Relazione Progetto Cybersecurity
A.A. 2024-25

Violazione dei CAPTCHA: Analisi e Sfruttamento delle Vulnerabilità tramite Machine Learning

Componenti del Gruppo

Leonardo Vorabbi
Matricola: 0001186597

Carlotta Nunziati
Matricola: 0001181860

Email

leonardo.vorabbi@studio.unibo.it

carlotta.nunziati@studio.unibo.it

1 Introduzione

1.1 Cosa sono i CAPTCHA e a cosa servono

I CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) sono una misura di sicurezza utilizzata per stabilire se la richiesta di accedere a un servizio o a una pagina arriva da un utente umano o da un programma informatico. I CAPTCHA vengono utilizzati per impedire ai Web crawler automatizzati, noti anche come bot, di inviare commenti, moduli o spam sui siti Web e per difendersi da minacce online in senso lato (come attacchi DDoS, malvertising o attacchi ransomware). I CAPTCHA richiedono agli utenti, per dimostrare di essere umani, di completare una rapida sfida o attività. Attualmente esistono diversi tipi di CAPTCHA come testuale, audio, basato sul riconoscimento di immagini, logici o puzzle. Google ha introdotto No CAPTCHA reCAPTCHA, un metodo volto a tracciare l'attività dell'utente nella selezione della casella "Non sono un robot" e Invisible reCAPTCHA per migliorare ulteriormente l'esperienza dello stesso, tracciando l'attività in modo invisibile. I CAPTCHA si sono evoluti per tenere il passo con le tecnologie sempre più sofisticate dei bot: infatti, la difficoltà di distinguere tra umani e macchine è destinata a diventare sempre più cruciale con i progressi delle tecnologie di Intelligenza Artificiale.

1.2 Il nostro progetto: overview

Nel nostro progetto ci proponiamo di indagare se e come sia possibile violare i CAPTCHA attraverso il machine learning. In particolare, ci siamo concentrati sulle seguenti tipologie:

- CAPTCHA basati sul riconoscimento di immagini
- CAPTCHA alfanumerici
- CAPTCHA logici

Abbiamo accolto approcci differenti per la risoluzione del problema, confrontando le prestazioni dei nostri modelli (basati su reti neurali convoluzioni) con le prestazioni di modelli pre-addestrati. I dataset di riferimento sono stati scaricati da Kaggle. Per l'utilizzo della GPU ci siamo avvalsi di Google Colab.

2 Modelli progettati

2.1 CAPTCHA basati sul riconoscimento di immagini

2.1.1 Modello basato su reti neurali convoluzionali

Lo script `captcha_immagini.py`, partendo da un dataset scaricato da Kaggle, si occupa di selezionare solo le immagini con estensioni valide (".jpg", ".jpeg", ".png", ".gif") che vengono poi suddivise in set di addestramento e test utilizzando la funzione `train_test_split`. Inoltre, le etichette vengono codificate tramite una funzione che converte i nomi delle cartelle in etichette numeriche.

Il modello è una rete neurale convoluzionale (CNN) composta da strati convoluzionali, di pooling, di normalizzazione, e densamente connessi. Ogni strato convoluzionale estrae caratteristiche dall'immagine, mentre gli strati di pooling riducono la dimensionalità. Il modello si conclude con strati completamente connessi che producono la probabilità di ciascuna classe (un'etichetta del CAPTCHA). L'ottimizzazione è realizzata tramite l'ottimizzatore Adam e la funzione di perdita utilizzata è la `sparse categorical crossentropy`.

Successivamente il modello viene addestrato per un massimo di 20 epoche. Dopo l'allenamento, vengono visualizzate alcune predizioni fatte dal modello su un campione di immagini del test set, confrontando le etichette predette con quelle reali.

Purtroppo, l'accuracy del modello si aggira solamente intorno al 50%, il che indica che il modello non ha ancora una buona capacità di generalizzare sui dati di test. Questo potrebbe essere dovuto alla complessità del CAPTCHA e la quantità limitata di dati di addestramento.

Figura 1: Esempio di immagine CAPTCHA riconosciuta dal modello.

2.1.2 Modello pre-addestrato

Abbiamo utilizzato il modello YOLOv3 pre-addestrato sul dataset COCO, uno dei più noti benchmark per il rilevamento di oggetti. Questo dataset contiene oltre 330.000 immagini con circa 80 categorie di oggetti annotati, come persone, animali, veicoli e oggetti quotidiani. L'immagine preprocessata viene passata al modello YOLO che: converte l'immagine in un formato compatto (blob), applica la rete neurale per rilevare gli oggetti e ritorna le classi identificate e le relative confidenze.
L'accuracy del modello, come dimostrano gli studi condotti, si aggira tra 80 e 90%.

Approfondimento: YOLO
YOLO (You Only Look Once) è un modello avanzato di rilevamento di oggetti, noto per la sua elevata velocità e accuratezza. Il rilevamento di oggetti è un'attività di visione artificiale che comporta l'identificazione e la localizzazione di oggetti in immagini o video. L'immagine di input viene preprocessata, successivamente, viene elaborata da una rete neurale convoluzionale profonda (CNN) che suddivide l'immagine in una griglia. Ogni cella della griglia è responsabile di rilevare gli oggetti il cui centro cade al suo interno, prevedendo le coordinate del bounding box, un punteggio di confidenza che indica la probabilità che la cella contenga un oggetto, e la categoria dell'oggetto rilevato, come "persona" o "auto". Grazie alla sua capacità di analizzare simultaneamente più aree dell'immagine, YOLO può identificare e localizzare diversi oggetti in una singola operazione. Per ridurre i risultati ridondanti, utilizza tecniche di post-elaborazione, come la soppressione non massima (Non-Maximum Suppression), che elimina le sovrapposizioni mantenendo solo le previsioni più attendibili. YOLO rappresenta insomma un modello di riferimento nel rilevamento di oggetti, grazie alla sua capacità di generalizzare su immagini non viste durante l'addestramento.

2.2 CAPTCHA alfanumerici

2.2.1 Modello basato su reti neurali convoluzionali

Dopo aver scaricato un dataset di CAPTCHA alfanumerici da Kaggle, implementiamo una funzione di preprocessing che include il ridimensionamento delle immagini, la conversione in scala di grigi e la normalizzazione, preparando i dati per il modello di machine learning. Il modello OCR (optical character recognition) è costituito da strati convoluzionali per l'estrazione delle caratteristiche, seguiti da livelli di pooling per ridurre la dimensionalità e da un livello di reshaping che adatta i dati per l'elaborazione sequenziale.

Utilizziamo strati LSTM bidirezionali per catturare la sequenza temporale e il contesto dei caratteri, e applichiamo la `perdita CTC (Connectionist Temporal Classification)` per la gestione della decodifica delle sequenze di testo.

Il dataset viene suddiviso in set di addestramento, validazione e test, associando ciascuna immagine con la sua etichetta testuale per un apprendimento supervisionato. Dopo aver scelto l'ottimizzatore Adam e configurato la funzione di perdita CTC, addestriamo il modello per migliorare la sua accuratezza nel riconoscimento. Al termine, valutiamo il modello su un test set per verificarne le capacità di generalizzazione su nuovi CAPTCHA, decodificando e confrontando le predizioni con i testi originali.

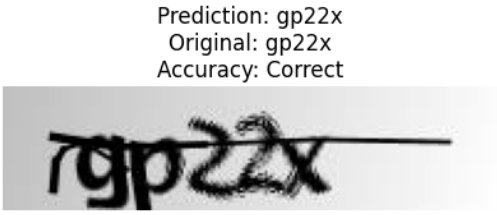

Figura 2: Esempio di CAPTCHA alfanumerico riconosciuto dal modello.

```
Accuracy: 93.75%
```

Figura 3: Accuracy del modello basato su CNN.

Approfondimento: CTC (Connectionist Temporal Classification)
La perdita CTC (Connectionist Temporal Classification) è una funzione di perdita progettata per gestire compiti di riconoscimento sequenziale in cui le previsioni di output possono avere lunghezze variabili rispetto agli input. L'obiettivo principale della perdita CTC è di consentire al modello di mappare un input sequenziale (come nel nostro caso un'immagine) a una sequenza di output (come del testo) senza dover allineare esattamente ogni elemento dell'output con una posizione fissa nell'input.

La CTC Loss quindi si occupa di ottimizzare il modello durante l'addestramento per mappare sequenze di input variabili su output discreti, come parole o frasi. La CTC Loss valuta l'accuratezza delle predizioni confrontando le possibili allineamenti tra l'output predetto e quello desiderato, guidando così il processo di apprendimento e miglioramento del modello.

Prendendo come esempio un'immagine di un CAPTCHA con la scritta "abc", durante l'addestramento, il modello può produrre una sequenza predetta più lunga, come "a_bc", dove "_" rappresenta uno spazio. La CTC considera valide tutte le possibili sequenze che si possono ridurre ad "abc" eliminando gli spazi.

2.2.2 Modello pre-addestrato

Dopo aver scaricato un dataset di CAPTCHA alfanumerici dalla piattaforma Kaggle, implementiamo una classe (CAPTCHADataset) per caricare le immagini, elaborarle e associarle al label corrispondente. Carichiamo il modello pre-addestrato `microsoft/trocr-base-stage1`, un modello Transformer per OCR: modello che, essendo pre-addestrato su grandi quantità di dati, può essere adattato a compiti specifici come la risoluzione di CAPTCHA tramite fine-tuning. Usiamo TrOCRProcessor per convertire le immagini del dataset in un formato comprensibile per il modello. Dividiamo il dataset in train, validation e test set. Inizializziamo il modello TrOCR con i pesi pre-addestrati e fissiamo i criteri di valutazione (valuteremo il modello sul Character Error Rate e sull'Accuracy). Addestriamo in modo supervisionato il modello sul nostro dataset specifico e aggiorniamo i pesi per adattarlo al nostro compito durante il fine-tuning.

[1498/1498 26:03, Epoch 7/7]

Epoch	Training Loss	Validation Loss	Cer	Accuracy
1	0.792300	1.224136	0.330841	0.669159
2	0.978300	0.839706	0.289720	0.710280
3	1.092200	0.711451	0.351402	0.648598
4	0.589900	0.688335	0.224299	0.775701
5	0.370100	0.506401	0.069159	0.930841
6	0.320600	0.463997	0.057944	0.942056
7	0.268100	0.454615	0.067290	0.932710

Figura 4: Evoluzione dei valori di addestramento con l'avanzamento delle epoche.

Infine, valutiamo le prestazioni sul test set, verificando la capacità di generalizzare su dati mai visti:

```
Accuracy: 96.72%
```

Figura 5: Accuracy del modello pre-addestrato.

Approfondimento: TrOCR
TrOCR (Transformer-based Optical Character Recognition) è un modello sviluppato da Microsoft per il riconoscimento del testo nelle immagini. Il modello TrOCR è un modello encoder-decoder, costituito da un image Transformer come encoder e da un text Transformer come decoder. L'image encoder è stato inizializzato dai pesi di BEiT (questo consente al modello TrOCR di comprendere in modo approfondito le caratteristiche visive delle immagini contenenti testo), mentre il text decoder è stato inizializzato dai pesi di RoBERTa (questo consente al modello di generare trascrizioni del testo estratto dalle immagini).

2.3 CAPTCHA logici

2.3.1 Modello basato su reti neurali convoluzionali

Dopo aver scaricato un dataset di CAPTCHA logici dalla piattaforma Kaggle, implementiamo una funzione per il preprocessing delle immagini e definiamo un modello che combina strati convoluzionali (responsabili dell'estrazione delle caratteristiche spaziali delle immagini) e pooling (responsabili della riduzione della dimensione spaziali delle stesse), dropout (utilizzati per prevenire l'overfitting), flatten (strato che converte la matrice multidimensionale prodotta dagli strati convoluzionali e di pooling in un vettore monodimensionale che può essere passato a uno strato denso) e dense (strato utilizzato per la classificazione finale, responsabile dell'assegnazione delle probabilità).

Dividiamo il dataset in train, validation e test set in modo da correlare le immagini alle rispettive etichette per un apprendimento di tipo supervisionato. Dopo aver definito l'ottimizzatore (Adam) e la funzione di perdita (categorical crossentropy) procediamo all'addestramento.

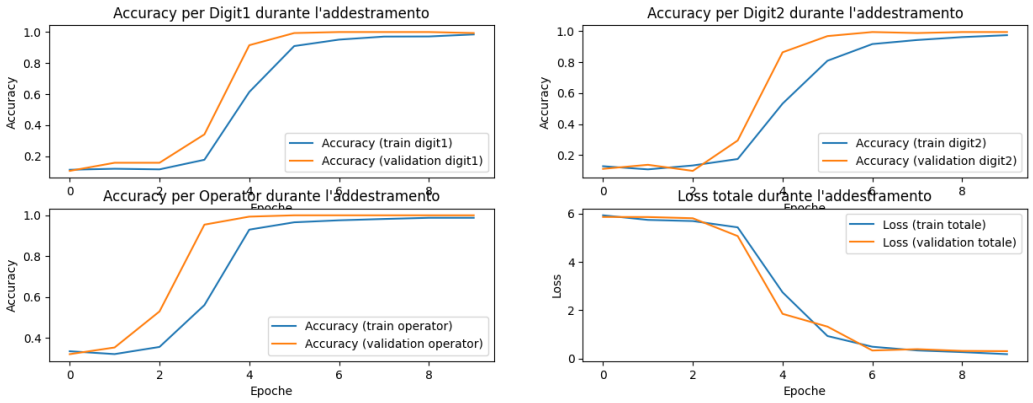

Figura 6: Grafici di prestazione del modello basato su CNN.

Valutiamo infine le prestazioni sul test set, verificando la capacità di generalizzare su dati mai visti.

```
Accuracy: 96.70%
```

Figura 7: Accuracy del modello basato su CNN.

2.3.2 Modello pre-addestrato

Per la risoluzione dei CAPTCHA logici, abbiamo utilizzato lo stesso procedimento descritto a 2.2.2 ma utilizziamo un numero di epoche inferiore dati i risultati in output.

[921/921 10:39, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Cer	Accuracy
1	0.366700	0.365756	0.002179	0.997821
2	0.265000	0.307918	0.008715	0.991285
3	0.222900	0.317449	0.002179	0.997821

Figura 8: Evoluzione dei valori di addestramento con l'avanzamento delle epoche.

Prestazioni sul test set:

```
Accuracy: 100.00%
```

Figura 9: Accuracy del modello pre-addestrato.

3 Simulazione

3.1 Simulazione di un'applicazione a contesti reali

Il primo testing dei modelli progettati, come già mostrato in precedenza, è stato condotto direttamente durante la loro creazione, negli script di costruzione e addestramento. Tuttavia, abbiamo successivamente sviluppato degli script per simulare un utilizzo più o meno realistico dei modelli e mettere in evidenza le potenziali vulnerabilità di alcuni tipi CAPTCHA. Per raggiungere questo obiettivo, è stata creata una pagina web che visualizza un'immagine casuale del dataset di CAPTCHA logici e verifica se l'input dell'utente corrisponde correttamente al risultato dell'equazione mostrata nell'immagine.

Utilizzando una sonda Selenium, è stato possibile automatizzare il processo di estrazione e analisi delle immagini CAPTCHA. Selenium è stato impiegato per simulare il comportamento di un utente reale: ha navigato nella pagina, acquisito l'immagine, e passato il contenuto al modello di decodifica. Siamo in questo modo riusciti a mostrare il potenziale utilizzo pratico dei modelli mettendo al contempo in luce le vulnerabilità di alcuni tipi di CAPTCHA.

4 Istruzioni

Per testare autonomamente gli script creati è sufficiente clonare il repository GitHub https://github.com/leovora/captcha_solver.git .

4.1 Struttura del progetto

```
captcha_solver
├── captcha_alfanumerici   -> contiene file per testare i modelli alfanumerici (demo.py, demo_preaddestrato.py)
├── captcha_immagini       -> contiene file per testare i modelli sulle immagini (demo.py, demo_preaddestrato.py)
├── captcha_logici         -> contiene file per testare i modelli logici (demo.py, demo_preaddestrato.py)
├── modelli                -> contiene gli script per la creazione e l'addestramento dei modelli
```

Figura 10: Organizzazione dei folder del progetto

4.2 Istruzioni per l'esecuzione

4.2.1 Preparazione del progetto

Prima di poter eseguire il codice è necessario scaricare e posizionare nella giusta directory alcune componenti, troppo pesanti per la consegna su GitHub.

Tutte le risorse sono disponibili al link OneDrive:
`https://liveunibo-my.sharepoint.com/:f:/g/personal/leonardo_vorabbi_studio_unibo_it/Eg93IJLyrzZAhrmpnuKTP48BmB3-RgeutZaYUDrtymgjkg?e=1iYQcd`

Il contenuto dei vari dataset andrà posizionato nella relativa directory del progetto, in cui sono già presenti le cartelle vuote con il corrispettivo nome. Inoltre, i file *model.safetensors* andranno posizionati nella directory del modello preaddestrato.

Esempio:
Path OneDrive: `Datasets/Captcha_alfanumerici/model.safetensor`
Path Progetto: `Cybersecurity-projects-2024/captcha_alfanumerici/modello_preaddestrato/model.safetensor`

Lo stesso vale per il file *yolov3.weights* in `Datasets/Captcha_immagini` che andrà posizionato in `Cybersecurity-projects-2024/captcha_immagini/yolo`

4.2.2 Esecuzione del codice

Per eseguire le demo e testare i modelli, è necessario avviare un server web locale utilizzando Python:

```
python3 -m http.server 8000
```

In un secondo terminale si possono eseguire gli script di test. In particolare, ogni cartella "captcha_..." contiene i file:

- demo.py : per testare i nostri modelli
- demo_preaddestrato.py : per testare i modelli pre-addestrati

Nella cartella "modelli" sono invece contenuti gli script che creano e addestrano i vari modelli.

4.3 Requisiti

Sono necessarie le seguenti dipendenze per eseguire i vari script:

```
pip install scikit-image scikit-learn tensorflow torch torchvision
torchaudio matplotlib Pillow opencv-python selenium transformers
-q datasets jiwer evaluate wandb pandas pydot graphviz
```

Si noti che il progetto è abbastanza pesante (più di 10GB), è quindi consigliato scaricarlo su un supporto di memoria esterna.

5 Conclusioni

5.1 Riflessioni sui diversi approcci utilizzati

Nel nostro studio, abbiamo esplorato diverse strategie di apprendimento automatico per affrontare il problema del riconoscimento dei CAPTCHA, attraverso l'uso di modelli pre-addestrati e non. La scelta di modelli pre-addestrati si è rivelata vantaggiosa, in quanto questi modelli, essendo già stati allenati su grandi volumi di dati, possiedono una rappresentazione delle caratteristiche generali che facilita l'adattamento al nostro specifico problema. L'uso di modelli pre-addestrati ha permesso di ottenere migliori prestazioni in termini di accuratezza delle previsioni in quanto dimostrano di beneficiare di un buon livello di generalizzazione. Nonostante i modelli pre-addestrati abbiano mostrato una maggiore accuratezza, i modelli non pre-addestrati hanno comunque prodotto risultati soddisfacenti per il compito proposto.

Per riassumere le nostre considerazioni, possiamo affermare ciò che segue: quando il dataset è limitato o non sufficientemente variegato, i modelli pre-addestrati tendono a generalizzare meglio e a essere meno suscettibili al sovraccarico del modello, rispetto ai modelli non pre-addestrati. Anche se i modelli addestrati da zero richiedono più risorse computazionali e una maggiore quantità di dati per evitare l'overfitting, ci hanno permesso di comprendere meglio le sfide legate al training sui nostri specifici dati.

5.2 Limitazioni del nostro studio

Sicuramente le limitazioni del nostro studio si possono riassumere come segue:

- l'impossibiltà di reperire gratuitamente ampi dataset variegati su cui addestrare i nostri modelli: i modelli, sebbene accurati per i dati utilizzati, potrebbero non generalizzare altrettanto bene su altre tipologie di input non incluse durante il training

- la disposizione di una GPU limitata
- alcuni CAPTCHA semplici sono facilmente decifrabili, ma sistemi più avanzati stanno diventando più resistenti agli attacchi automatizzati

5.3 Sviluppi futuri

Interrogandoci su quali possano essere gli sviluppi futuri del nostro progetto, riteniamo interessante dedicare il nostro impegno futuro a ciò che segue:

- Reperire o realizzare noi stessi dataset maggiormente variegati per far apprendere ai nostri modelli nuove caratteristiche
- Migliorare il modello per la risoluzione di CAPTCHA basati sul riconoscimento di immagini, la cui accuracy ha mostrato buone possibilità di miglioramento
- Implementare nuovi modelli per la risoluzione di CAPTCHA che in questo progetto non abbiamo affrontato: come, ad esempio, quelli basati su puzzle
- Confrontare l'utilizzo dei modelli pre-addestrati da noi adottati con altri, che non abbiamo considerato in questo studio, al fine di comprendere se possano dimostrarsi maggiormente performanti
- Applicare i nostri modelli a contesti reali

5.4 Conclusioni finali

Nel corso dello studio, oltre ad analizzare l'efficacia di modelli pre-addestrati e non, abbiamo anche affrontato le vulnerabilità di alcuni tipi di CAPTCHA. Questi sistemi, progettati per differenziare l'accesso umano da quello automatizzato, spesso presentano debolezze che un modello ben addestrato può sfruttare per decifrare correttamente le sfide. La nostra esplorazione ha messo in luce come CAPTCHA semplici o con pattern prevedibili possano essere attaccati più facilmente, evidenziando la necessità di meccanismi di protezione più sofisticati.

È importante notare che l'adozione di sistemi CAPTCHA più sicuri e avanzati è già in atto, con approcci adattivi e dinamici progettati per resistere meglio agli attacchi automatizzati. Questi sistemi possono includere CAPTCHA che variano in complessità visiva e contestuale, garantendo una maggiore imprevedibilità. Parallelamente, la ricerca nell'ambito del machine learning orientato al riconoscimento dei CAPTCHA sta contribuendo allo sviluppo

di nuove strategie difensive, migliorando così la sicurezza delle interazioni digitali e l'affidabilità dei sistemi di verifica.

Bibliografia

- https://medium.com/@natsunoyuki/ocr-with-the-ctc-loss-efa62ebd8625
- Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2021). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2109.10282
- Wang, Zhong, and Peibei Shi. "CAPTCHA recognition method based on CNN with focal loss." Complexity 2021.1 (2021): 6641329.
- Kopp, Martin, Matej Nikl, and Martin Holena. "Breaking CAPTCHAs with Convolutional Neural Networks." ITAT. 2017.
- Plesner, A., Vontobel, T., & Wattenhofer, R. (2024). Breaking reCAPTCHAv2. arXiv.Org. https://doi.org/10.48550/arxiv.2409.08831