UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Elettronica

DIGITAL ADAPTIVE CIRCUITS AND LEARNING SYSTEMS

Analisi delle prestazioni della rete ibrida PARCnet per il packet loss concealment su segnali vocali

Michele Marmorè

Anno Accademico 2023/2024

Indice

1 Introduzione

Le comunicazioni audio in tempo reale su IP sono essenziali nell'uso quotidiano, tuttavia sono affette intrinsecamente da problemi dovuti alla trasmissione. Un segnale per la comunicazione in tempo reale è trasmesso con reti di pacchetti. Per ridurre al minimo la latenza e garantire un utilizzo fluido ed ininterrotto le reti di comunicazione a pacchetti si basano su protocolli RTP/UDP anche detti "best-effort". Quest'ultime sono concepite per garantire una massima velocità di trasmissione a discapito della qualità. A causa di ciò, le reti best-effort sono soggette a perdite di dati: in ricezione un decoder legge iterativamente un buffer jitter che contiene l'informazione dell'ultima trasmissione. In caso di perdita di un pacchetto il suddetto buffer risulta essere vuoto. Al verificarsi di questa anomalia, le precedenti tecniche di compensazione, sostituivano il segnale mancante con silenzio, rumore o frammenti di segnale ricevuto precedentemente [2]. Oltre alla semplice ripetizione dell'ultimo pacchetto ricevuto [3], quest'ultimo tipo di PLC può sfruttare le informazioni sul tono e la correlazione di segmenti di forma d'onda validi e sostituiti per rendere l'inserimento dei dati fluido riducendo al minimo le discontinuità [4]. Poiché i segnali vocali sono in genere "ben comportanti"[1] vale a dire che possono essere considerati stazionari entro finestre di breve durata, i modelli autoregressivi lineari (AR) si sono dimostrati efficaci e relativamente poco costosi per applicazioni vocali [5], [6], [7] e musicali in rete [8]. I metodi PLC più recenti prevedono l'uso di reti neurali per prevedere pacchetti futuri dal contesto audio ricevuto in precedenza. In questo elaborato si propone l'uso della rete PARCnet come metodo PLC per segnali speech.

2 PARCnet

PARCnet è un metodo PLC ibrido che utlizza una rete neurale feed-forward per stimare il segnale residuo nel dominio del tempo di un modello auto-regressivo lineare parallelo. La rete è formata da due rami paralleli come illustrato in figura 1. Un ramo è composto da un solo blocco linear predictor: il linear predictor è un modello che predice un valore futuro in una sequenza di dati basandosi su una combinazione lineare di valori passati. In un predittore lineare il valore stimato $\hat{x}$[n] è calcolato come somma pesata di "**k**" valori precedenti come mostrato in (1):

$$\hat{x}[\mathrm{n}] = \sum_{i=1}^{k} a_i \cdot x[n-i] \tag{1}$$

Per come definito il ramo che contiene il linear predictor è molto importante nell'architettura PARCnet perché permette una previsione veloce ed accurata a breve termine. Grazie alla presenza del linear predictor si va a compensare la predizione del ramo a strati convolutivi che invece coglie la dipendenza temporale a lungo termine del segnale. Come detto in precedenza PARCnet è una rete ibrida formata da due rami paralleli. Ora descriviamo il secondo ramo. Il modello implementato è a bottleneck interamente convoluzionale [10].

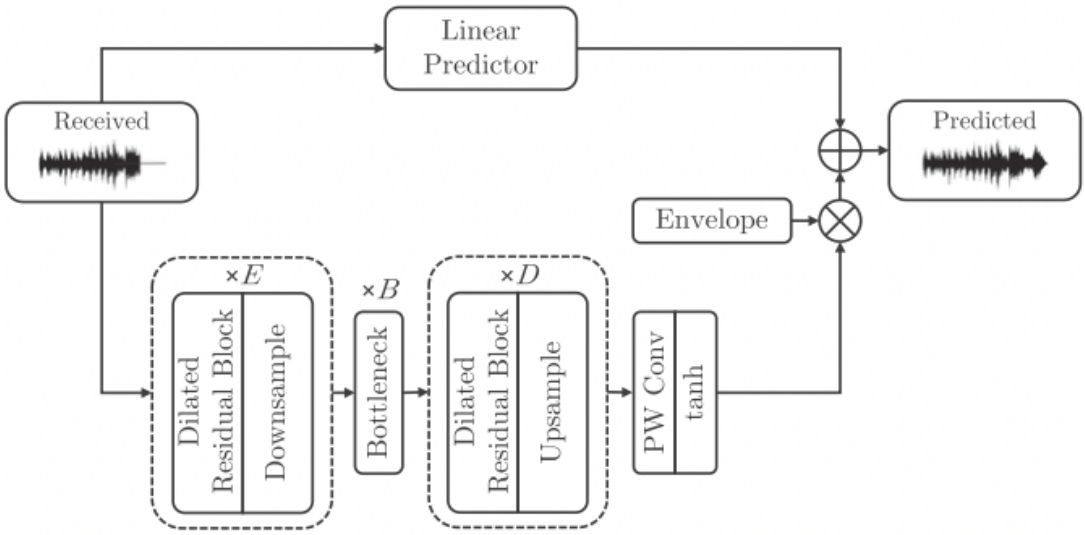

Fig.1 Struttura PARCnet

In Figura 1 è illustrata la struttura del modello PARCNnet. I blocchi "E" e "D" rappresentano rispettivamente il codificatore ed il decodificatore. Ognuno comprende blocchi di dilatazione residua, con in caduta rispettivamente un blocco downsample basato su max-pooling ed un blocco upsample [11],entrambi di ordine 2.

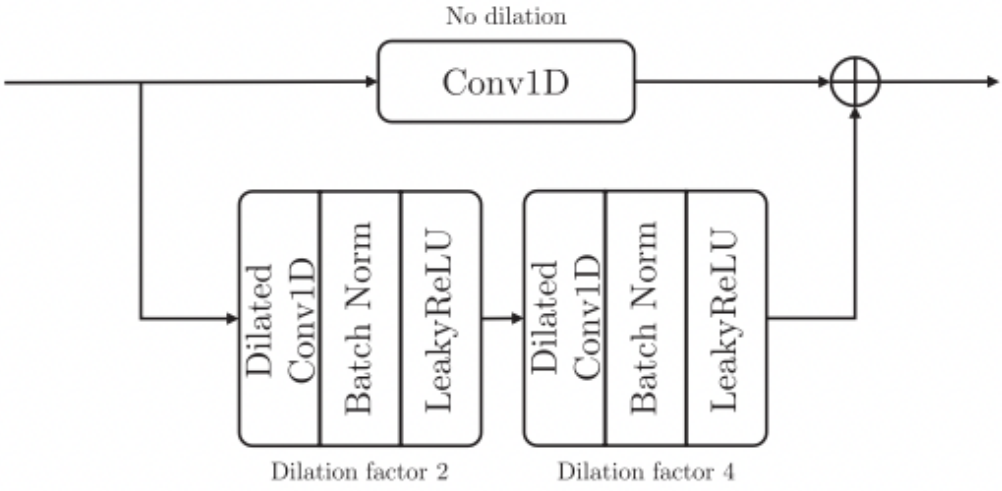

Fig.2 Dilated Residual Block

Ogni blocco residuo dilatato, illustrato in figura 2, comprende una connessione che passa attraverso uno strato convoluzionale senza dilatazione e due pile di convoluzioni con fattore di dilatazione rispettivamente di due e quattro, normalizzazione batch e LeakyReLU con pendenza $\alpha = 0{,}2$ [1]. Il numero di filtri cresce progressivamente nell'encoder (8, 16, 32, 64) e diminuisce simmetricamente nel decoder (64, 32, 16, 8). Gli strati convoluzionali hanno filtri di dimensione 11 nell'encoder e sette nel decoder [1].
In figura 3 è illustrato il blocco che implementa il bottleneck.

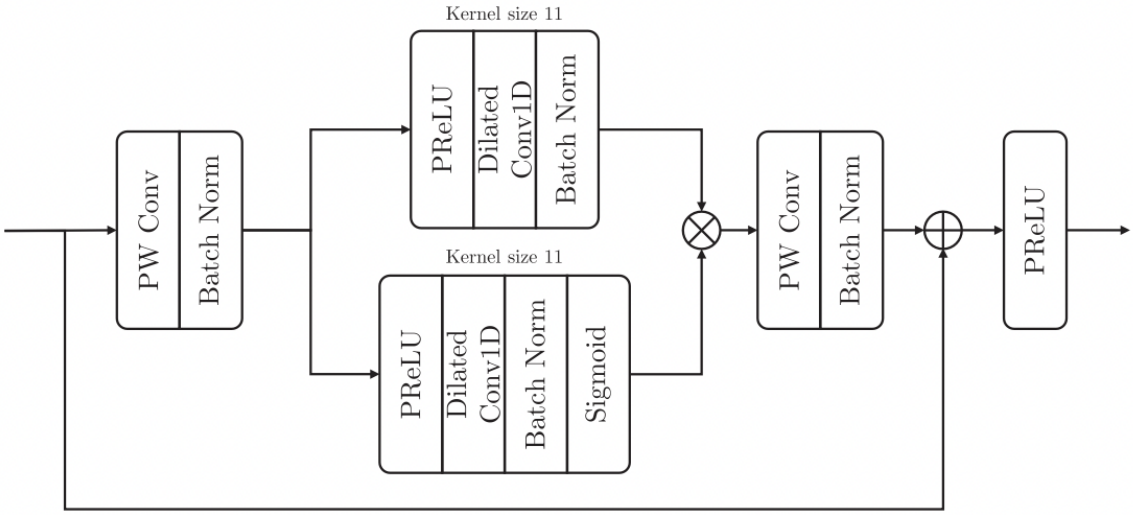

Fig.3 bottleneck, blocco GLU

Il bottleneck B, invece, comprende 6 blocchi costituiti da uno stack di input, una gated linear unit (GLU) [12] con dimensione kernel 11 e uno stack di output, nonché un percorso residuo che abbrevia l'input e l'output del blocco, seguito da PReLU [13]. Gli stack di input e output presentano una convoluzione puntuale (PW) con fattore di dilatazione di 1 e 32 e 64 canali, rispettivamente, seguita da normalizzazione batch [1]. Similmente a [10], il tasso di dilatazione cresce esponenzialmente con ogni GLU al fine di catturare la correlazione tra campioni sempre più distanti. Vale a dire, lo strato

j-esimo del collo di bottiglia ha una velocità di dilatazione di 2^{j-1} , j=1,....,B [1].

L'output del decoder viene quindi immesso in una convoluzione PW seguita da una funzione di attivazione della tangente iperbolica. Tutte le convoluzioni nel modello sono 1D [1].

3 Procedura di Train e Test

3.1 Dataset

Gli esperimenti sono stati eseguiti utilizzando al completo il dataset "MS PLCchallenge2022" che contiene circa 64 ore di file audio speech. Questo è stato creato e fornito da microsoft per la sfida "INTERSPEECH 2022", [16] ed è costituito originariamente da un set di file audio e metadati suddivisi in tre gruppi:

- train-set
- validation-set
- blind-set

Le porzioni train-set e validation-set contengono per ogni file audio incluso:

- clean-signals
- lossy-signals
- un file di testo con metadati di perdita

I metadati indicano, con una riga per ogni segmento di audio di 20 millisecondi, se il pacchetto è stato perso (la riga contiene un "1") o meno (la riga contiene uno "0"). I file audio "lossy" hanno i segmenti corrispondenti azzerati. La suddivisione di training include un totale di 23184 clip, mentre la suddivisione di validazione e test cieco include ciascuna 966 clip [16]. Il set di dati è stato costruito tramite un campionamento stratificato di tracce di perdita di pacchetti effettive osservate nelle chiamate effettuate dagli utenti di Microsoft Teams, applicate a segmenti di audio scelti casualmente da un set di dati di podcast. Le tracce sono state campionate come segue:

in primo luogo, segmenti di 10 secondi con almeno un pacchetto perso sono stati estratti casualmente dalle tracce di perdita di pacchetti dalle chiamate di Teams [16]. Questi segmenti sono stati quindi divisi in tre sottoinsiemi in base alla lunghezza massima della perdita di burst nella traccia:

- fino a 120 millisecondi (504blindsetclip)
- tra 120 e 320 millisecondi (308blindsetclip)
- tra 320 e 1000 millisecondi (154blindsetclip)

Le tracce con perdite di burst superiori a 1000 millisecondi sono state scartate. Ogni sottoinsieme è stato quindi suddiviso in 14 celle in base ai quantili di perdita percentuale dei pacchetti. Infine, è stata campionata una quantità uguale di tracce da ogni cella (con più tracce campionate per i sottoinsiemi con perdite di burst massime più brevi) [16]. Per creare l'audio con perdite di pacchetti, sono state campionate clip audio di circa 10 secondi di lunghezza da un dataset di podcast di pubblico dominio (LibriVox Community Podcasts), tagliando le regioni a basso volume per non dividere le parole. Le perdite dalle tracce campionate sono state quindi applicate ai file audio risultanti azzerando le regioni corrispondenti nelle clip audio [16].

3.2 Train

Per la procedura di train, i segnali da elaborare sono stati caricati ricampionandoli ad una SR di 32kHz, idonea a contenere tutte le caratteristiche speech. Non viene effettuata alcuna normalizzazione in fase di caricamento in modo da rendere il sistema robusto in temini di silenzio e grandi variazioni di ampiezza del segnale [1]. Rispetto a quanto presente nel dataset, per il train è stata utilizzata solo la porzione designata che comprende 23184 file, dei

quali una piccola percentuale è stata riservata per la validation(10 tracce). L'addestramento è stato completato con 500 epoche ognuna caratterizzara da 500 step, batch size di 128 e fade dimension di 64. Le funzioni di loss utilizzate nella fase di training sono le seguenti:

- **SpectralConvergenceLoss:** viene utilizzata per confrontare lo spettro di ampiezza tra un segnale predetto $\hat{Y}$ e un segnale di riferimento Y. Questa perdita misura la similarità spettrale tra i due segnali e si calcola come:

$$\mathrm{SCL}(Y, \hat{Y}) = \frac{\|Y - \hat{Y}\|_F}{\|Y\|_F} \tag{2}$$

  L'obiettivo è minimizzare questa funzione di perdita in modo tale che lo spettro predetto $\hat{Y}$ si avvicini il più possibile allo spettro reale Y. Questa loss è calcolata con la libreria "torch"di python.

- **LogMagnitudeSTFTLoss:** misura la differenza tra lo spettro di magnitudine in scala logaritmica di un segnale di riferimento Y e di un segnale predetto $\hat{Y}$, utilizzando la Trasformata di Fourier a breve termine (STFT). La formula è definita come:

$$\mathrm{LogMagSTFTLoss}(Y, \hat{Y}) = \| \log(|Y| + \epsilon) - \log(|\hat{Y}| + \epsilon)\|_F \tag{3}$$

  dove:

  - $|Y|$ è lo spettro di magnitudine del segnale di riferimento,
  - $|\hat{Y}|$ è lo spettro di magnitudine del segnale predetto,
  - ϵ è una piccola costante per evitare il logaritmo di zero,
  - $\| \cdot \|_F$ è la *norma di Frobenius.*

Questa perdita è particolarmente utile perché il logaritmo applicato alla magnitudine tiene conto: della percezione umana e delle differenze relative tra i livelli di intensità sonora, migliorando la qualità percepita del segnale audio elaborato. Questa loss è calcolata con la libreria "torch"di python.

- **SingleResolutionSTFTLoss:** è una funzione di perdita utilizzata per confrontare le rappresentazioni STFT di un segnale di riferimento Y e di un segnale predetto $\hat{Y}$. Questa perdita si calcola su una singola risoluzione della finestra STFT e include due termini principali:

$$\text{STFTLoss}(Y, \hat{Y}) = \frac{\| |Y| - |\hat{Y}| \|_F}{\| |Y| \|_F} + \| \log(|Y| + \epsilon) - \log(|\hat{Y}| + \epsilon) \|_F \quad (4)$$

dove:

  - $|Y|$ è lo spettro di magnitudine del segnale di riferimento,
  - $|\hat{Y}|$ è lo spettro di magnitudine del segnale predetto,
  - ϵ è una piccola costante per evitare problemi numerici nel calcolo del logaritmo,
  - $\| \cdot \|_F$ è la *norma di Frobenius*.

Il primo termine rappresenta la *Spectral Convergence*, che misura quanto le magnitudini degli spettri sono simili tra il segnale predetto e quello di riferimento. Il secondo termine confronta le magnitudini degli spettri in scala logaritmica per tener conto della sensibilità dell'orecchio umano alle differenze relative nei segnali audio. Questa loss è calcolata con la libreria "torch"di python.

- **MultiResolutionSTFTLoss:** è una funzione di perdita che confronta le rappresentazioni STFT di un segnale di riferimento Y e di un segnale predetto $\hat{Y}$, ma lo fa su diverse risoluzioni, utilizzando finestre e passi differenti per la trasformata di Fourier. La perdita è definita come:

$$\text{MR-STFTLoss}(Y, \hat{Y}) = \frac{1}{R} \sum_{r=1}^{R} \text{STFTLoss}_r(Y, \hat{Y}) \tag{5}$$

dove:

  - R è il numero di risoluzioni,
  - $\text{STFTLoss}_r(Y, \hat{Y})$ è la perdita calcolata alla risoluzione r:

$$\text{STFTLoss}_r(Y, \hat{Y}) = \frac{\| |Y_r| - |\hat{Y}_r| \|_F}{\| |Y_r| \|_F} + \| \log(|Y_r| + \epsilon) - \log(|\hat{Y}_r| + \epsilon) \|_F$$

  - $|Y_r|$ è lo spettro di magnitudine del segnale di riferimento alla risoluzione r,
  - $|\hat{Y}_r|$ è lo spettro di magnitudine del segnale predetto alla risoluzione r,
  - ϵ è una costante positiva per evitare instabilità numeriche.

Questa funzione di perdita permette di confrontare i segnali su più risoluzioni temporali e spettrali, migliorando l'accuratezza del confronto rispetto a una singola risoluzione. Questa loss è calcolata con la libreria "typing"di python.

3.3 Test

Per quanto riguarda quest'ultimo, seguendo [9], sono state simulate perdite uniformemente distanziate con un tasso di perdita del 10% e successivamente valutate le prestazioni PLC del metodo proposto rispetto al metodo zero-filling, cioè il riempimento di zero banale. In totale per il test sono state utilizzate 964 tracce speech contenute nella cartella blind-set. Oltre ai file audio la procedura di test richiede in input altri elementi:

1. il checkpoint dell'addestramento;
2. un file trace per ogni audio da analizzare.

Per quanto riguarda il checkpoint viene caricato un file .ckpt generato al termine dell'addestramento che contiene i parametri che la rete utilizzerà per la ricostruzione. I file .npy per le tracce devono invece essere generati manualmente. Ciò viene fatto mediante l'uso di un altro script appositamente creato che simula le perdite di pacchetto. Nel codice citato vengono letti sequenzialmente i file audio.wav presenti nella directory test del dataset utilizzando la libreria "LIBROSA". I pacchetti persi vengono simulati con una frequenza specificata nella variabile loss-rate. Successivamente viene calcolata la lunghezza della traccia audio in base alla lunghezza dei pacchetti specificata (10ms), quindi vengono impostati alcuni valori della traccia ad 1 per simulare la perdita dei pacchietti con frequenza pari a loss-rate. In figura 4 è illustrata la porzione di codice che implementa la creazione dei file trace.npy.

```python
def create_trace(audio_test_path: Path, trace_folder: Path, packet_dim: int, sr: int, loss_rate: int) -> None:
    # Load the clean signal
    y_true, sr = librosa.load(audio_test_path, sr=sr, mono=True)

    # Simulate packet losses
    trace_len = math.ceil(len(y_true) / packet_dim)
    trace = np.zeros(trace_len, dtype=int)
    trace[np.arange(loss_rate, trace_len, loss_rate)] = 1
    print(loss_rate)

    # Save trace
    if not os.path.exists(trace_folder):
        os.makedirs(trace_folder)
```

Fig.4 porzione di codice che implementa la simulazione PLC e crea le tracce con le informazioni

4 Metriche

4.1 Metriche oggettive

Al fine di valutare il lavoro di ricostruzione svolto sono state calcolate diverse metriche. Inizialmente come per [1] sono stati calcolati, per tutti i segnali, NMSE e Mel-sc. La prima calcola l'errore quadratico medio normalizzato ed è calcolato tra i pachetti stimati e quelli veri con la formula (6) seguente:

$$\text{NMSE} := 10 \log_{10} \left(\frac{\|y - \hat{y}\|_2^2}{\|y\|_2^2} \right) \tag{6}$$

Il fatto che venga calcolato l'MSE normalizzato è per compensare la dipendenza del termine di errore dall'energia del pacchetto ground-truth. Il Mel-sc calcola invece la convergenza spettrale come riportato in (7).

$$\text{Mel-SC} := \frac{\|Y_{\text{mel}} - \hat{Y}_{\text{mel}}\|_F}{\|Y_{\text{mel}}\|_F} \tag{7}$$

4.2 Metriche che simulano un giudizio soggettivo

Le metriche oggettive sono note per non essere così ben correlate con le valutazioni umane. Pertanto sono state incluse alcune misure fatte con algoritmi che tengono conto dell'ascolto umano:

1. PLCMOS, un codificatore convoluzionale ricorrente sviluppato per l'INTERSPEECH 2022 Audio Deep Packet Loss Concealment Challenge [14]. PLCMOS produce punteggi tra uno e cinque volti a stimare il Punteggio di Opinione Medio (MOS) degli ascoltatori umani secondo ITU-T Rec. P.808 [15]. PLCMOS è addestrato proprio sui segnali speech, quindi è idoneo alla valutazione per cui è utilizzato;

2. PESQ: è una metrica utilizzata per valutare la qualità percettiva del parlato in applicazioni di elaborazione del segnale, come la compressione audio, il miglioramento del parlato e la trasmissione su reti a pacchetto. Si tratta di un indice che misura la qualità del segnale vocale percepita dall'orecchio umano, prendendo in considerazione le distorsioni o le perdite introdotte da algoritmi o condizioni di trasmissione;

3. STOI: è una metrica utilizzata per misurare l'intelligibilità del parlato, ovvero quanto è comprensibile il parlato per l'orecchio umano. È ampiamente utilizzata nell'elaborazione del segnale audio, in particolare per valutare la qualità e l'intelligibilità di segnali vocali degradati o migliorati, come quelli affetti da rumore o trasmessi attraverso canali con interferenze;

4. DNSMOS: è una metrica basata su reti neurali profonde utilizzata per valutare la qualità audio o vocale, in particolare in contesti di elaborazione del parlato e comunicazione vocale. DNSMOS fornisce una stima della qualità percettiva del parlato, simile a quanto farebbe un punteggio di Mean Opinion Score (MOS) tradizionale, ma utilizzando tecniche di apprendimento automatico per ottenere valutazioni più precise e automatizzate.

5. Mel-cepstral: una rappresentazione parametrica utilizzata per valutare le PLC per segnali speech[1]. Deriva dalla combinazione della 'scala Mel' e del 'cepstrum'. Quest'ultimo è una trasformazione matematica che viene applicata al segnale audio e permette di analizzare gli aspetti sia rapidi che lenti della forma d'onda, facilitando l'analisi della voce e separando i componenti correlati alla voce stessa (come le caratteristiche

vocali) e ai filtri che influenzano il suono (come il tratto vocale). La scala Mel è una scala psicoacustica basata sulla percezione umana delle frequenze sonore. Essa è progettata per corrispondere alla sensibilità dell'orecchio umano che percepisce le differenze di frequenza in modo non lineare. Il Mel-cepstral combina queste due idee: si parte dal calcolo della rappresentazione spettrale di un segnale audio (attraverso la trasformata di Fourier), si converte questo spettro nella scala Mel e infine si applica la trasformazione cepstrale.

6. WER: è una metrica comunemente utilizzata per valutare la precisione dei sistemi di riconoscimento vocale, traduzione automatica e altri sistemi di elaborazione del linguaggio naturale. Misura l'accuratezza con cui un sistema riesce a trascrivere o tradurre il parlato o il testo, confrontando il testo prodotto dal sistema con il testo di riferimento corretto. Osservando (8) risulta:

$$\text{WER} = (\frac{S + D + I}{N}) \tag{8}$$

dove:

- S è il substitution, ossia numero di parole sostituite;
- D è il deletion: il numero di parole eliminate;
- I è l'insertion: il numero di parole inserite;
- N è il numero totale di parole nella trascrizione di riferimento.

Insieme al WER è stata calcolata la ACC(accuracy). Quest'ultima misura la percentuale di parole riconosciute confrontandole al riferimento. È definita come segue in (9):

$$\text{ACC} = (\frac{C}{N}) \tag{9}$$

dove:

- C il numero di parolo riconosciute correttamente;
- N è il numero totale di parole nella trascrizione di riferimento.

L'ACC è strettamente legato al WER quindi possiamo definire la seguente relazione diretta(10):

$$\text{ACC} = (1 - WER) \tag{10}$$

IL WER verrà valutato sui segnali ricostruiti. Se la ricostruzione con PARCnet è qualitativamente elevata, nei risultati ci si aspetta quindi un WER basso ed un ACC alto.

5 Risultati ottenuti

5.1 Tabelle

Le metriche sopra elencate sono state calcolate per tutti i file destinati al test set. Nella tabella 1 sono riportati i risultati ottenuti per 10 segnali scelti in modo random tra tutti quelli analizzati, elaborati con il metodo di ricostruzione che implementa PARCnet.

File name	**nmse sig** (dB)	**nmse pckt** (dB)	**mel-sc sig**	**mel-sc pckt**	**PESQ**	**STOI**	**DNS mos**	**PLC mos**	**Mel cepstral**
425422	-15,232	-6,685	0,104	0,283	2,596	0,924	3,264	2,818	-6,421
63022	-11,987	-2,672	0,158	0,524	3,152	0,960	2,570	2,365	-8,461
475285	-15,283	-6,288	0,098	0,268	3,096	0,969	3,923	3,249	-9,107
12865	-11,709	-2,418	0,149	0,404	2,388	0,973	3,489	2,875	-9,466
307528	-10,792	-1,626	0,156	0,418	2,398	0,949	3,385	2,673	-7,980
2307	-13,666	-4,048	0,117	0,298	2,814	0,977	3,259	3,064	-8,943
407950	-11,484	-2,607	0,161	0,422	2,040	0,967	3,677	3,190	-10,356
276335	-10,586	-1,367	0,167	0,493	2,863	0,964	3,637	2,874	-9,876
288077	-10,283	-1,028	0,188	0,593	1,949	0,956	3,482	1,874	-8,688
61483	-11,937	-3,027	0,137	0,371	3,072	0,965	3,819	3,241	-9,889

Tabella 1: *Risultati metriche calcolate con segnali elaborati con PARCnet*

La tabella 2 riporta i risultati ottenuti per gli stessi 10 segnali della tabella precedente, elaborati con il metodo zero-filling, ossia compensando la perdita dei pacchetti con riempimento di zeri.

File name	**nmse sig** (dB)	**nmse pckt** (dB)	**mel-sc sig**	**mel-sc pckt**	**PESQ**	**STOI**	**DNS mos**	**PLC mos**	**Mel cepstral**
425422	-10,128	0,000	0,220	1,000	1,409	0,907	2,261	2,767	-6,254
63022	-10,176	0,000	0,196	1,000	2,153	0,937	2,134	2,199	-8,403
475285	-9,853	0,000	0,230	1,000	1,487	0,917	2,536	3,038	-8,890
12865	-10,419	0,000	0,188	1,000	1,620	0,956	2,314	2,921	-9,387
307528	-9,921	0,000	0,218	1,000	1,973	0,947	2,591	2,805	-7,891
2307	-10,522	0,000	0,212	1,000	1,553	0,947	2,350	3,058	-8,806
407950	-9,861	0,000	0,217	1,000	1,576	0,946	2,027	3,288	-10,314
276335	-9,925	0,000	0,212	1,000	2,177	0,956	2,725	2,760	-9,958
288077	-10,031	0,000	0,198	1,000	1,788	0,954	2,697	1,890	-8,640
61483	-9,983	0,000	0,211	1,000	2,250	0,959	3,069	3,285	-9,839

Tabella 2: *Risultati metriche calcolate con segnali elaborati con zero-filling*

Nella tabella 3 sono indicati i valori di media e varianza calcolati con le metriche di tutti i 964 segnali utilizzati per il test set. In figura 5 è illustrato l'istogramma che mostra i valori medi calcolati su tutti i segnali utilizzati per il test.

Metrics-Name	Mean	Variance
zero-filling-nmse-signal	-10,010	0,128
zero-filling-nmse-packet	0,000	0,000
parcnet-nmse-signal	-13,039	3,787
parcnet-nmse-packet	-4,089	4,234
zero-filling-mel-sc-signal	0,216	0,000
zero-filling-mel-sc-packet	1,000	0,000
parcnet-mel-sc-signal	0,144	0,161
parcnet-mel-sc-packet	0,385	0,764
zero-filling-PESQ	1,697	0,082
parcnet-PESQ	2,685	0,102
zero-filling-STOI	0,933	0,001
parcnet-STOI	0,962	0,001
dnsmos-zero-filling	2,386	0,090
dnsmos-parcnet	3,588	0,232
plcmos-zero-filling	2,887	0,105
plcmos-parcnet	2,933	0,099
mel-cepstra-zero-filling	-9,053	1,029
mel-cepstra-parcnet	-9,135	0,907

Tabella 3: *Media e varianza calcolata sui valori delle metriche di tutti i segnali di test*

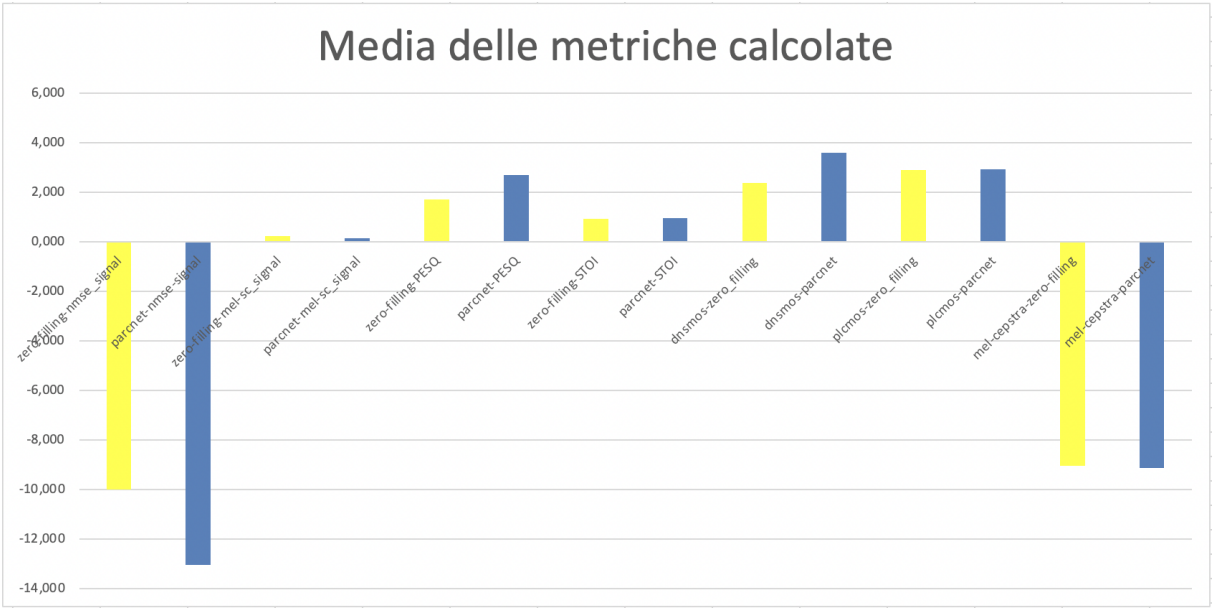

Fig.5 Istrogramma dei valori medi delle metriche calcolate

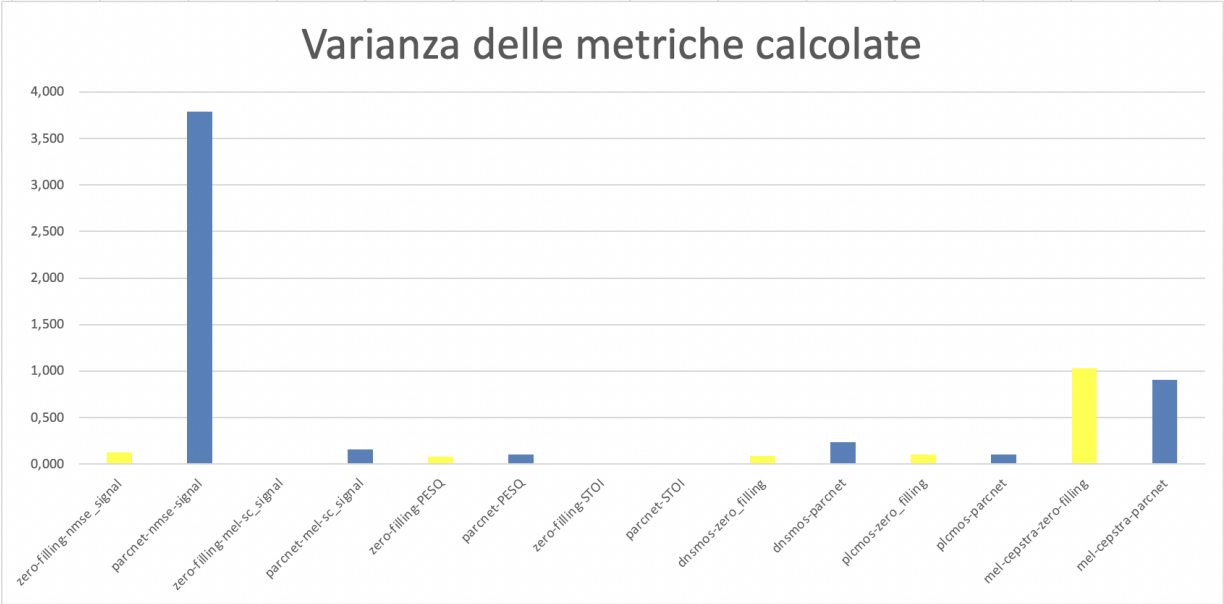

Fig.6 Istrogramma della varianza delle metriche calcolate

5.2 Grafici

Dopo la procedura di test, si è passato a visualizzare alcune ricostruzioni fatte con i metodi sopracitati. In figura 7 è illustrata una porzione del file "54.wav" contenente: l'andamento del segnale di riferimento, la ricostruzione fatta con zero-filling e la ricostruzione eseguita con PARCnet. Nelle figure 8 e 9 sono illustrate le stesse caratteristiche simulate sui segnali "16976.wav" e "44660.wav"

```
plt.plot(y_clean[2000:3000],label= 'clean')
plt.plot(y_zerofilling[2000:3000], label ='zero_filling')
plt.plot(y_parcnet[2000:3000], label= 'PARCnet')
plt.legend(loc='upper right')
plt.show()
```

✓ 0.1s

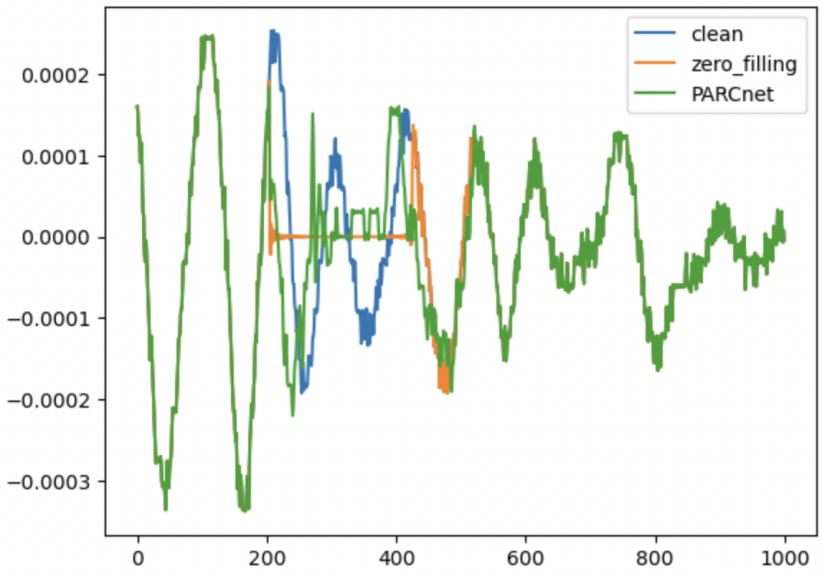

Fig.7 Porzione del segnale 54.wav: originale, zero-filling, PARCnet

```
plt.plot(y_clean[2000:2500],label= 'clean')
plt.                                        zero_filling')
plt.                                        net')
plt.legend(loc='upper right')
plt.show()
```

(function) def legend(...) -> Legend

✓ 0.1s

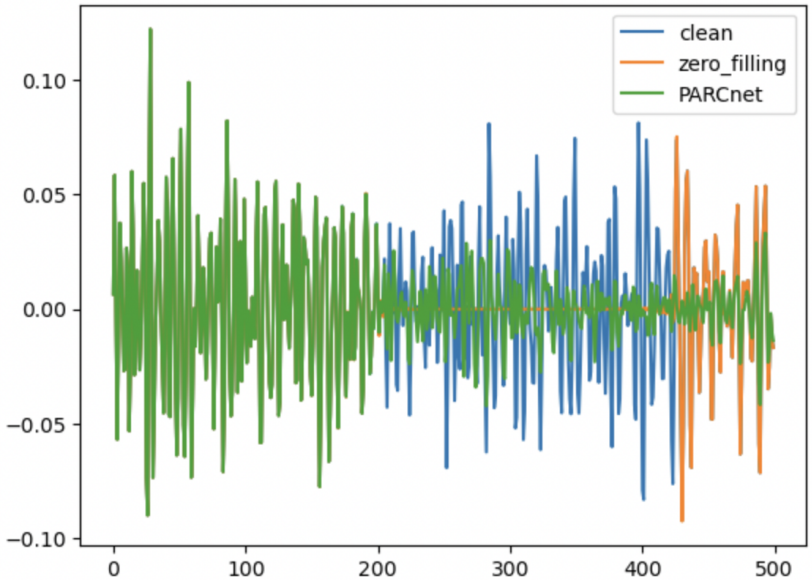

Fig.8Porzione del segnale 16976.wav: originale, zero-filling, PARCnet

```
plt.plot(y_clean[2000:2500],label= 'clean')
plt.plot(y_zerofilling[2000:2500], label ='zero_filling')
plt.plot(y_parcnet[2000:2500], label= 'PARCnet')
plt.legend(loc='upper right')
plt.show()
```

✓ 0.1s

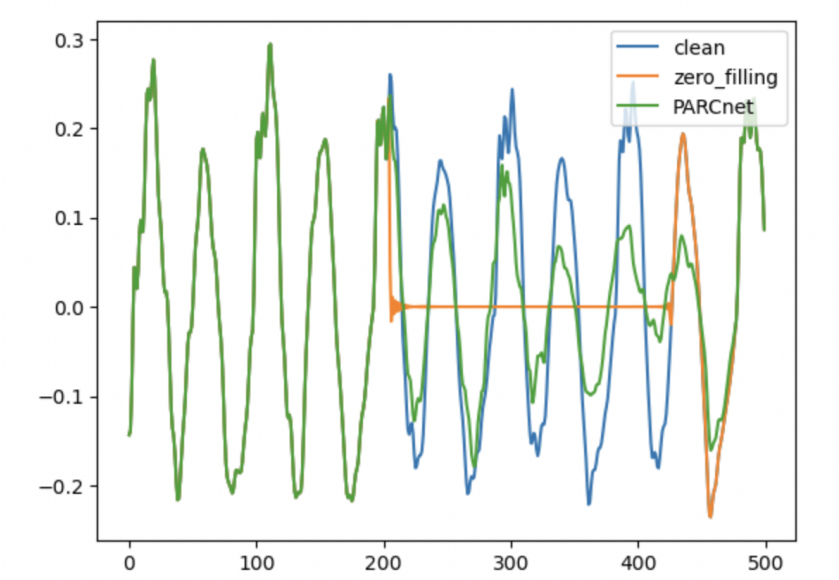

Fig.9 Porzione del segnale 44660.wav: originale, zero-filling, PARCnet

5.3 Word Error Rate (WER)

Come indicato nel paragrafo 4 è stato calcolato il Word Error Rate. Ciò e stato possibile utilizzando le librerie "wishper" e "jiver". Whisper è un modello di riconoscimento vocale (automatic speech recognition, ASR) sviluppato da OpenAI. È progettato per convertire l'audio contenente parlato in testo scritto. JiWER è la libreria appositamente creata per il calcolo della metrica in questione. La procedura è la seguiente: dopo aver elaborato i file di test vengono dati in pasto all'algoritmo di speech recognition il quale fa una trascrizione di tutte le parole che riconosce in ogni segnale audio considerato; successivamente, l'algoritmo che calcola il WER, confronta le parole individuate da whisper con il testo di riferimento fornito in un file.csv allegato e calcola la metrica. In figura 10 è illustrata una porzione dell' output generato

da Whisper (hypothesis) ed il testo di riferimento (reference). Dal calcolo risultata un WER pari al 10.06% ed una conseguente accuratezza ACC pari a 89.94%. La figura 11 illustra la porzione di codice che implementa il WER con i risultati ottenuti dal calcolo.

	hypothesis	reference
0	Before he leaves for basic training, why did h...	Before he leaves for basic training, why did ...
1	occasionally is a selection from the weekly po...	Occasionally is a selection from the weekly p...
2	Beverage wise is any kind of dairy product. Mi...	Beverage wise it's any kind of dairy product....
3	That's just joy of creating and making it happ...	That's just joy of creating and making it hap...
4	out of six, I think history of the decline of ...	Out of six, I think history of the decline an...
...	...	...
958	So Jim, enjoy your well-earned respite from th...	So Jim, enjoy your well earned respite from t...
959	With his lace and embroideries and his crown o...	(excluded)
960	book suggestions board. And it comes from the ...	Book Suggestions Board and it comes from the ...
961	Sometimes it sounds more mellow and then it so...	Sometimes it's it's. I don't know, sounds mor...
962	I went looking online to see if there was a co...	(excluded)

963 rows × 4 columns

Fig.10 Testo ipotizzato e di riferimento

```
# Filtra i dati per escludere riferimenti o ipotesi vuote
reference_clean = list(data["reference_clean"])
hypothesis_clean = list(data["hypothesis_clean"])

# Crea una nuova lista per includere solo coppie non vuote
filtered_reference = []
filtered_hypothesis = []

for ref, hyp in zip(reference_clean, hypothesis_clean):
    if ref.strip() and hyp.strip():  # Escludi le stringhe vuote o spazi bianchi
        filtered_reference.append(ref)
        filtered_hypothesis.append(hyp)

# Calcola il WER solo sulle coppie non vuote
if filtered_reference and filtered_hypothesis:
    wer = jiwer.wer(filtered_reference, filtered_hypothesis)
    print(f"WER: {wer * 100:.2f} % - ACC: {(1-wer) * 100:.2f}")
else:
    print("Non ci sono dati validi per calcolare il WER.")
```

```
WER: 10.06 % - ACC: 89.94
```

Fig.11 WER codice e risultati

6 Analisi dei risultati e osservazioni

Osservando le tabelle sopra illustrate si possono notare diversi particolari. Per primo, alcune metriche sono calcolate signal e packet. Il termine "signal" indica che la valutazione è fatta sull'intera ricostruzione del segnale in test; con la dicitura "packet" viene indicato il calcolo effettuato non sul segnale intero bensì sulle porzioni dove dal trace.npy è indicata la perdita di pacchetti e quindi viene valutata la ricostruzione solo in quelle porzioni. Ovviamente in entrambi i casi il segnale ricostruito è confrontato con quello di riferimento. Un altro aspetto da evidenziare è la presenza di valori sempre uguali a zero. Come si può notare, la colonna "nmse-packet(dB)" della tabella 2 è piena di valori nulli. Questo è dovuto al calcolo della metrica. Come detto in precedenza il tag "packet" indica che il calcolo è effettuato solo sui pacchetti considerati dalla simulazione come "persi". Poiché facendo zero-filling i campioni vengono banalmente sostituiti con zeri, il calcolo NMSE risulta pari a zero. Per tutte le metriche calcolate si nota una migliore prestazione dell'algoritmo PARCnet rispetto al metodo zero-filling. Per quanto rifuarda i grafici riportati, è ben visibile come la ricostruzione fatta con PARCnet segua l'andamento del segnale originale rispetto alla ricostruzione zero-filling che invece risulta come una continuità di zeri. In riferimento al fatto che i segnali elaborati sono speech (quindi contengono porzioni silenziose dovute alle pause del parlato) la ricostruzione con il metodo zero-filling è da considerarsi migliore in termini di accettabilità rispetto all'utilizzo dello stesso metodo per file audio musicali. Le prestazioni dell'algoritmo cambiano notevolmente variando la lunghezza dei pacchetti persi. Infatti accorciando la dimensione dei lost ad esempio da 10ms a 5ms ,figura 12, la rete riesce

a fornire una ricostruzione piu fedele. Al contrario allungando la dimensione dei pacchetti figura 13 le prestazioni degradano ma sono molto migliori rispetto al metodo tradizionale.

```
plt.plot(y_clean[2000:2500],label= 'clean')
plt.plot(y_zerofilling5ms[2000:2500], label ='zero_filling_5ms')
plt.plot(y_parcnet5ms[2000:2500], label= 'PARCnet_5ms')
plt.legend(loc='upper right')
plt.show()
```

✓ 0.1s

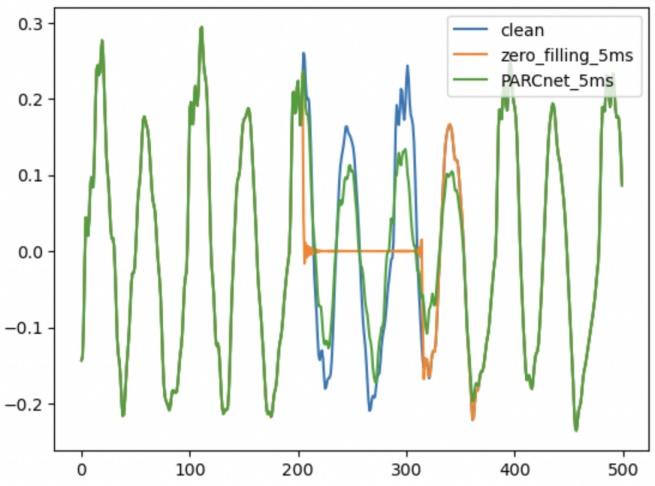

Fig.12 Test con packet-dim=5ms

```
plt.plot(y_clean[4000:5000],label= 'clean')
plt.plot(y_zerofilling20ms[4000:5000], label ='zero_filling_20ms')
plt.plot(y_parcnet20ms[4000:5000], label= 'PARCnet_20ms')
plt.legend(loc='upper right')
plt.show()
```

✓ 0.1s

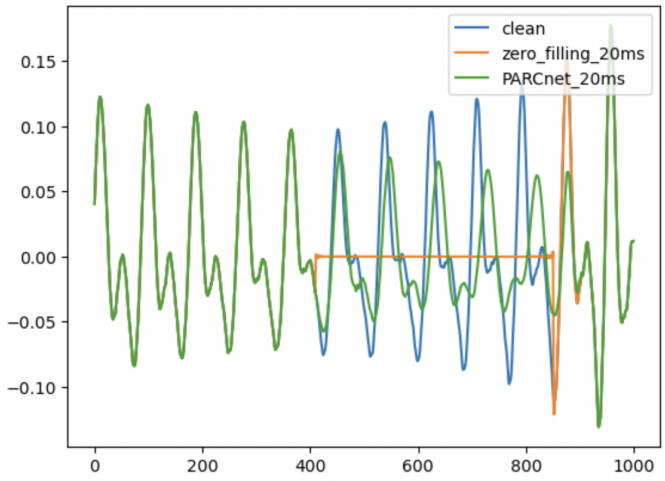

Fig.13 Test con packet-dim=20ms

7 Conclusioni

In questo elaborato è stato implementato un algoritmo PLC per segnali speech utilizzando la rete PARCnet, quindi una rete ibrida concepita per fare PLC con segnali audio musicali [1]. I risultati sono stati ottenuti effettuando il train della rete adattandola alle caratteristiche dei segnali speech utilizzati e le prestazioni ottenute sono perfettamente in linea con le aspettative. Infatti, la rete riesce a ricostruire in modo notevole i pacchetti persi con prestazioni superiori alle tecniche precedenti come la compensazione di zeri. In fase di addestramento sono state implementare alcune modifiche con lo scopo di migliorare le caratteristiche della rete: è stato aggiunto l'AttentionBlock per aiutare la rete a focalizzarsi su regioni significative del segnale audio durante il processo di decodifica, il blocco LSTM dopo il blocco GLU per catturare le dipendenze temporali a lungo termine. Tali ottimizzazioni non hanno portato miglioramenti in termini di loss una volta concluso l'addestramento.

8 Bibliografia

[1] A. I. Mezza, M. Amerena, A. Bernardini and A. Sarti, "Hybrid Packet Loss Concealment for Real-Time Networked Music Applications," in IEEE Open Journal of Signal Processing, vol. 5, pp. 266-273, 2024, doi: 10.1109/OJSP.2023.3343318.

[2] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," IEEE Netw., vol. 12, no. 5, pp. 40–48, Sep./Oct. 1998.

[3] SubstitutionandMutingofLostFramesforFullRateSpeechChannels, Rec. ETSI GSM 6.11, European Telecommunications Standards Insti- tute, Sophia-Antipolis, France, Feb. 1992.

[4] J. Yeh, P. Lin, M. Kuo, and Z. Hsu, "Bilateral waveform similarity overlap-and-add based packet loss concealment for voice over IP," J. Appl. Res. Technol., vol. 11, pp. 559–567, 2013.

[5] K. Kondo and K. Nakagawa, "A speech packet loss concealment method using linear prediction," IEICE Trans. Inf. Syst., vol. 89, no. 2, pp. 806–813, 2006.

[6] G. Zhang and W. B. Kleijn, "Autoregressive model-based speech packet-loss concealment," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2008, pp. 4797–4800.

[7] T. L. Jensen, D. Giacobello, T. van Waterschoot, and M. G. Christensen, "Fast algorithms for high-order sparse linear prediction with applica- tions to speech processing," Speech Commun., vol. 76, pp. 143–156, 2016.

[8] M. Sacchetto, Y. Huang, A. Bianco, and C. Rottondi, "Using autore- gressive models for real-time packet loss concealment in networked music performance applications," in Proc. Int. Audio Mostly Conf., 2022, pp. 203–210.

[9] P. Verma, A. I. Mezza, C. Chafe, and C. Rottondi, "A deep learning approach for low-latency packet loss concealment of audio signals in networked music performance applications," in Proc. Conf. Open Innov. Assoc., 2020, pp. 268–275.

[10] J. Wang, Y. Guan, C. Zheng, R. Peng, and X. Li, "A temporal- spectral generative adversarial network based end-to-end packet loss concealment for wideband speech transmission," J. Acoust. Soc. Amer., vol. 150, no. 4, pp. 2577–2588, 2021.

[11] J.Pons,S.Pascual,G.Cengarle,andJ.Serrà,"Upsampling artifacts in neural audio synthesis," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2021, pp. 3005–3009.

[12] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in Proc. Int. Conf. Mach. Learn., 2017, pp. 933–941.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1026–1034.

[14] L. Diener, S. Sootla, S. Branets, A. Saabas, R. Aichner, and R. Cutler, "INTERSPEECH 2022 audio deep packet loss concealment challenge," in Proc. Interspeech, 2022, pp. 580–584.

[15] Subjective Evaluation of Speech Quality With a Crowdsourcing Ap-proach, Rec. ITU-T P.808, International Telecommunications Union, Geneva, Switzerland, Jun. 2021.

[16] Diener, Lorenz, Sootla, Sten, Branets, Solomiya, Saabas, Ando, Aichner, Robert, Cutler, Ross. (2022). INTERSPEECH 2022 Audio Deep Packet Loss Concealment Challenge. 10.48550/arXiv.2204.05222.