Elaborazione di Segnali Multimediali
a.a. 2020/2021

Creazione di attacchi avversari

L.Verdoliva, D.Cozzolino

Di seguito vi descriviamo come generare esempi avversari tramite il toolbox Foolbox avendo il modello in Keras della rete addestrata. Per poter usare questo toolbox in COLAB bisogna installarlo tramite la seguente istruzione:

```
!pip instal foolbox=3.3.1
```

Se volete lavorare in locale, potete anche installare il toolbox Foolbox nell'environment del corso scrivendo il seguente comando nel Prompt di Anaconda:

```
conda install -n Corso -c conda-forge foolbox==3.3.1
```

Dopo l'installazione, la prima operazione da fare è convertire il modello della rete da Keras a Foolbox, tramite la seguente istruzione:

```
from foolbox.models import TensorFlowModel
model_foolbox = TensorFlowModel(model_keras, bounds=(vmin,vmax))
```

dove `model_keras` è il modello addestrato in keras e `bounds=(vmin,vmax)` specifica la dinamica delle immagini che la rete accetta in ingresso. Prima di eseguire l'attacco calcolate la prestazioni della rete sul batch di immagini che volete attaccare, considerando `x_true` il batch di immagini in formato Channels-Last e `y_true` le relative etichette:

```
y_pred = model_keras.predict(x_true)
y_pred = np.argmax(y_pred, -1)
acc = np.mean(y_true==y_pred) % Accuratezza
```

Come attacco provate il Fast Gradient Signed Method (FGSM) [1]. Questo è un attacco estremamente semplice in cui tramite back-propagation viene calcolato il gradiente dell'uscita della rete rispetto all'immagine in ingresso. Poi, in base al segno del gradiente, viene sommata o sottratta una quantità fissata ϵ ad ogni pixel dell'immagine. Il valore di ϵ regola la quantità di distorsione introdotta. Create la funzione per eseguire l'attacco FGSM:

```
from foolbox.attacks import FGSM
attack = FGSM()
```

e poi eseguite l'attacco:

```
from tensorflow import convert_to_tensor
preclip, x_advs, res = attack(model_foolbox,
                              convert_to_tensor(x_true),
                              convert_to_tensor(y_true), epsilons=eps)
x_advs = x_advs.numpy()
```

Notate che sia le immagini che le etichette prima di eseguire l'attacco vanno convertite da `numpy` a `tensorflow` con la funzione `tensorflow.convert_to_tensor` e il risultato va riconvertito in `numpy` eseguendo il metodo omonimo. Alla funzione `attack` vanno forniti 4 ingressi: la rete da attaccare, le immagini da modificare, le relative etichette e l'`epsilon` che indica il livello di distorsione da inserire. Come risultato della funzione `attack` abbiamo: le immagini modificate, le immagini modificate riportate nella dinamica corretta e un vettore che indica se l'attacco ha avuto successo per ogni immagine. Adesso valutate la distorsione introdotta in termini di MSE e le prestazioni della rete sulle immagini attaccate:

```
y_pred = model_keras.predict(x_advs)
y_pred = np.argmax(y_pred, -1)
acc_advs = np.mean(y_true==y_pred) % Accuratezza dopo l'attacco
mse = np.mean((x_true-x_advs)**2)  % MSE
```

Per utilizzare un altro metodo di attacco basta cambiare la definizione della funzione `attack`, l'elenco degli attacchi possibili è riportato alla pagina:
`https://foolbox.readthedocs.io/en/v3.3.1/modules/attacks.html`

Riferimenti bibliografici

[1] I.J. Goodfellow and J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," International Conference on Learning Representations (ICLR), 2015