ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual neTworks

Corso di *Digital adaptive circuits and learning systems*

Università Politecnica delle Marche

Giulia Temperini

1 Introduzione

Negli ultimi decenni si è assistito a un notevole progresso nei sistemi di autenticazione biometrica vocale basati sulla tecnologia ASV (*Automatic Speaker Verification*). Con quest'ultimo termine si intende indicare un sistema che estrae le caratteristiche vocali di un individuo per stabilirne l'identità, imponendo vincoli di vocabolario fissi(dipendenti dal testo) o in modo dinamico(indipendente dal testo)[1]. Un tipico sistema ASV prevede due fasi: una registrazione offline nella quale viene addestrato un modello di speaker target utilizzando le features estratte da un campione di discorso e una fase di verifica successiva in cui dapprima l'individuo pone un reclamo di identità e successivamente fornisce un campione del suo discorso. Le features estratte in modo simile da quest'ultimo vengono confrontate con il modello per determinare se l'individuo stesso corrisponde o meno all'identità rivendicata. Al giorno d'oggi, l'ASV è diventata oramai una tecnologia piuttosto matura al punto da essere implementata in una crescente varietà di applicazioni pratiche e commerciali come nei call-centre, nei sistemi di dialogo parlato e in molti prodotti di largo consumo. Sfortunatamente però, come nel caso di qualsiasi tecnologia biometrica, tali sistemi si mostrano fortemente vulnerabili ad attacchi di spoofing. Un attacco di spoofing non è altro che un tentativo malevolo di mascheramento dell'utente target attuato sottoponendo al sistema di verifica falsi tratti biometrici provocando così, in caso di successo, un aumento del tasso dei falsi positivi (FAR). Come riportato in [2], nell'ambito dei sistemi ASV attualmente si hanno le seguenti tipologie di spoofing: imitazione, replay, sintesi vocale e voice conversion. Dal punto di vista delle contromisure adottate, nel corso degli ultimi anni la letteratura a riguardo è andata via via arricchendosi [3]. Nonostante questo però, la maggior parte dei lavori focalizza lo studio su un unica tipologia di spoofing e così facendo la conoscenza a priori dell'attacco rende le contromisure piuttosto specifiche non riflettendo uno scenario pratico nel quale si possono verificare attacchi di spoofing mai visti prima[1]. Inoltre l'utilizzo di un numero ridotto di attacchi o algoritmi di spoofing potrebbe non offrire il massimo potenziale per la generalizzazione di attacchi diversi o imprevisti che quasi certamente si incontreranno in natura [4]. Infine la mancanza di un dataset, protocolli e metriche comuni non rende semplice il confronto tra studi differenti. Considerando tutto ciò nel 2015 è stata indotta la prima challenge riguardante lo spoofing (ASVspoof) nella quale è prevista proprio l'implementazione di un sistema di anti-spoofing, definito sistema CM ed utilizzato separatamente dal sistema ASV. Fornendo un dataset consistente di speech spoof e bonafide,

[1] Ad esempio le contromisure per la voice conversion potrebbero non classificare come spoof speech generati da algoritmi di sintesi vocale.

l'obbiettivo finale è quello di fornire un quadro di valutazione comune.

Il progetto qui presentato rientra nell'ambito dell'ASVspoof2019 [5]. Tale edizione focalizza l'analisi sui tre principali attacchi di spoofing: TTS (text-to-speech), VC (voice conversion) e replay. Andando più nel dettaglio gli obbiettivi tecnici possono essere riassunti come segue:

- Rafforzamento delle contromisure per gli attacchi con TTS e VC constatando il fatto che negli ultimi anni gli stessi sistemi TTS e VC hanno subito un notevole progresso al punto che oggi, un linguaggio sintetico ben addestrato e una voce convertita sono tanto buoni quanto percettibilmente indistinguibili da uno speech reale
- Creazione di una configurazione più controllata per gli attacchi di replay, simulando opportune condizioni acustiche dell'ambiente e del microfono e definendo qualità predefinite del dispositivo di riproduzione
- Adozione di una metrica di valutazione per valutare gli impatti dei sistemi autonomi di anti-spoofing su un sistema ASV fissato

Inoltre, sulla base della tipologia degli attacchi, l'edizione prevede due challenge differenti: una denominata *Logical Access*(LA) che mira all'implementazione di contromisure per attacchi generati con TTS e VC e l'altra, definita *Physical Access*(PA), si riferisce unicamente agli attacchi di tipo replay.
Il sistema anti-spoofing qui proposto [6] consiste di una fase iniziale di *feature engineering* seguita da uno stadio di classificazione. Quest'ultimo è basato su una *Deep Neural Network* (DNN), in particolar modo è stato studiato e implementato il modello SEnet (*Squeeze and Excitation Network*) basato sull'architettura ResNet (*Residual Neural Network*).

Il lavoro presentato è strutturato come segue: nella Sezione 2 viene fornita una breve descrizione dei dataset delle due sub-challenge, mentre nella Sezione 3 viene introdotta un'analisi teorica e approfondita sul modello SEnet. L'implementazione del sistema è presentata nelle Sezioni 4, 5, 6 dove rispettivamente viene descritto lo stadio di *feature engineering*, il classificatore DNN e l'ottimizzazione dei parametri della rete. Infine nelle Sezioni 8, 9, 10 vengono illustrati i dettagli della sperimentazione, presentati i risultati ottenuti e le relative conclusioni.

2 Dataset

2.1 Logical Access (LA)

Il dataset per il *Logical Access* prevede attacchi di spoofing generati con la tecnologia di text-to-speech (TTS) e voice conversion (VC) basandosi su un dataset standard di sintesi vocale multi-speaker chiamato VCTK. Gli speech bonafide vengono raccolti da differenti speaker senza effetti significativi di rumore di fondo o canale e sempre da questi si ottengono speech spoof utilizzando una serie di diversi algoritmi di spoofing appartenenti a una delle due categorie. L'intero dataset è partizionato in tre parti: una per il training, una per il development e l'ultima per la fase di evaluation. In tutte e tre lo speech spoof è stato generato utilizzando in totale sei algoritmi di spoofing(due per la voice conversion e quattro per la sintesi vocale). La partizione di evaluation include

anche campioni di spoof ottenuti tramite algoritmi sconosciuti, ovvero varianti di quelli utilizzati nelle altre due partizioni; tutto questo con l'obbiettivo di studiare le prestazioni di generalizzazione dei modelli in un ipotetico scenario "in the wild".

2.2 Physical Access (PA)

Il dataset relavito alla sub-challenge *Physical Access* include attacchi di spoofing eseguiti a livello di sensore. Ciò implica che sia i segnali bonafide che quelli spoof si propaghino attraverso uno spazio fisico prima dell'acquisizione assumendo inoltre, che prima di essere riprodotta sul microfono ASV, la registrazione dello speech viene catturata da un tentativo di accesso bonafide. L'edizione 2019 si basa su configurazioni acustiche e replay simulate e attentamente controllate. L'attacco di spoofing è dato dalla sequenza di una registrazione di un tentativo di accesso bonafide e della riproduzione di quest'ultima sul microfono. Le registrazioni vengono catturate a distanza D_a dallo speaker, mentre si assume che la loro successiva presentazione al microfono sia effettuata dalla stessa distanza D_s dei tentativi di accesso bonafide. La configurazione acustica è la seguente: si considera una stanza di dimensione S con riverbero R e una distanza D_s tra lo speaker e il microfono. I tentativi di attacco tramite replay sono caratterizzati dalla propagazione nella stessa configurazione acustica in cui l'autore dell'attacco registra una copia del tentativo di accesso bonafide a distanza D_a dallo speaker. Le registrazioni vengono poi riprodotte utilizzando un altoparlante di qualità Q. I campioni bonafide derivano sempre dal dataset VCTK. Per simulare gli attacchi replay tali campioni vengono pre-processati al fine di ricreare la fase di cattura e riproduzione secondo lo scenario sopra esposto. Come nel caso del *Logical Access* il dataset finale è diviso in tre parti. Tutte e tre includono differenti configurazioni acustiche (parametri S, D_s, R) e di replay (parametri Q, D_a), inoltre anche in questo caso la partizione di evaluation include attacchi non conosciuti a priori, ovvero configurazioni acustiche e replay random.

3 Squeeze and Excitation Network

Una *Squeeze and Excitation Network* è una rete neurale ottenuta dall'integrazione di un'unità chiamata SE, *Squeeze and Excitation*, in un'architettura CNN (*Convolutional Neural Network*). Per quanto riguarda quest'ultima in tale progetto è stato utilizzato il modello ResNet.
Di seguito viene fornita un'analisi dettagliata delle reti CNN, ResNet e SEnet, descrivendo in primo luogo i motivi che hanno portato alla loro introduzione e successivamente i dettagli architetturali e implementativi di ciascuna di esse.

3.1 Convolutional Neural Networks

Le *Convolutional Neural Networks* (definite con l'acronimo CNN) sono reti neurali specializzate nel processamento di dati che presentano una struttura a griglia e utilizzano l'operazione matematica lineare di convoluzione. Quest'utima è definita come la somma degli elementi del prodotto di Hadamard fra un set di parametri (che prende il nome di filtro o kernel) e una porzione dell'input di pari dimensioni. L'operazione di convoluzione viene quindi ripetuta spostando il filtro lungo tutta la superficie dell'input, sia in altezza che in larghezza. Questo produce quella che viene chiamata *features map*.

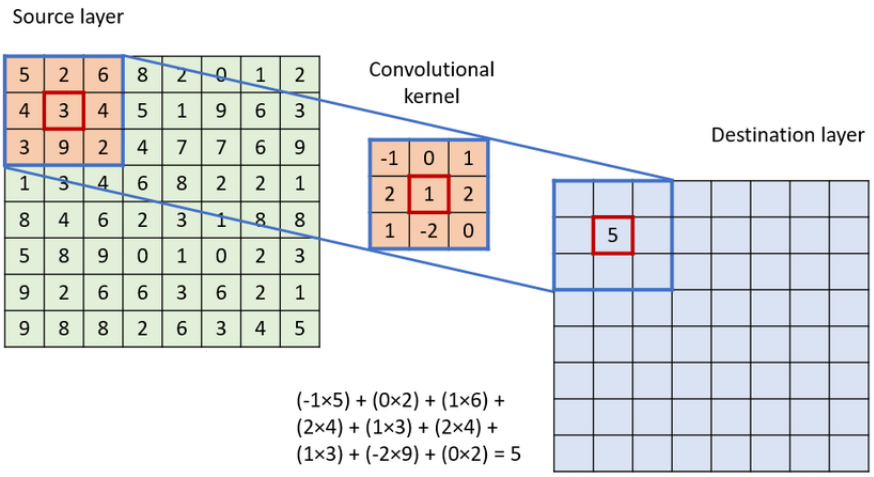


Figure 1: Operazione di convoluzione

Uno dei motivi che hanno portato all'introduzione delle *Convolutional Neural Networks* è l'inadeguatezza della struttura fully-connected. Le reti neurali tradizionali ricevono in input un singolo vettore, e lo trasformano attraverso una serie di strati nascosti. In tal modo ogni neurone è connesso ad ogni singolo neurone sia dello strato precedente che di quello successivo (ovvero "fully-connected ") e funziona quindi in maniera completamente indipendente, dal momento che non vi è alcuna condivisione delle connessioni con i nodi circostanti. Nel caso l'input sia costituito da immagini di dimensioni ridotte, ad esempio 32x32x3 (32 altezza, 32 larghezza, 3 canali colore), un singolo neurone connesso in questa maniera comporterebbe un numero totale di $32\times32\times3=3072$ pesi, una quantità abbastanza grande ma ancora trattabile. Le cose si complicano però quando le dimensioni si fanno importanti: salire ad appena 256 pixel per lato comporterebbe un carico di $256\times256\times3=196608$ pesi per singolo neurone, ovvero quasi 200.000 parametri per una semplice rete con un singolo strato nascosto da dieci neuroni. L'architettura fully-connected risulta perciò troppo onerosa in questo contesto ed inoltre un'architettura di questo tipo fatica a cogliere la struttura di correlazione tipica dei dati a griglia. Le *Convolutional Neural Networks* prendono invece vantaggio dall'assunzione che gli input hanno proprio una struttura di questo tipo. Il risultato è una rete più efficace e allo stesso tempo parsimoniosa in termini di parametri.

Gli strati composti da operazioni di convoluzione prendono il nome di convolutional layers, ma non sono gli unici strati che compongono questa tipologia di rete: la tipica architettura (Figura 2) prevede infatti l'alternarsi di convolutional layers, pooling layers e fully connected layers.

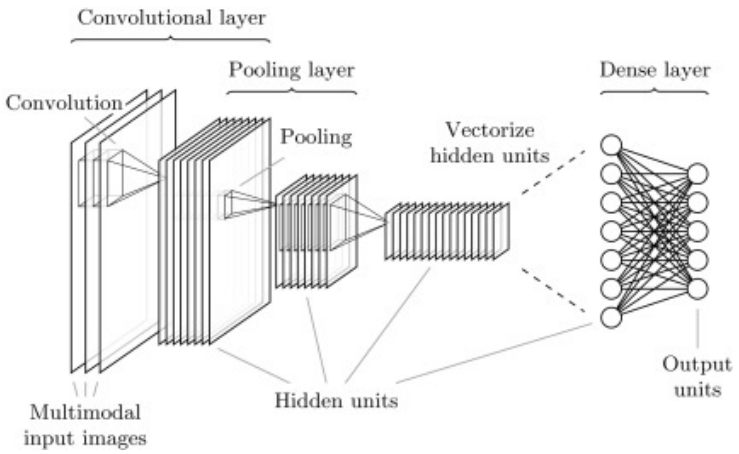


Figure 2: Architettura tipica di una rete CNN

- **Convolutional layer**: generalmente un convolutional layer è formato da un set di filtri piuttosto numeroso, indicato qui con N_F, tutti con la medesima estensione spaziale. Durante lo stage feed-forward ogni filtro viene fatto convolvere lungo la larghezza e l'altezza del volume di input, e pertanto vengono prodotte N_F *features map*, le quali forniscono ognuna la risposta del relativo filtro in ogni posizione spaziale. Disposizione finale e numero di neuroni che compongono il volume di ouput del convolutional layer sono controllati inoltre da tre iperparametri:
  - **Profondità**: corrisponde al numero di filtri N_F che compongono lo strato, ognuno in cerca di caratteristiche differenti nel volume di input.
  - **Stride**: specifica il numero di pixel di cui si vuole traslare il filtro ad ogni spostamento. Quando lo stride è pari ad uno significa che stiamo muovendo il filtro un pixel alla volta, e di conseguenza viene scannerizzata ogni possibile posizione dell'input. Valori più alti muovono il filtro con salti maggiori, e pertanto viene generato un output di dimensioni minori.
  - **Zero-padding**: a volte può risultare conveniente aggiungere un bordo di zeri al volume di input, in modo così da controllare le dimensioni dell'output ed evitare incongruenze durante le operazioni. Lo spessore di questo bordo è determinato dall'iperparametro di zero-padding, ed è spesso utilizzato per far combaciare la dimensione dell'input con quella dell'output.

  Per quanto riguarda le funzioni di attivazioni tipicamente i convolutional layers prevedono in cascata la funzione di attivazione *relu* così definita: tutti i pixel con un valore negativo vengono sostituiti da zero:

$$R(z) = max(0, z) \tag{1}$$

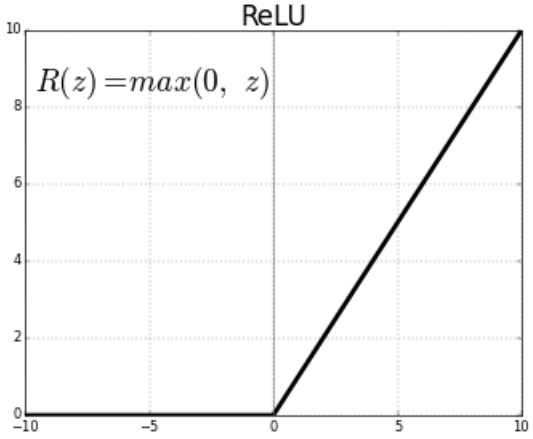


Figure 3: Relu

- **Pooling layer**: nell'architettura di una CNN è pratica comune inserire fra due o più convolutional layers uno strato di pooling, la cui funzione è quella di ridurre progressivamente la dimensione spaziale degli input (larghezza e altezza), in modo da diminuire numero di parametri e carico computazionale. Il pooling layer opera indipendentemente su ogni *feature map* applicando un filtro di dimensione opportuna che esegue una determinata operazione deterministica (tipicamente il massimo o la media) non comportando la presenza di pesi e quindi parametri aggiuntivi.

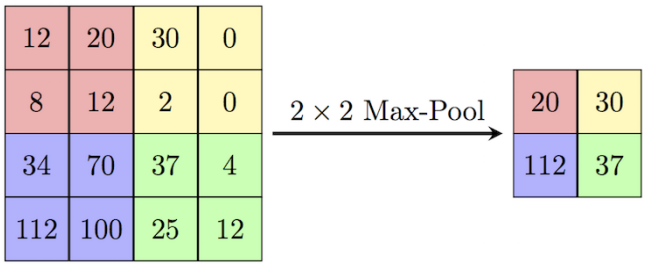


Figure 4: Max-Pooling Layer

- **Fully-connected layer**: il fully-connected layer è esattamente uguale ad un qualsiasi strato nascosto che compone le tradizionali reti neurali ed opera sul volume di output vettorizzato dello strato che lo precede. La funzione principale di tali strati, inseriti solo per ultimi a completare la struttura delle rete, è quella di eseguire un raggruppamento delle informazioni ottenute negli strati precedenti, esprimendole attraverso un numero (attivazione neuronale) che servirà nei successivi layer per la classificazione finale. Intuitivamente, l'idea è quella che la rete apprenda filtri che si attivino alla visione di determinati tipi di caratteristiche (features) come ad esempio angoli, linee o blocchi di colore nello strato iniziale (features di basso livello), oppure combinazioni via via sempre più complesse negli strati superiori (features di alto livello).

3.1.1 Problema della degradazione del gradiente

Nell'ultimo decennio il crescente bisogno di reti più profonde e più performanti ha fatto emergere il problema della degradazione del gradiente così definito: all'aumentare della

profondità della rete, l'accuratezza tende a saturare per poi degradarsi rapidamente comportando dunque un aumento dell'errore di training. Intuitivamente, si è portati a pensare che reti neurali più profonde non dovrebbero performare peggio di quelle poco profonde, o almeno non durante la fase di training. Considerando infatti una rete composta da n strati che raggiungono una determinata accuratezza, come minimo una rete con $n + 1$ strati dovrebbe essere in grado di raggiungere lo stesso grado di accuratezza, copiando i primi n strati ed eseguendo un identity mapping per l'ultimo. Allo stesso modo, reti di $n+2$, $n+3$ e $n+4$ strati possono, con lo stesso metodo, ottenere la medesima accuratezza. Nonostante ciò è stato dimostrato come questo non sia sempre vero e allo stato attuale la soluzione più accattivante per tale problema è rappresentata dalle *Residual Neural Networks*.

3.2 ResNet

Gli sviluppatori del modello ResNet hanno ricondotto il problema della degradazione del gradiente all'ipotesi che gli identity mapping siano difficili da apprendere proponendo l'introduzione del concetto di *residual learning* e unità architetturale definita residuale [7].

Si consideri un blocco di una generica rete neurale costituito da layer contigui di cui alcuni non lineari, si denoti con x l'ingresso di tale blocco e con $H(x)$ il mapping che si vuole ottenere dall'apprendimento. L'idea alla base dell'unità residuale è quella di far apprendere ai layer non la funzione $H(x)$ ma un nuova funzione $F(x)$ definita come $F(x) = H(x) - x$. In questo modo riscrivendo $H(x)$ come $H(x) = F(x) + x$ si osserva che i layer cercheranno di apprendere la differenza tra x e $H(x)$, aggiustando quest'ultima all'input. Poichè tale differenza è proprio $F(x)$ da qui il termine unità residuale. L'efficienza di tale metodo si afferma dimostrando che è più semplice ottimizzare l'apprendimento del residuale piuttosto che quello della funzione originale. Nel caso estremo infatti se $H(x)$ fosse un'identità ($H(x) = 1$) il blocco di layer raggiungerebbe più facilmente l'azzeramento del residuale invece che la condizione di mapping unitario. La realizzazione della formulazione $H(x) = F(x) + x$ viene realizzata tramite l'aggiunta di connesioni *shortcut* come mostrato in Figura 5.

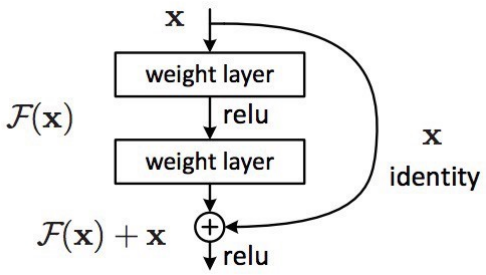


Figure 5: Blocco residuale

L'operazione di addizione viene effettuata elemento per elemento. Nel caso in cui l'input e il residuale siano di dimensioni differenti, vengono tipicamente utilizzate delle tecniche di zero-padding o si applica una proiezione lineare (ad esempio tramite una convoluzione 1×1) all'input per creare dimensioni corrispondenti. La composizione dell'unità residuale è flessibile potendo integrare quanti più layer si vogliono considerando però che residui con un solo layer non comportano vantaggi. Sempre in [7] sono state implementate e

studiate due tipologie dell'unità residuale: una base e l'altra denominata bottleneck. In Tabella 1 vengono mostrate le rispettive architetture:

	Baseline	**Bottleneck**
layer 1	Conv 3×3	Conv 1×1
layer 2	Conv 3×3	Conv 3×3
layer 3	-	Conv 1×1

Table 1: Unità residuale del tipo baseline e bottleneck

Architetture tipiche basate sul *residual learning* sono: ResNet18, ResNet34, ResNet50, ResNet101[2]. Le prime due prevedono l'integrazione di unità residuali base mentre le restanti utilizzano unità bottleneck.

3.3 SEnet

Le *Squeeze-and-Excitation Networks* [8] sono reti CNN che integrano un'unità architetturale denominata *Squeeze-and-Excitation*, più brevemente SE. In un layer convoluzionale di una tradizionale CNN la *features map* in uscita viene ottenuta pesando equamente ciascuno dei canali in ingresso, l'introduzione di un blocco SE invece, permette di ponderare in modo adattativo ciascun canale modellando esplicitamente le relazioni tra i canali stessi. Dal punto di vista implementativo il blocco SE (Figura 7) agisce su una feature map U, derivante dall'operazione di convoluzione F_{tr} applicata all'input X.

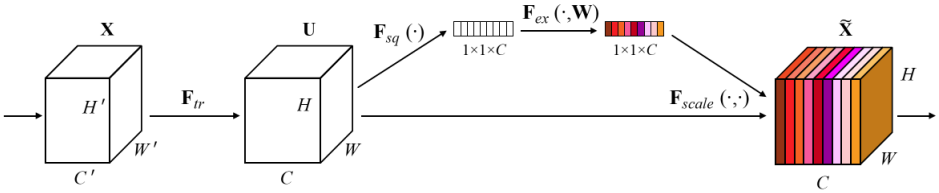


Figure 6: Integrazione del blocco *Squeeze-and-Excitation*

In primo luogo è prevista l'applicazione di un'operazione denominata squeeze che ha il compito di fornire una rappresentazione globale di ciascun canale descrivendo esso con un singolo valore numerico. Di conseguenza, considerando un set di C filtri l'operazione di squeeze ha come risultato un vettore unidimensionale di C elementi. Successivamente si applica un'operazione di excitation che ha come obbiettivo l'apprendimento delle interdipendenze tra i canali stessi. Quest'operazione è implementata tramite due layer fully-connected seguiti rispettivamente da una funzione di attivazione di tipo 'relu' e da una di tipo 'sigmoid' di modo che sia possibile apprendere una relazione non reciprocamente esclusiva e non lineare tra i canali. Moltiplicando a questo punto l'uscita del blocco di excitation, stadio finale dell'unità SE, con la feature map U d'ingresso si ottiene una pesatura ponderata e adattiva dei vari canali. La struttura del blocco SE è mostrata in Figura 7.

[2]La componente numerica nel nome sta ad indicare il numero totale di weighted layer nella rete.

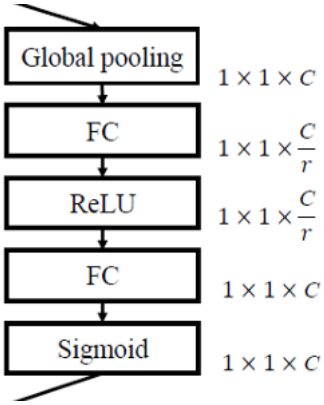


Figure 7: Architettura blocco *SE*

4 Feature engineering

L'estrazione delle *features* dai segnali audio costituisce lo stadio iniziale del sistema e si pone l'obbiettivo di ottenere determinate caratteristiche acustiche al fine di ridurre la ridondanza e fornire una rappresentazione più informativa. Nel presente lavoro è stata estratta un'unica *feature* acustica che risulta essere il *logspec (log power magnitude spectra)*. Lo spettrogramma di ciascun audio è stato calcolato suddividendo quest'ultimo in frame temporali overlappati del 50% e utilizzando una finestra di hamming. Sulla base della frequenza di campionamento pari a $16kHz$, la lunghezza dei frame è stata fissata a 512 campioni, ovvero $32ms$, rientrando così nell'intervallo di stazionarietà dello speech, inoltre, onde evitare una perdita di informazione, la STFT è stata calcolata utilizzando una FFT di lunghezza pari a 512 campioni. Infine, lo spettrogramma convertito in dB rappresenta la *feature map*. Considerando le dimensioni di quest'ultima, definite dalla tupla (h,w), ne scaturisce che il parametro h (direttamente correlato alla FFT) è fisso e pari a 257, mentre w varia al variare dell'audio dipendendo infatti dalla lunghezza temporale di quest'ultimo. A questo punto, a causa della disomogeneità di w e della necessità di fornire alla rete neurale input della medesima dimensione, per ciascuna *feature map* è stata definita una *unified feature map*. Riprendendo il metodo descritto in [9] quest'ultima viene ottenuta replicando la *feature map* lungo la dimensione w (ovvero lungo l'asse temporale). Volendo fornire un approccio più generale possibile la lunghezza delle *unified feature map* è stata fissata a valori multipli di 400. Andando più nel dettaglio: considerando una generica *feature map* di lunghezza w la *unified feature map* ottenuta presenta una lunghezza pari $M * 400$ dove M è l'intero superiore più vicino al quoziente $\frac{w}{400}$[3]. Una volta calcolata la *unified feature map*, essa viene suddivisa in segmenti di lunghezza pari a 400 ed overlappati del 50%. I segmenti così ottenuti di dimensioni 257×400 costituiscono gli ingressi della rete neurale. A tal proposito risulta evidente che per alcuni audio saranno presenti più segmenti; questo fatto viene risolto andando a mediare nella fase di classificazione gli output della rete neurale relativi a segmenti appartenenti allo stesso audio. In Figura 8 viene riportato l'intero processo di *feature engineering* e la modalità di classificazione quando l'audio presenta *feature map* multiple.

[3] Ad esempio da una feature map (257×625) si ottiene una *unified feature map* 257×800.

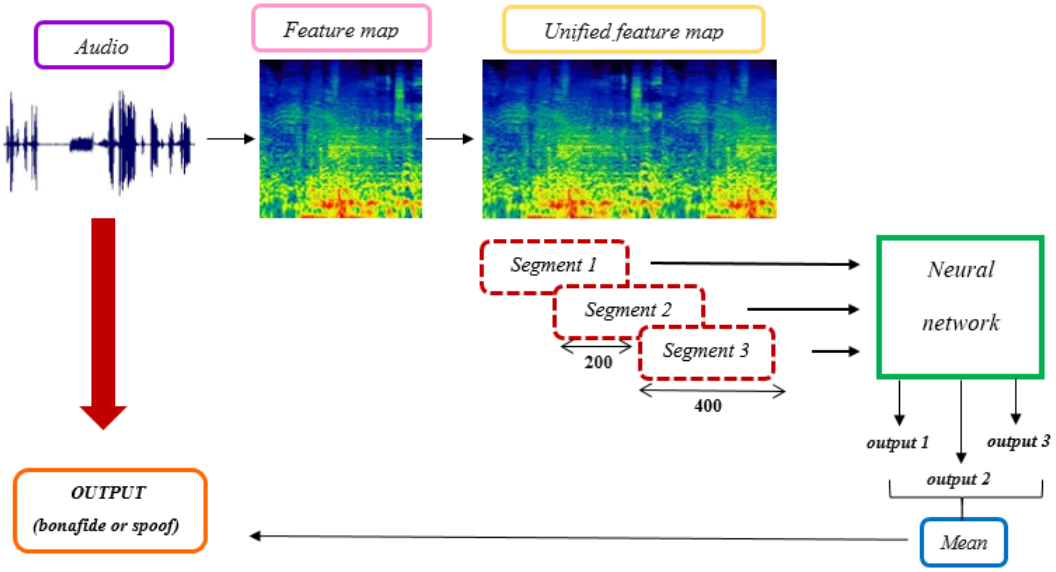


Figure 8: Processo di feature engineering

5 Classificatore

Il sistema anti-spoofing implementato ha l'obbiettivo di classificare un generico speech in ingresso in bonafide o spoof, per cui viene applicata una classificazione binaria. Come definito in precedenza il classificatore proposto è una *Squeeze and Excitation Network*(SEnet) del cui modello sono state implementate due varianti: SEnet34 e SEnet50. Entrambe basate sull'architettura ResNet, prevedono l'integrazione del modulo SE applicato all'uscita del residuo e anteposto alla somma con il ramo identità. Per quanto riguarda l'unità residuale, SEnet34 prevede un'unità base mentre SEnet50 utilizza il tipo bottleneck. In Tabella 2 vengono riportate entrambe le architetture:

Output size	**SEnet34**	**SEnet50**
129×200	conv 7×7, 16 channel, stride 2	
65×100	max-pooling 3×3, stride 2	
65×100	$\left\{\begin{array}{l}\text{conv } 3 \times 3,\ 16,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 16,\ \text{stride } 1\\ \text{fc } 16\end{array}\right\} \times 3$	$\left\{\begin{array}{l}\text{conv } 1 \times 1,\ 16,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 16,\ \text{stride } 1\\ \text{conv } 1 \times 1,\ 32,\ \text{stride } 1\\ \text{fc } 32\end{array}\right\} \times 3$
33×50	$\left\{\begin{array}{l}\text{conv } 3 \times 3,\ 32,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 32,\ \text{stride } 1\\ \text{fc } 32\end{array}\right\} \times 4$	$\left\{\begin{array}{l}\text{conv } 1 \times 1,\ 32,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 32,\ \text{stride } 1\\ \text{conv } 1 \times 1,\ 64,\ \text{stride } 1\\ \text{fc } 64\end{array}\right\} \times 4$
17×25	$\left\{\begin{array}{l}\text{conv } 3 \times 3,\ 64,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 64,\ \text{stride } 1\\ \text{fc } 64\end{array}\right\} \times 6$	$\left\{\begin{array}{l}\text{conv } 1 \times 1,\ 64,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 64,\ \text{stride } 1\\ \text{conv } 1 \times 1,\ 128,\ \text{stride } 1\\ \text{fc } 128\end{array}\right\} \times 6$
9×13	$\left\{\begin{array}{l}\text{conv } 3 \times 3,\ 128,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 128,\ \text{stride } 1\\ \text{fc } 128\end{array}\right\} \times 3$	$\left\{\begin{array}{l}\text{conv } 1 \times 1,\ 128,\ \text{stride } 1\\ \text{conv } 3 \times 3,\ 128,\ \text{stride } 1\\ \text{conv } 1 \times 1,\ 256,\ \text{stride } 1\\ \text{fc } 256\end{array}\right\} \times 3$
1×1	average-pooling	
1×1	fc 1, sigmoid	

Table 2: Architettura delle reti SEnet34 e SEnet50

Come si osserva le reti sono costituite da macroblocchi ciascuno dei quali include un numero fissato di unità residuali identiche. Le unità appartenenti a macroblocchi differenti si differenziano per il numero di filtri applicati nei layer convoluzionali, e per le dimensioni delle *feature map* d'ingresso e d'uscita: il numero dei filtri aumenta all'aumentare della profondità della rete mentre le feature map vengono progressivamente dimezzate, mantenendo così costante la complessità per blocco. Nel passaggio da un macro-blocco all'altro il dimezzamento delle dimensioni dell'input viene effettuato applicando una convoluzione 1×1 con uno stride pari a 2. Seguendo sempre l'implementazione proposta in [7] è stata utilizzata come tecnica di regolarizzazione la Batch Normalization [10] posta in seguito ad ogni layer convoluzionale e prima della funzione di attivazione.
Per quanto riguarda l'output della rete, avendo un classificatore binario, è stato inserito un solo nodo di uscita. Etichettato lo spoof con una label numerica fissata a 1 e lo speech bonafide con 0, una generico output del classificatore, rientra nel range $[0, 1]$ e rappresenta la probabilità che lo speech in ingresso sia spoof [4].

6 Ottimizzazione

Per quanto riguarda l'ottimizzazione di entrambe le reti la scelta è ricaduta sull'utilizzo dell'algoritmo Adam. Come descritto in [11] l'Adam, il cui nome sta per *Adaptive Moment Estimation*, è un metodo di ottimizzazione della discesa del gradiente in grado di implementare il calcolo adattivo del learning rate per ogni parametro richiedendo solo gradienti del primo ordine e stime dei momenti di primo e secondo ordine dei gradienti. L'algoritmo fa uso delle medie mobili esponenziali m_t e v_t rispettivamente del gradiente

[4] Da ciò la probabilità che lo speech in ingresso sia bonafide è data da $1 - output$.

(g_t) e del gradiente quadrato (g_t^2) calcolate nel seguente modo:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2}$$
$$v_t = \beta_1 v_{t-1} + (1 - \beta_1) g_t^2 \tag{3}$$

Dove gli iper-parametri β_1,β_2 determinano il decadimento esponenziale delle medie stesse. Tali medie mobili sono esse stesse rispettivamente stime del momento del primo e secondo ordine del gradiente. Essendo inizializzati a zero, m_t e v_t risultano distorti, in particolar modo nelle fasi iniziali dell'algoritmo, e soprattutto quando i tassi di decadimento sono bassi, perciò si considerano le seguenti correzioni:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{5}$$

Infine l'aggiornamento dei parametri è determinato nel seguente modo:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} - \epsilon} \tag{6}$$

Nel presente progetto sono stati fissati i seguenti valori:

$$\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$$

Per quanto riguarda il learning rate invece è stato utilizzato un valore variabile riprendendo l'approccio proposto in [12]:

$$\alpha = d_{model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5}) \tag{7}$$

Impostando il valore *warmup_steps* a 1000 e considerando uno step pari ad una batch ($step\_num = batch\_num$) ne scaturisce che l'equazione (7) si traduce in un learning rate che cresce linearmente per le prime 1000 batch e successivamente decresce in modo proporzionale all'inverso della radice quadrata del numero delle batch ($\alpha \propto \frac{1}{\sqrt{batch\_num}}$). Il parametro d_{model} indica la dimensionalità I/O del modello, perciò è previsto il valore 128 per la rete SEnet34 e 256 per SEnet50.

7 Metriche di valutazione

Come riportato in precedenza uno degli obbiettivi principali dell'ASVspoof è l'introduzione di protocolli e metriche comuni, al fine di confrontare le prestazioni di studi differenti. Nell'edizione 2019 sono state utilizzate: EER *(equal error rate)* e la recente *tandem detection cost function* (t-DCF). Entrambe si basano sui detection scores, punteggi che per ogni audio riflettono il supporto ad una delle due ipotesi (bona fide o spoof).
Essendo il classificatore di tipo binario, si ha una classe positiva e una negativa. In questo caso la prima corrisponde ai segmenti di speech che devono essere accettati(bona fide) mentre la seconda è relativa a quei segmenti che devono essere respinti dal sistema (spoof). Un detection score elevato sta a significare che l'audio in questione con alta probabilità appartiene alla classe positiva al contrario, un punteggio piuttosto basso riflette la quasi certa presenza di spoofing.

7.1 Tandem detection cost function (t-DCF)

La prima metrica utilizzata è la *minimum tandem detection cost function* definita in (11). La t-DCF si basa sulla teoria del rilevamento statistico e comporta specifiche dettagliate riguardanti un'applicazione prevista. La sua introduzione nasce dal fatto che i miglioramenti nella tecnologia CM non si traducono necessariamente in un sistema completo migliorato che prevede la cooperazione tra CM e ASV. Considerando infatti che entrambi i sottosistemi comporteranno errori di classificazione, l'obiettivo è valutare le prestazioni del sistema nel suo insieme tenendo conto non solo degli errori di classificazione di entrambi i sottosistemi, ma anche della probabilità a priori del tipo di utente e delle perdite subite nel caso uno dei due sottosistemi commetta un errore. La forma base della t-DCF adottata è la seguente:

$$\text{t-DCF}(s) = C_1 P_{miss}^{cm} + C_2 P_{fa}^{cm} \tag{8}$$

Dove P_{miss}^{cm} e P_{fa}^{cm} sono rispettivamente il tasso dei falsi negativi e quello dei falsi positivi. Le costanti C_1 e C_2 dipendono dai costi e dai detection scores di entrambi i sistemi (CM e ASV):

$$\begin{cases} C_1 = \pi_{tar}(C_{miss}^{cm} - C_{miss}^{asv} P_{miss}^{asv}) - \pi_{non}(C_{fa}^{asv} P_{fa}^{asv}) \\ C_2 = \pi_{spoof} C_{fa}^{cm}(1 - P_{miss,spoof}^{asv}) \end{cases} \tag{9}$$

I parametri C_{miss}^{asv} e C_{fa}^{asv} sono rispettivamente: i costi del sistema ASV nel mancare uno speaker target e nell'accettarne uno non target. In modo analogo per il sistema CM vengono assegnati due costi C_{miss}^{cm} e C_{fa}^{cm} rispettivamente per lo scarto di un speech bonafide e per l'accettazione di uno spoof. Vengono inoltre inserite delle probabilità a priori π_{tar}, π_{non}, π_{spoof} i cui valori si basano su applicazioni tipiche di autenticazione utente, riflettendo la presenza di speaker target, non target o spoof in uno scenario tipico. Si osserva inoltre che la metrica risulta essere una media ponderata dei tassi d'errore; i pesi di tale somma però non sono noti a priori ma dipendono anche dai tassi d'errore del sistema ASV P_{miss}^{asv}, P_{fa}^{asv}, $P_{miss,spoof}^{asv}$.

	Priors			**ASV costs**		**CM costs**	
Parameter	π_{tar}	π_{non}	π_{spoof}	C_{miss}^{asv}	C_{fa}^{asv}	C_{miss}^{cm}	C_{fa}^{cm}
LA	0.9405	0.0095	0.05	1	10	1	10
PA	0.9405	0.0095	0.05	1	10	1	10

Table 3: Parametri della metrica *t-DCF*

La (8) viene normalizzata rispetto a un sistema anti-spoofing che accetta o rigetta ciascun input:

$$\text{t-DCF}_{norm}(s) = \frac{\text{t-DCF}(s)}{\text{t-DCF}_{default}(s)} = \frac{\text{t-DCF}(s)}{min(C_1, C_2)} \tag{10}$$

Infine la *minimum tandem detection cost function* viene calcolata come segue:

$$\text{t-DCF}_{norm}^{min}(s) = \text{t-DCF}_{norm}(s*) \tag{11}$$

Dove $s* = argmin_s \text{t-DCF}_{norm}(s)$.

7.2 Equal error rate (EER)

La seconda metrica utilizzata è l'EER il quale corrisponde alla soglia del sistema CM s_{EER} alla quale il tasso dei falsi negativi e quello dei falsi positivi si uguagliano, vale a

dire $EER = P_{fa}^{cm}(s_{EER}) = P_{miss}^{cm}(s_{EER})$. Poiché i due rate variano in step discreti come valore di soglia s si utilizza quel valore s_{EER} che minimizza la differenza:

$$|P_{fa}^{cm}(s) - P_{miss}^{cm}(s)| \tag{12}$$

Andando poi a mediare i due valori ottenuti, ovvero:

$$EER = \frac{P_{fa}^{cm}(s_{EER}) + P_{miss}^{cm}(s_{EER})}{2} \tag{13}$$

con s_{EER} tale per cui $|P_{fa}^{cm}(s_{EER}) - P_{miss}^{cm}(s_{EER})| = min|P_{fa}^{cm}(s) = P_{miss}^{cm}(s)|$

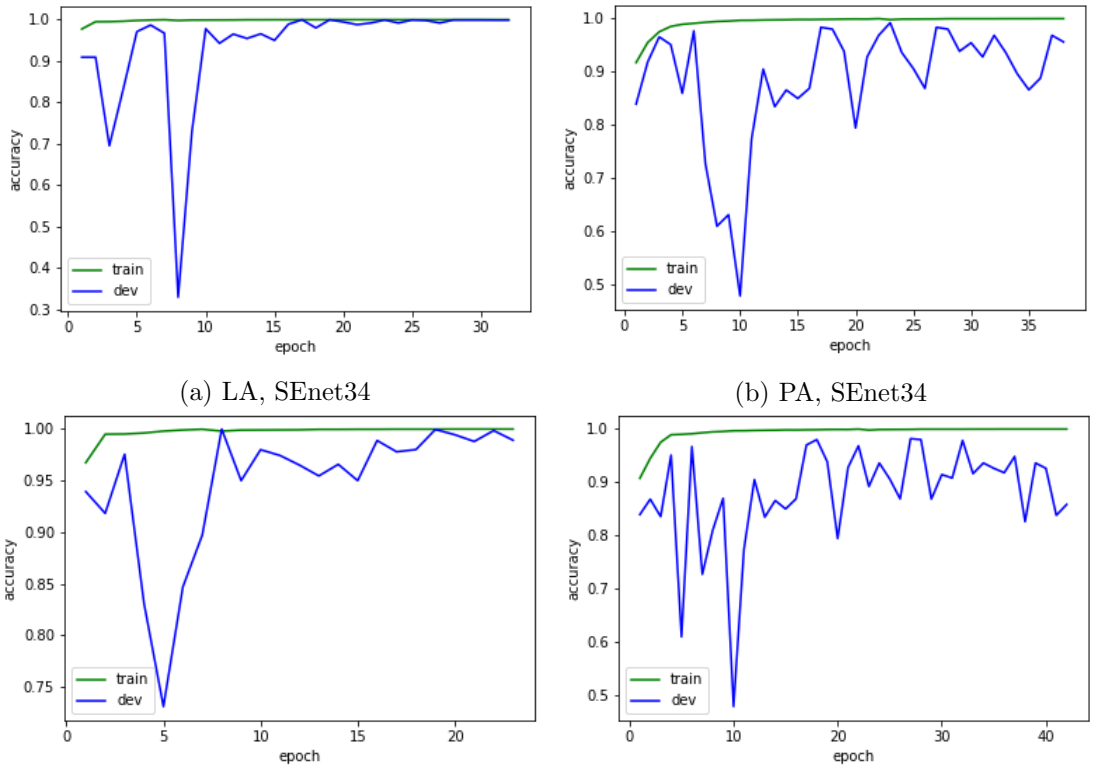
8 Sperimentazione

Nella fase di sperimentazione, sia per LA che per PA è stata utilizzata la partizione originaria del dataset in training, development, ed evaluation (Tabella 4).

	Logical Access (LA)		**Physical Access (PA)**	
	bonafide	spoof	bonafide	spoof
Training	2580	22800	5400	48600
Development	2548	22296	5400	24300
Evaluation	7355	63882	18090	116640

Table 4: Partizione dataset LA e PA

Il training di entrambe le reti (SEnet34, SEnet50) è stato effettuato considerando mini-batch contenenti 64 *feature map*, la binary cross-entropy come loss e come descritto precedentemente (Sezione 6) è stato implementato un learning variabile utilizzando l'algoritmo Adam come ottimizzatore. Al termine di ogni epoca è stata calcolata l'accuracy sul development-set, selezionando al termine della fase di training il modello con il valore migliore di *dev-accuracy*. Per il monitoraggio di quest'ultima è stato inserito un meccanismo di Early-Stopping con una patience pari a 15 epoche. In Figura 9 viene riportato l'andamento della *train-accuracy* e *dev-accuracy* di entrambi i dataset e le reti.



(a) LA, SEnet34

(b) PA, SEnet34

(c) LA, SEnet50

(d) PA, SEnet50

Figure 9: Andamento train-accuracy e dev-accuracy

Nella fase di valutazione, il calcolo delle metriche, sia sul development che evaluation set, ha richiesto la classificazione di ciascun campione delle due partizioni. Come riportato nella Sezione 4 gli output del classificatore relativi a *feature map* appartenenti allo stesso speech vengono mediati facendo in questo modo corrispondere un unico valore di classificazione ad ogni audio. Infine come detection score è stato utilizzato il logaritmo della probabilità della classe positiva (bonafide) calcolato come segue:

$$detection\ score = \log(1 - output_{DDN}) \tag{14}$$

9 Risultati

In Tabella 5 e 6 vengono riportate i valori di accuracy di entrambe le reti per entrambi i dataset:

	SEnet34	**SEnet50**
Development	99.93%	100%
Evaluation	75.19%	75.98%

Table 5: Accuracy *Logical Access*(LA)

	SEnet34	**SEnet50**
Development	99.22%	98.19%
Evaluation	98.91%	96.87%

Table 6: Accuracy *Physical Access*(PA)

Mentre in Tabella 7 e 8 vengono mostrati i valori delle metriche di valutazione:

	SEnet34		**SEnet50**	
	t-DCF$_{norm}^{min}$	**EER**(%)	**t-DCF**$_{norm}^{min}$	**EER**(%)
Development	0.002	0.079	0	0
Evaluation	0.161	7.927	0.156	8.812

Table 7: Risultati dataset *Logical Access*(LA)

	SEnet34		**SEnet50**	
	t-DCF$_{norm}^{min}$	**EER**(%)	**t-DCF**$_{norm}^{min}$	**EER**(%)
Development	0.036	1.538	0.040	1,518
Evaluation	0.053	1.979	0.063	2.398

Table 8: Risultati dataset *Physical Access*(PA)

La prima osservazione da fare è che entrambe le reti sul dataset LA ottengono ottimi risultati per la partizione development ma faticano a generalizzare quando si presentano attacchi di spoofing sconosciuti, difatti si ha un EER piuttosto elevato e un'accuracy intorno al 75%. Nel caso PA invece la generalizzazione è notevole tanto che i risultati tra development ed evaluation si discostano relativamente poco. Confrontando le due reti invece risulta evidente che la SEnet34 è più performante sull'evaluation mentre sul development entrambe raggiungono le stesse prestazioni.

10 Conclusioni

E' stato introdotto e presentano un sistema anti-spoofing costituito da uno stadio di *feature engineering*, che ha previsto l'estrazione della *feature* acustica *logspec* per ogni campione d'ingresso, e un successivo stadio di classificazione basato su una *Squeeze and Excitation Network*. I risultati ottenuti mostrano come l'integrazione del deep learning nei sistemi di contromisure per attacchi di spoofing sia un approccio valido soprattutto nel campo negli attacchi di tipo replay. Ulteriori miglioramenti risultano invece necessari nell'ambito degli attacchi dovuti alla sintesi vocale e alla voice conversion.

References

[1] B. Hao and X. Hei, "Voice Liveness Detection for Medical Devices," *Design and Implementation of Healthcare Biometric Systems*, pp. 109–136, 2019.

[2] N. Evans, T. Kinnunen, and J. Yamagishi, "Spoofing and countermeasures for automatic speaker verification," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 01 2013.

[3] Z.Wu, N.Evans, T.Kinnunen, J.Yamagishi, F.Alegre, and H.Li, "Spoofing and countermeasures for speaker verification: a survey," *Speech Communication*, vol. 66, p. 130–153, 2015.

[4] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilci, M. Sahidullah, and A. Sizov, "ASVspoof 2015: the First Automatic Speaker Verification Spoofing and Countermeasures Challenge," in *Interspeech*, 2015.

[5] "ASVspoof 2019: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan." `https://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf`.

[6] C.-I. Lai, N. Chen, J. Villalba, and N. Dehak, "ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual neTworks," *arXiv preprint arXiv:1904.01120*, 2019.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[9] C.-I. Lai, A. Abad, K. Richmond, J. Yamagishi, N. Dehak, and S. King, "Attentive Filtering Networks for Audio Replay Attack Detection," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019.

[10] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, p. 448–456, 2015.

[11] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017.