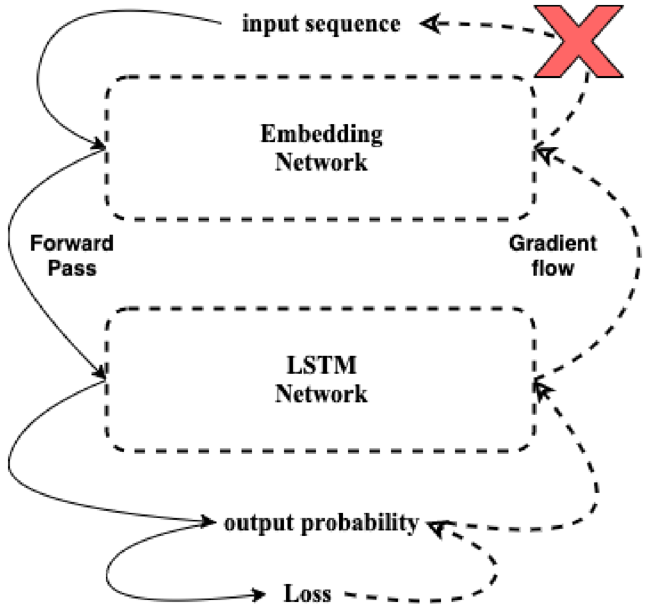
- How does that help us?
- This won't let gradients flow back to the input sequence.
- Let us see how...

Non-Differentiable Embedding Layer

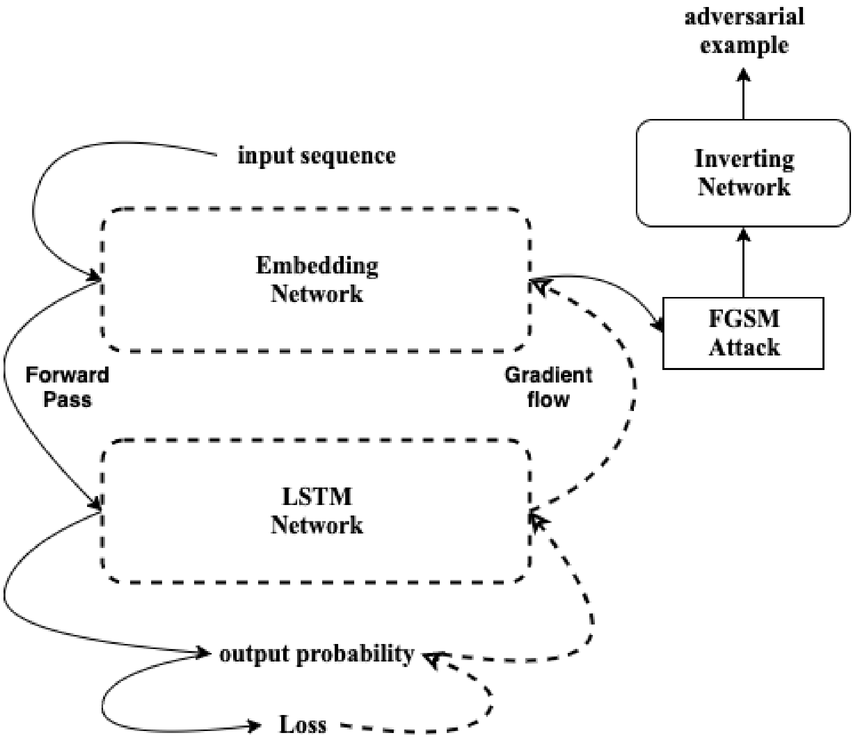


A forward pass through the network.

Non-Differentiable Embedding Layer



Our original setting.



The inverting solution.

Non-Differentiable Embedding Layer

Training the inverting network

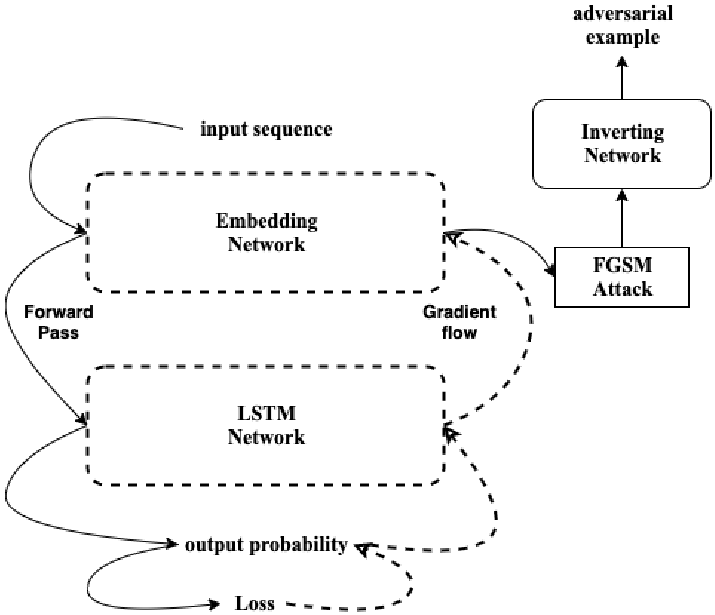
inputs : Output of the Keras embedding layer.
label : The input sequence.
Trained similar to an autoencoder.

Input domain	Inverted domain
amazon.co.de	amazon.co.de
this-is-it-security.rwth	this-is-it-security.rwth
google.com	google.com
gst.gov.in	gst.gov.in
apple.com	apple.com

Table 3: Evaluation of the inverting network.

FGSM attack

$$x \leftarrow x + \epsilon sgn(\nabla_x L(\theta, x, y))$$



Gradient-based Attack

FGSM attack

$$x \leftarrow x + \epsilon sgn(\nabla_x L(\theta, x, y))$$

Keep attacking till probability drops below 10% or max-epochs reached.

Generated adversarial samples

*-9.z07?*hvd].a*

*!n !f7=s*1d/.;*

jf];;g;;0?2;

- *Problem* : None of the adversarial domains are valid.
- *Catch* : The benign dataset was unfiltered.

FGSM attack

- *Problem* : None of the adversarial domains are valid.
- *Catch* : The benign dataset was unfiltered.
- *Solution* : Remove invalid domains from the dataset. Retrain.

Valid generated adversarial samples

rjqqyq4q.net
9cbq48qq.space
wt34h8o0.space
qqqqqqq.net

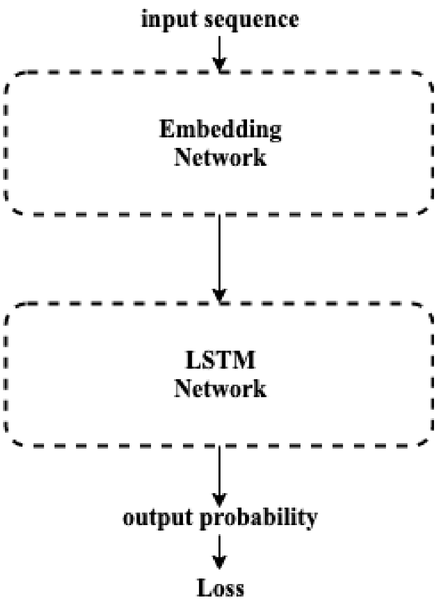
- *Catch* : Keep the top-level domain unchanged.
- *Too much change?* : $x \leftarrow x + \epsilon \nabla_x L(\theta, x, y)$
- *Time/epoch* : $0.7092s$

Iterative Hardening

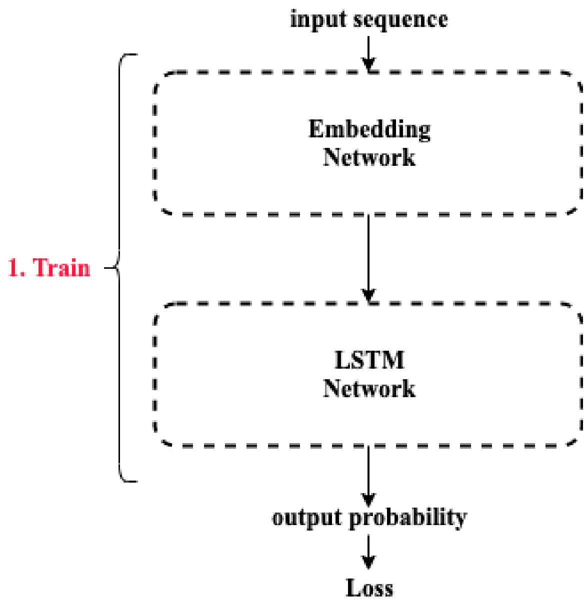
The algorithm

- Train the network.
- *Inner Maximization* : Generate adversarial samples.
- Augment the training set with the adversarial samples.
- *Outer Minimization* : Retrain.

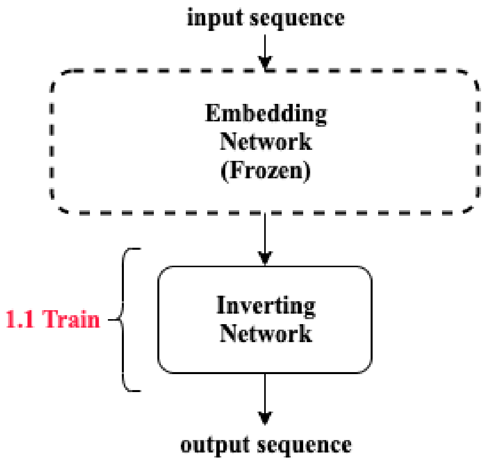
Iterative Hardening



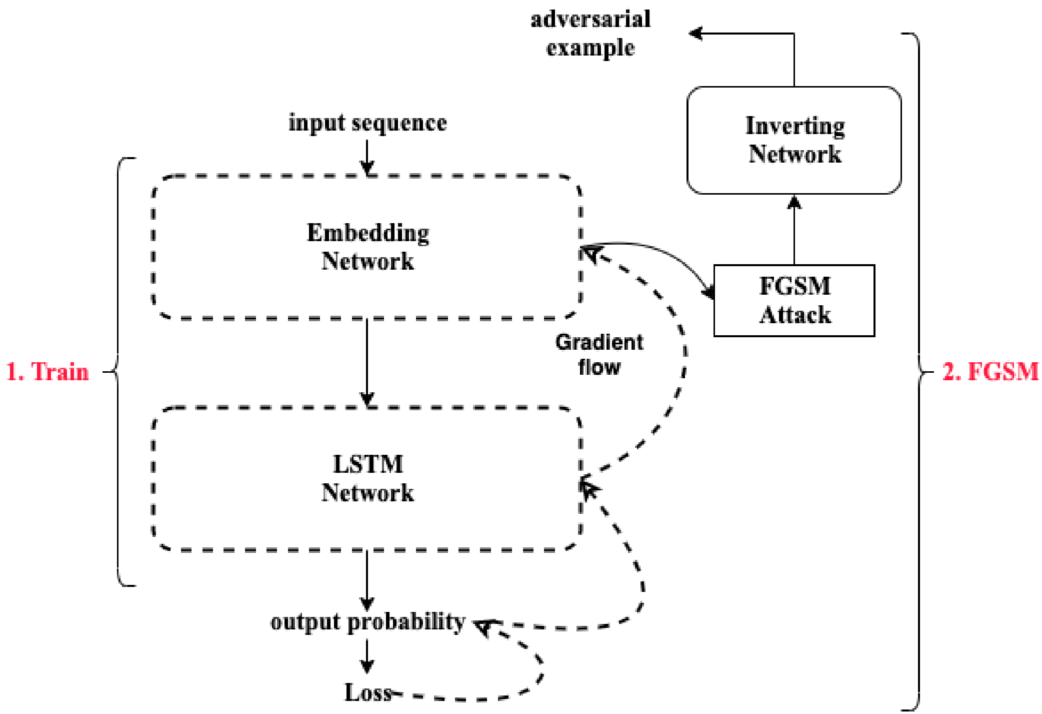
Iterative Hardening



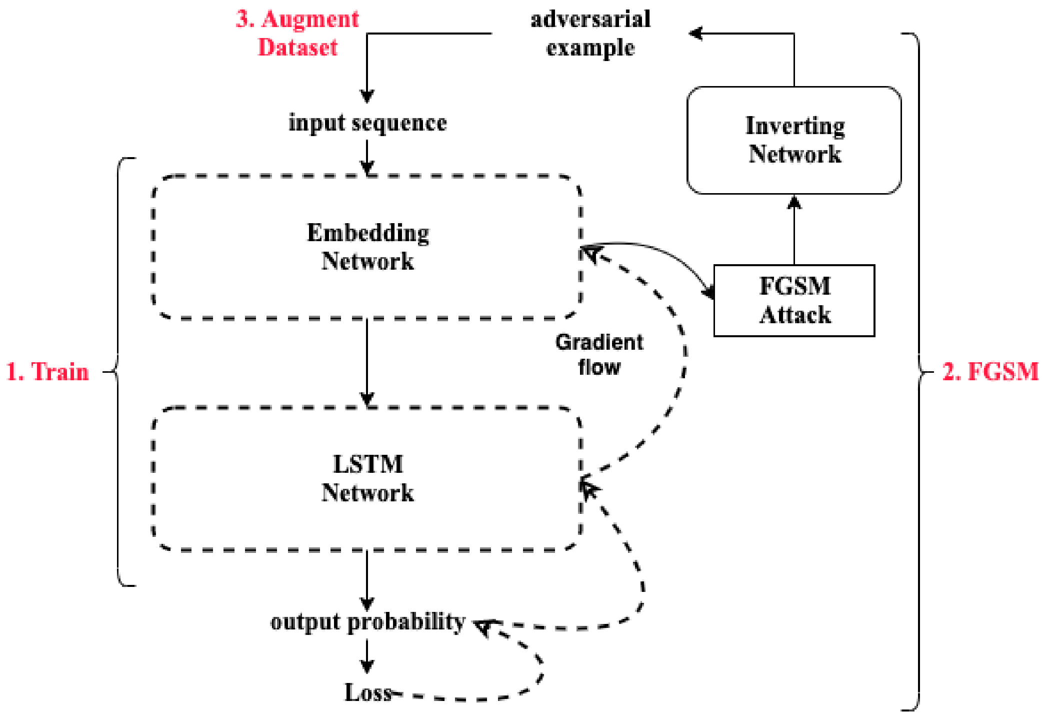
Iterative Hardening



Iterative Hardening



Iterative Hardening



Iterative Hardening

The algorithm

- Train the network.
- *Inner Maximization* : Generate adversarial samples.
- Augment the training set with the adversarial samples.
- *Outer Minimization* : Retrain.
- *Expectation* : If it all works well, FGSM should generate **actually** benign samples.

Remember to freeze the embedding network after the first iteration.

Iterative Hardening

Results

057754af.space

[99.7%]

Results

057754af.space $\xrightarrow{\text{FGSM}}$

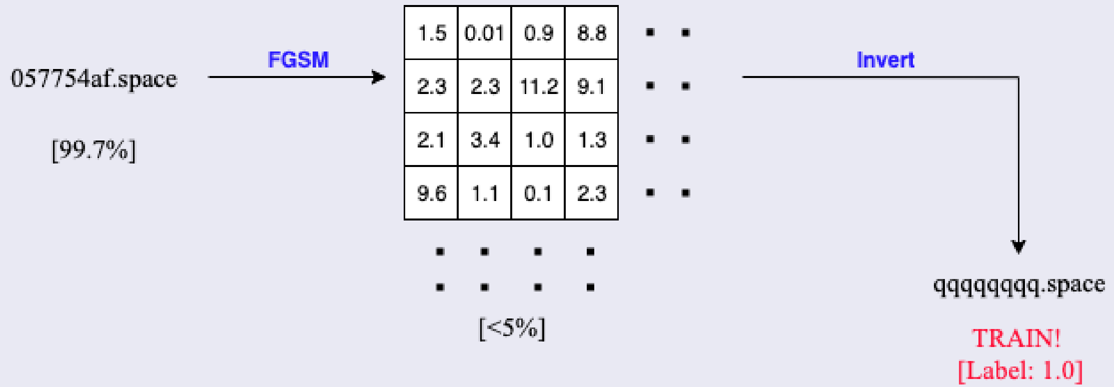
[99.7%]

1.5	0.01	0.9	8.8
2.3	2.3	11.2	9.1
2.1	3.4	1.0	1.3
9.6	1.1	0.1	2.3

[<5%]

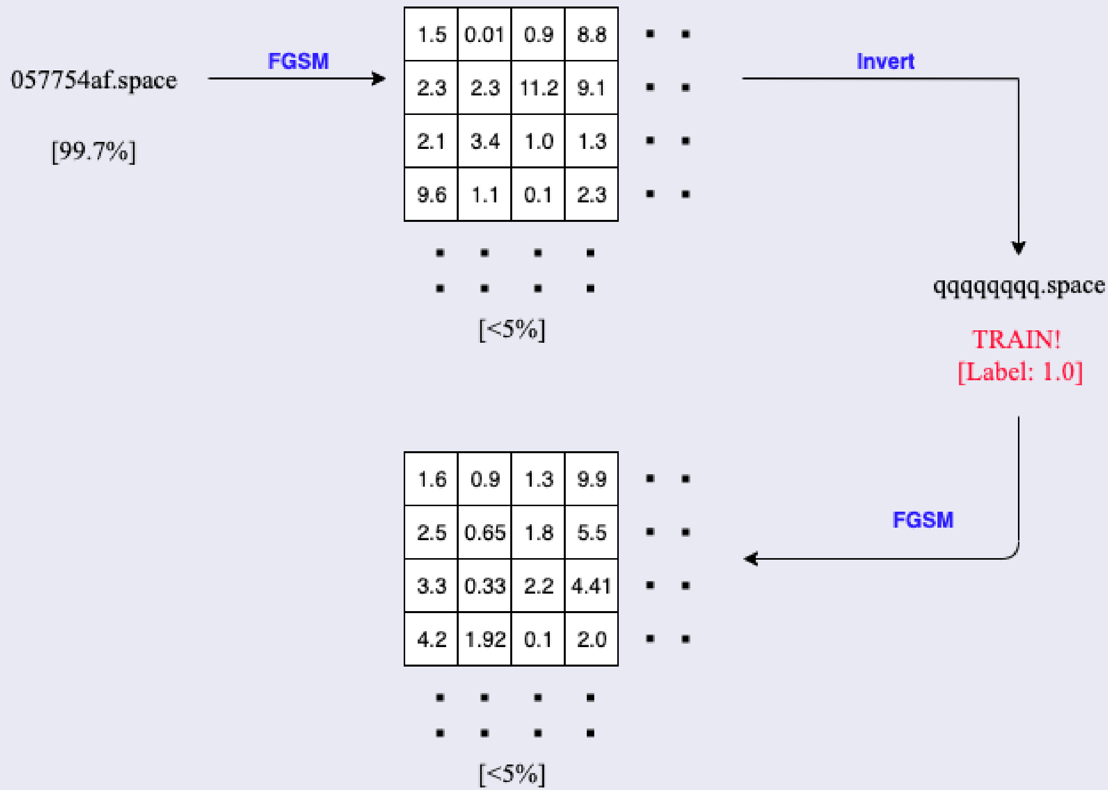
Iterative Hardening

Results



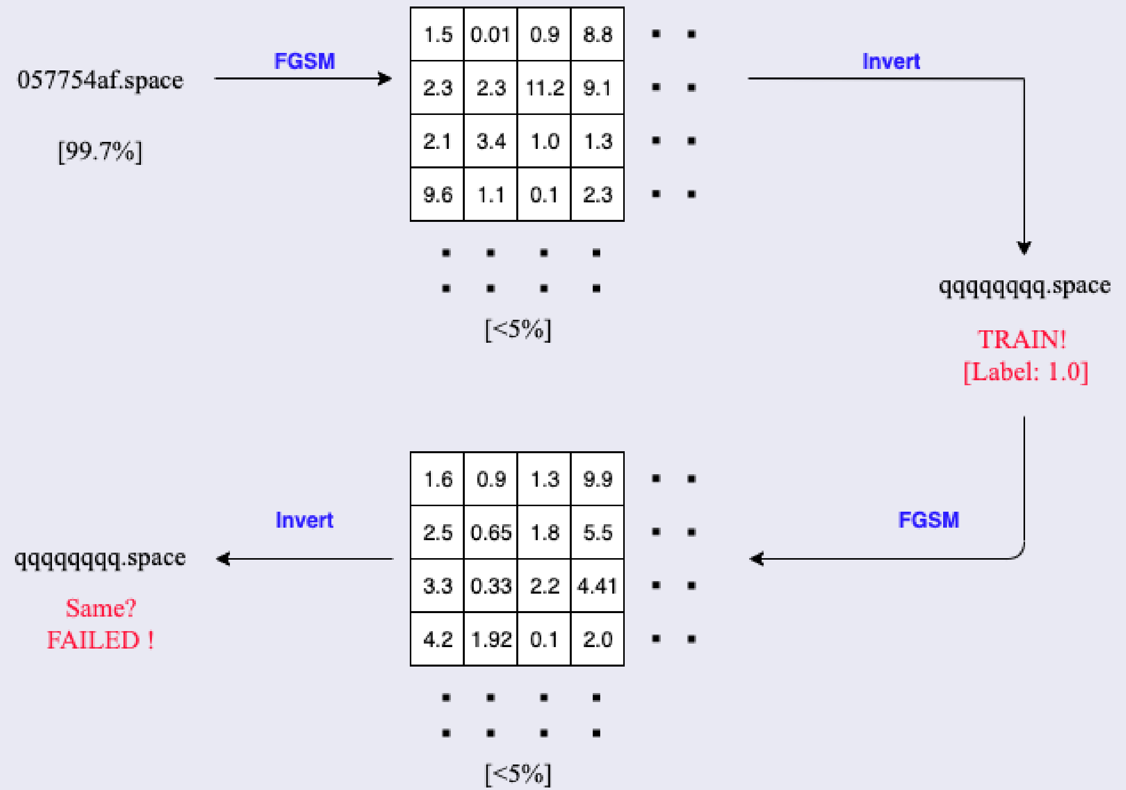
Iterative Hardening

Results



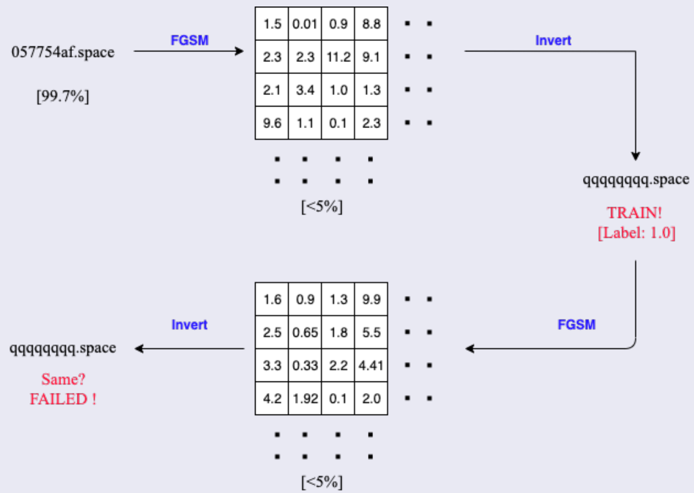
Iterative Hardening

Results



Iterative Hardening

Results



Analysis

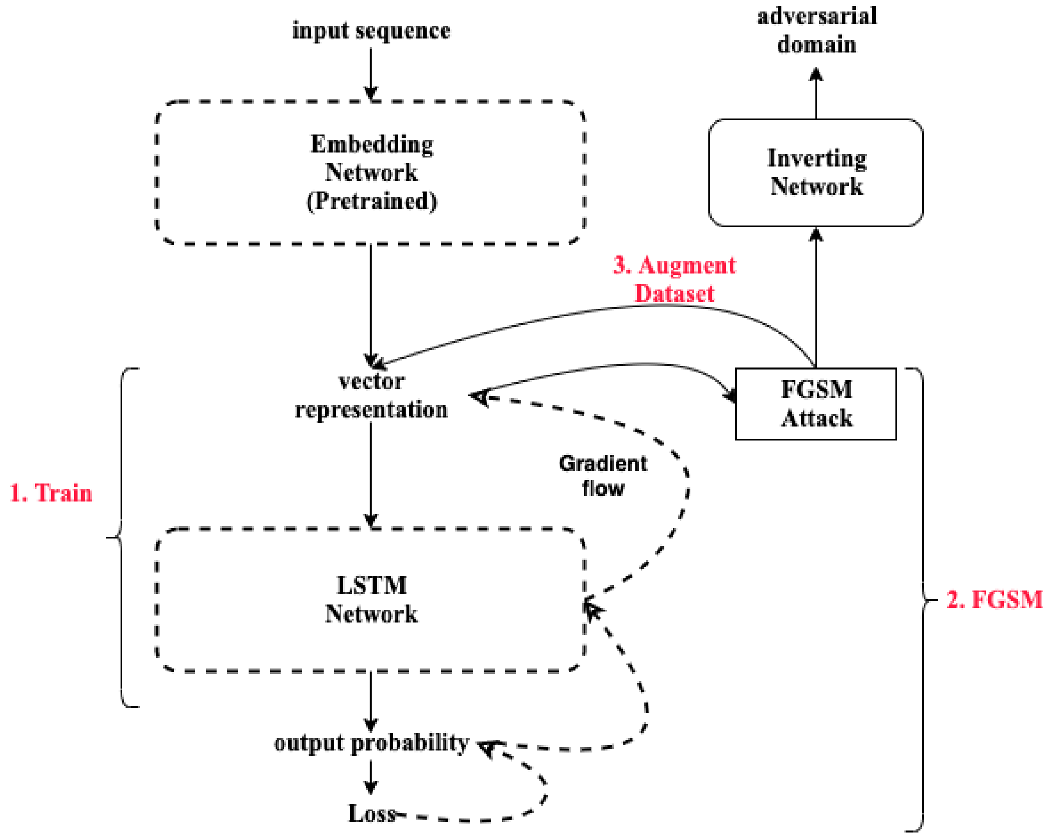
- *Vector Representation* : −0.07421777 0.05503434 0.10850348 ...
- *Mean L2 Distance* : 5.89923604974
- *Baseline Prediction* : qqqqqqq.space [99.8%]

Iterative Hardening over vector representations

The algorithm

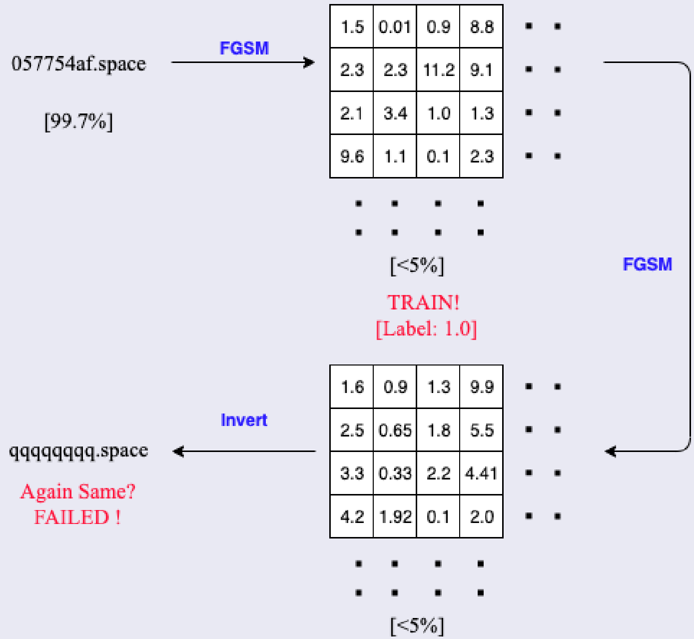
- Train the complete network.
- Create a dataset with vector representations.
- *Inner Maximization* : Generate adversarial vector representations.
- Augment the dataset with the adversarial samples.
- *Outer Minimization* : Retrain only the LSTM Network with this dataset.
- *Expectation* : If it all works well, FGSM should generate **actually** benign samples.

Iterative Hardening over vector representations



Iterative Hardening over vector representations

Results



- *Average Epochs per Attack* : Increases on each iteration
- Harder to find adversaries.

Iterative Hardening

Why gradient based attacks don't work?

- The Embedding Layer **selects** a vector corresponding each character in the input sequence.
- *Discrete* : A character can be represented by a unique vector in the high dimensional space.
- The gradient based attack makes a continuous change in the direction of gradient.
- *Iterative Training in the character space* : Mapping the continuous change to discrete levels can disrupt the attack.
- *Iterative Training in the vector space* : The adversarial vector will never be generated by the embedding network. No useful learning.

Transferring Adversaries

Mike Lorang, in his Master Thesis used transferred adversaries from a network without embedding layer to one with embedding layer.

-	Baseline	CharIterH	VectorIterH
Acc	0.9427	0.9739	0.9010
FNR	0.0573	0.0260	0.0989

Table 4: Evaluation against Transferred FGSM (LSTM).

-	Baseline	CharIterH	VectorIterH
Acc	0.9993	0.9879	0.9538
FNR	0.0007	0.0121	0.0462

Table 5: Evaluation against Transferred FGSM (CNN).

Transferring Adversaries

-	Baseline	CharIterH	VectorIterH
Acc	0.7050	0.8860	0.7650
FNR	0.2950	0.1140	0.2350

Table 6: Evaluation against Hotflip Adversaries (LSTM).

-	Baseline	CharIterH	VectorIterH
Acc	0.5696	0.7264	0.4629
FNR	0.4303	0.2735	0.5370

Table 7: Evaluation against Hotflip Adversaries (CNN).

-	Baseline	CharIterH	VectorIterH
Acc	0.4667	0.6000	0.4667
FNR	0.5333	0.4000	0.5333

Table 8: Evaluation against SeqGAN Adversaries (Very few samples).

Transferring Adversaries

Benign
195.126.129.124.in-addr.arpa
zjekmjf.germanistik.rwth-aachen.de
ejgvgxp.ad.fh-aachen.de
fe-prg007.nos-avg.cz

Hotflip (CNN)
wli-hcg-.de
z0n-e8tzmz7mrby-.be
5jkat2oz5gz8ei2.name
zt-sf–lm.at

SeqGAN
qbutbtwbswul7a6anl.laanwh.ad
e0ehl136oesqe.sfpspeeld.a.th
qbutbtwbswul7a6anl.laanwh.ad
o21pfr2o.e.s.hn

Hotflip (LSTM)
9-qqeidkufm28qd9j1.fr
9tnl777ld53b758.org
e0dbmmgsm2-uav1-.jp
bwh3pku3qm9e7.nz

Table 9: Benign and adversarial domain names.

Future Work

Flow gradients to the inputs without accuracy loss

- Train the original network.
- Generate embeddings
- Train the emulated embedding layer with the generated embeddings as labels.
- Improved the Emulated Embedding Layer?

Projection of adversarial vector representation

- Generate adversarial vector representations.
- Calculate distance of the adversarial vectors to all possible embedding vectors. (L2 Distance)
- Choose the corresponding nearest embedding vectors.
- Projection from continuous to discrete space.

Future Work

Adversarial examples for discrete data / text

- SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient (2017), Lantao Yu et al.
- HotFlip: White-Box Adversarial Examples for Text Classification (2018), Javid Ebrahimi et al.

Try other attacks for discrete data.