2024年度後期　修士論文

深層学習による口唇動画を用いた音声合成に関する研究

A Study on Speech Synthesis from Lip Video Using Deep Learning

2025年1月15日

九州大学芸術工学府音響設計コース

2DS23095M

南 汰翼

MINAMI Taisuke

研究指導教員　鏑木 時彦　教授

概要

本研究では，代用音声の新たな手法となりうる動画音声合成について，近年有効性の示された手法をベースラインとし，大規模音声データによって事前学習されたモデルの転移学習に着目した，新たな手法を提案した．実験の結果，客観評価と主観評価の両面において，提案手法がベースラインを上回る明瞭性および類似性を達成したことが示された．

目 次

1 序論 **1**
1.1 背景 . . . 1
1.2 目的 . . . 2
1.3 本論文の構成 . . . 2

2 音声信号処理 **3**
2.1 音声のフーリエ変換 . . . 3
2.2 メルスペクトログラム . . . 5

3 深層学習 **7**
3.1 DNN の構成要素 . . . 7
3.1.1 全結合層 . . . 7
3.1.2 畳み込み層 . . . 7
3.1.3 転置畳み込み層 . . . 8
3.1.4 活性化関数 . . . 9
3.1.5 再帰型ニューラルネットワーク . . . 11
3.1.6 正規化層 . . . 13
3.1.7 Transformer . . . 14
3.2 学習方法 . . . 16
3.2.1 損失関数 . . . 16
3.2.2 勾配降下法 . . . 17
3.2.3 正則化 . . . 18
3.2.4 最適化手法 . . . 19
3.2.5 学習率のスケジューリング . . . 21
3.2.6 誤差逆伝播法 . . . 21
3.2.7 学習の安定化 . . . 24

4 動画音声合成モデルの検討 **25**
4.1 音声合成法 . . . 25
4.1.1 全体像 . . . 25
4.1.2 ネットワーク A . . . 26
4.1.3 ネットワーク B . . . 28
4.1.4 ボコーダ . . . 30
4.1.5 損失関数 . . . 31
4.2 実験方法 . . . 32
4.2.1 利用したデータセット . . . 32

4.2.2 データの前処理 32
4.2.3 本実験で利用した事前学習済みモデルについて 33
4.2.4 本実験で独自に構築したモデルについて 34
4.2.5 学習方法 35
4.2.6 客観評価 36
4.2.7 主観評価 37
4.3 結果 40
4.3.1 客観評価 1: ベースラインと提案手法の比較 40
4.3.2 客観評価 2: 提案手法のさらなる検討 41
4.3.3 主観評価 45
4.4 考察 47
4.4.1 ネットワーク B（Randomized）について 47
4.4.2 ネットワーク B（Pretrained）について 48
4.5 今後の課題 48

5 結論 **49**

謝辞 **50**

参考文献 **51**

1　序論

1.1　背景

音声は基本的なコミュニケーション手段として，人々の日常生活において重要な役割を果たしている．しかし，癌などの病気で喉頭を摘出すると，声帯振動による音声生成が不可能になり，従来の発声手段を失ってしまう．このような場合の代用音声手法として，電気式人工喉頭や食道発声，シャント発声がある．しかし，これらには人工的な音声になる，習得に訓練が必要である，器具の交換のため定期的な手術を要するなどの課題がある．そこで本研究では，新たな代用音声手法として，ビデオカメラで撮影された口唇動画を入力として音声を合成する，深層学習モデルを利用したアプローチを提案する．この手法により，訓練や手術を必要とせずとも，自然な声でのコミュニケーションを可能にすることを目指す．

従来の動画音声合成では，動画からの予測対象にメルスペクトログラムが選択されることが多かった．例えば，テキスト音声合成で高い品質を達成した系列変換モデル [1] を動画音声合成に応用した Lip2Wav[2] や，GAN（Generative Adversarial Network）を活用した VCA-GAN[3]，畳み込み層と Transformer[4] から構成される Conformer[5] を利用した SVTS[6] などがある．これらでは，損失関数として MAE Loss や MSE Loss が用いられ，原音声から計算されるメルスペクトログラムと，予測されたメルスペクトログラムの間の距離を最小化するよう学習が行われた．これに対し，[7] では，メルスペクトログラムは発話内容だけでなく，話者性やイントネーションといった音響的特徴を含むため，発話内容に対する正確な制約を十分に与えられないことを課題として指摘した．この課題に対し，モデルの中間特徴量を用いたテキストの予測によって計算される CTC（Connectionist Temporal Classification） Loss と，メルスペクトログラムを事前学習済み音声認識モデルによって変換した特徴量における MSE Loss を損失に加え，発話内容に関する制約を強化した．これにより，客観評価において従来手法を上回る結果が得られた．これに続き，[8] では手法 [7] が教師ラベルとしてテキストを必要とするため，テキストアノテーションされていないデータで用いることができない点を課題として指摘した．この課題に対し，言語情報を中心とした特徴量として扱うことが可能な，HuBERT[9] を利用して得られた離散特徴量をテキストに代わる予測対象として導入した．また，モデルから予測されるメルスペクトログラムがノイジーになってしまう課題に対し，言語情報である HuBERT 離散特徴量も入力として情報を補完する，Multi-input Vocoder が合わせて提案された．これにより，客観評価と主観評価の両面において従来手法を上回る結果が得られた．加えて，[8] では AVHuBERT[10] の fine-tuning も検討された．AVHuBERT は，英語動画音声データセットである LRS3[11] と，多言語動画音声データセットである VoxCeleb2[12] の英語データを用いて，動画と音声の間の複雑な関係を Masked Prediction という自己教師あり学習方法によって学習したモデルである．動画音声合成に対する fine-tuning の結果，AVHuBERT を用いない場合と比較して，客観評価指標における改善が確認された．

1.2　目的

本研究では，動画音声合成モデルによって得られる合成音声の品質が依然として低く，自然音声に迫る合成音の実現に至っていない点を課題とする．この課題に対し，近年高い精度を達成した,「AVHuBERT を利用したメルスペクトログラムと HuBERT 離散特徴量を予測対象とするマルチタスク学習手法」をベースラインとして採用し，この手法を上回る新たなモデルを提案することで，自然音声に迫る合成音声の実現に近づくことを目的とする．

ここで，近年有効性の示された，テキストや HuBERT 離散特徴量を利用するマルチタスク学習手法は，動画を入力とするモデルの学習方法に対する工夫だったと考えられる．しかし，同じような口唇の動きでも場合によって異なる音素となる場合があることが，特に動画音声合成を困難にしていると考える．これに対して本研究では，動画を入力として最終予測値を出力する従来のネットワークとは異なる新たなネットワークを導入し，これらを組み合わせることで精度改善を狙った.

1.3　本論文の構成

本論文は本章を含め，全 5 章から構成される．2 章では，音声データを取り扱う上で必要な信号処理について述べる．3 章では，動画から音声を予測するために用いた深層学習について述べる．4 章では，本研究の提案手法とベースラインとの比較について述べる．これを踏まえ，5 章では本研究を通した結論を述べる．

2　音声信号処理

音声にはフォルマントや基本周波数（ピッチ）など，様々な周波数的な特徴が存在している．フォルマントは母音や子音を知覚するため，ピッチはアクセントやイントネーションを表現するために重要なものである．このような音声信号の持つ複雑さから，時間波形のままその特徴を分析することは困難である．これに対し，本節では音声の特徴を捉えやすくするための信号処理について説明する．

2.1　音声のフーリエ変換

音声の時間波形から周波数領域の情報を得るためには，フーリエ変換（Fourier transform）が用いられる．音声は通常マイクロフォンで収録され，コンピュータ内で処理される際には，アナログ信号ではなくデジタル信号として扱われる．このデジタル信号は，サンプリング周波数と量子化ビット数に従って離散化されている．離散化された信号に対しては，離散フーリエ変換（discrete Fourier transform; DFT）が適用される．また，信号の系列長をゼロパディングして 2 のべき乗の長さに調整すれば，計算量を抑えた高速フーリエ変換（fast Fourier transform; FFT）を用いることができる．

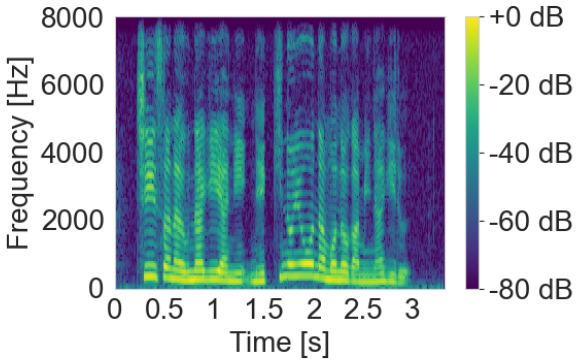
離散信号を $x[n]$，それに対するフーリエ変換を $X[f]$ とする．ここで，n はサンプルのインデックス，f は周波数インデックスである．$X[f]$ は複素数であるから，以下のように極座標表示することができる．

$$X[f] = \mathrm{Re}(X[f]) + j\,\mathrm{Im}(X[f]) \tag{2.1}$$
$$= |X[f]| \exp^{j\angle X[f]} \tag{2.2}$$

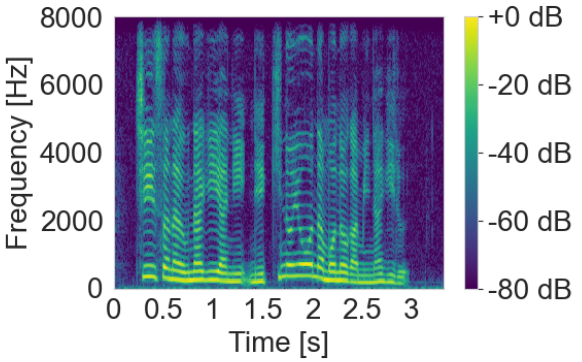
ここで，$|X[f]|$ は振幅特性（振幅スペクトル），$\angle X[f]$ は位相特性（位相スペクトル）であり，

$$|X[f]| = \sqrt{\mathrm{Re}(X[f])^2 + \mathrm{Im}(X[f])^2} \tag{2.3}$$
$$\angle X[f] = \tan^{-1} \frac{\mathrm{Im}(X[f])}{\mathrm{Re}(X[f])} \tag{2.4}$$

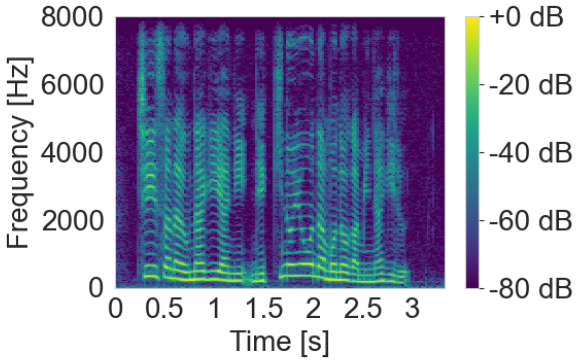
と表される．また，$|X[f]|^2$ はパワースペクトルと呼ばれる．これにより，信号中にどのような周波数成分がどれくらい含まれているかを調べることができる．しかし，音声はフォルマントやピッチが時々刻々と変化するため，信号全体に対して直接フーリエ変換を適用したとしても有用な結果が得られない．このような音声の持つ非定常性の問題に対して，十分短い時間幅においては信号の定常性が成り立つという仮定のもと，短時間フーリエ変換（short-time Fourier transform; STFT）が用いられる．STFT では，音声信号に対して窓関数による窓処理を適用し，短時間に区切られた信号それぞれに対して DFT を適用する．ここで，窓処理とはある特定の窓関数と音声信号を時間領域でかけ合わせることであり，窓関数の時間幅を窓長という．また，窓関数を時間方向にシフトするときの時間幅をシフト幅という．STFT には，時間分解能



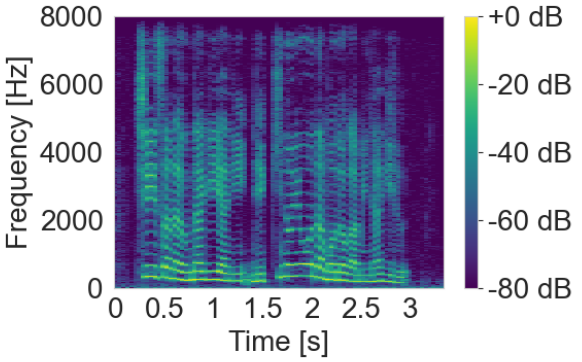
(a) 窓長 12.5 ms，シフト幅 5 ms



(b) 窓長 25 ms，シフト幅 10 ms



(c) 窓長 50 ms，シフト幅 20 ms



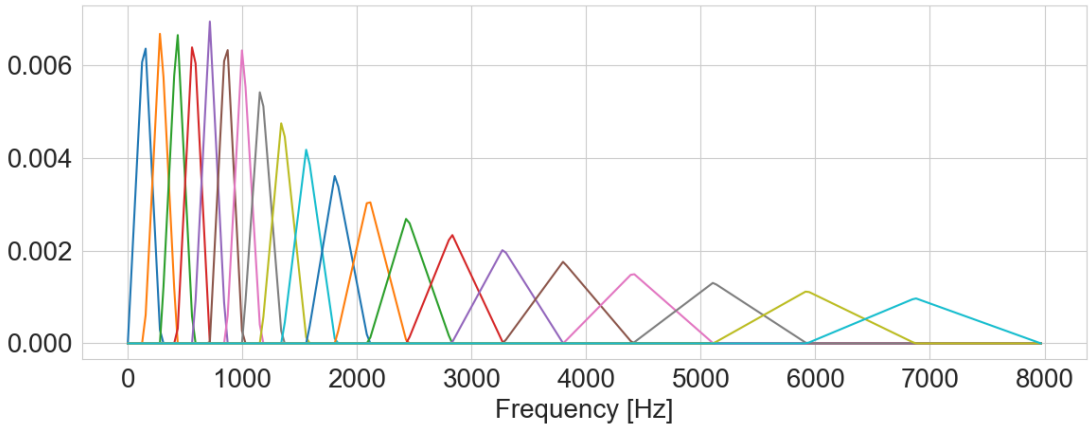
(d) 窓長 100 ms，シフト幅 40 ms

図 2.1: 「小さな鰻屋に，熱気のようなものがみなぎる」と発話した音声から計算された対数パワースペクトログラム

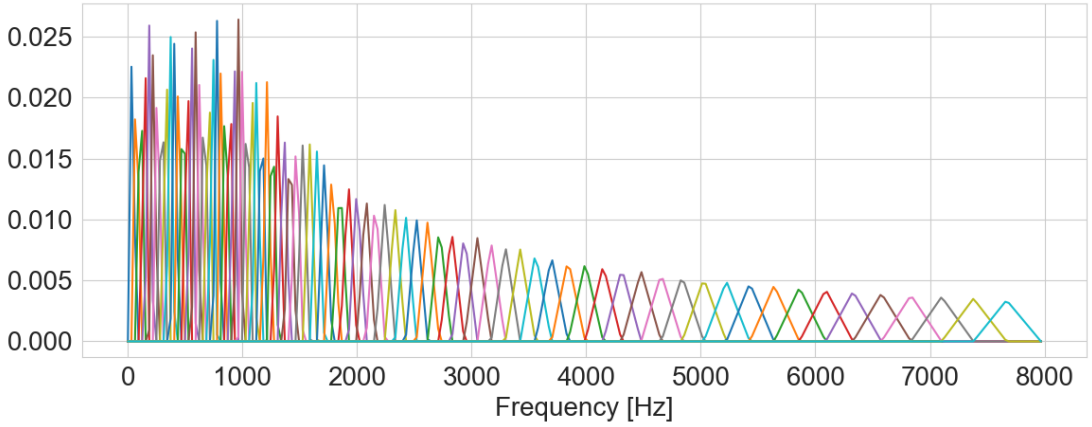
と周波数分解能の間にトレード・オフの関係がある．窓長が長い場合には周波数分解能が向上する一方，時間分解能が低下する．窓長が短い場合にはその逆となる．音声信号 $x[n]$ の STFT を時刻 t，周波数インデックスを f として $X[t,f]$ と表すと，$X[t,f]$ は時間周波数領域における複素数となる．これを複素スペクトログラムと呼ぶ．また，$|X[t,f]|$ を振幅スペクトログラム，$\angle X[t,f]$ を位相スペクトログラム，$|X[t,f]|^2$ をパワースペクトログラムと呼ぶ．ここで，「小さな鰻屋に，熱気のようなものがみなぎる」と発話した，サンプリング周波数 16 kHz の音声波形に対し，窓関数としてハニング窓を用いた上で，複数の窓長・シフト幅によって計算した対数パワースペクトログラムを，図 2.1に示す．窓長が 100 ms と長い場合には周波数分解能が高いが，時間分解能が低下することでスペクトルの時間変化が滑らかでないことがわかる．一方，窓長が 12.5 ms と短い場合には時間分解能が高いが，周波数分解能が低下することでスペクトルがぼやけていることがわかる．これが窓長に対する時間分解能と周波数分解能のトレード・オフであり，窓長 25 ms や 50 ms が程よいパラメータであることがわかる．

2.2　メルスペクトログラム

メルスペクトログラムは，パワースペクトログラムにメルフィルタバンクをかけることによって得られる音響特徴量であり，音声認識や音声合成といったタスクにおいて広く用いられている．メルフィルタバンクは，周波数軸を人間の聴感特性を考慮して変換したメル尺度上で，指定した数のフィルタを等間隔に配置して得られる．図 2.2に，フィルタを配置する帯域を 0 kHz から 8 kHz とした場合におけるメルフィルタバンクを示す．中心周波数の低いフィルタほど帯域幅が狭くなっており，低域ほど細かく，高域ほど荒く周波数成分を扱っていることがわかる．また，フィルタ数を 20 から 80 に増やすことによって各フィルタの帯域幅が狭くなり，より細かく周波数成分を考慮できることがわかる．例として,「小さな鰻屋に，熱気のようなものがみなぎる」と発話した，サンプリング周波数 16 kHz の音声波形に対し，窓長 25 ms のハニング窓を用いてシフト幅 10 ms でパワースペクトログラムを計算し，フィルタ数 80 のメルフィルタバンクを適用して得られた対数メルスペクトログラムを図 2.3に示す.



(a) フィルタ数を 20 とした場合



(b) フィルタ数を 80 とした場合

図 2.2: フィルタを配置する帯域を 0 kHz から 8 kHz とした場合におけるメルフィルタバンク

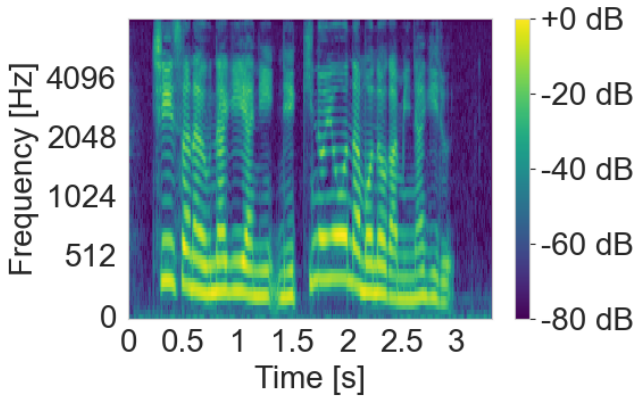


図 2.3: 「小さな鰻屋に，熱気のようなものがみなぎる」と発話した音声に対する対数メルスペクトログラム

3　深層学習

深層学習とは，人間の神経細胞の仕組みを模擬したニューラルネットワークを用いる機械学習手法のことである．特に近年ではその層を深くしたディープニューラルネットワーク（Deep Neural Network; DNN）が用いられ，大量のパラメータによる表現力により，自然言語処理や画像処理，音声処理など様々な分野で成果を上げている．本章では，DNN の構成要素及び，構築した DNN の学習方法について説明する．

3.1　DNN の構成要素

3.1.1　全結合層

全結合層は，入力に対して線型変換を施す層である．全結合層への入力を $\boldsymbol{x} \in \mathbb{R}^{D_{\text{in}}}$ とすると，出力 $\boldsymbol{y} \in \mathbb{R}^{D_{\text{out}}}$ は，

$$\boldsymbol{y} = \mathrm{FC}\left(\boldsymbol{x}; \boldsymbol{\theta}\right) = \left(\boldsymbol{x}^\top \boldsymbol{W}\right)^\top + \boldsymbol{b} \tag{3.1}$$

で与えられる．ここで，$D_{\text{in}}, D_{\text{out}}$ は入出力の次元，$\boldsymbol{W} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$ は重み，$\boldsymbol{b} \in \mathbb{R}^{D_{\text{out}}}$ はバイアスであり，$\boldsymbol{\theta}$ は重みとバイアスをまとめて表す変数とする．また，入力が行列 $\boldsymbol{X} \in \mathbb{R}^{T \times D_{\text{in}}}$ である時，出力 $\boldsymbol{Y} \in \mathbb{R}^{T \times D_{\text{out}}}$ は，

$$\boldsymbol{Y} = \mathrm{FC}\left(\boldsymbol{X}; \boldsymbol{\theta}\right) = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}\boldsymbol{1}^\top \tag{3.2}$$

で与えられる．ここで，T は系列長，$\boldsymbol{1} \in \{1\}^T$ は全成分が 1 のベクトルである．全結合層は DNN 内部での特徴量の次元の変換や，最終層において所望の出力に次元を合わせるのに用いられる．

3.1.2　畳み込み層

畳み込み層は，入力に対して畳み込み演算を行う層である．1 次元畳み込み層について，入力を $\boldsymbol{X} \in \mathbb{R}^{D_{\text{in}} \times T_{\text{in}}}$，出力を $\boldsymbol{Y} \in \mathbb{R}^{D_{\text{out}} \times T_{\text{out}}}$ とし，それぞれ d 次元目，t 番目の成分を $x_{d_{\text{in}},t}, y_{d_{\text{out}},t}$ で表す．$D_{\text{in}}, D_{\text{out}}$ は入出力の次元，$T_{\text{in}}, T_{\text{out}}$ は入出力の系列長である．このとき，$y_{d_{\text{out}},t}$ は，

$$y_{d_{\text{out}},t} = b_{d_{\text{out}}} + \sum_{d_{\text{in}}=1}^{D_{\text{in}}} \sum_{k=1}^{K} x_{d_{\text{in}},t-\left\lfloor \frac{K}{2} \right\rfloor+k-1} w_{d_{\text{in}},d_{\text{out}},k} \tag{3.3}$$

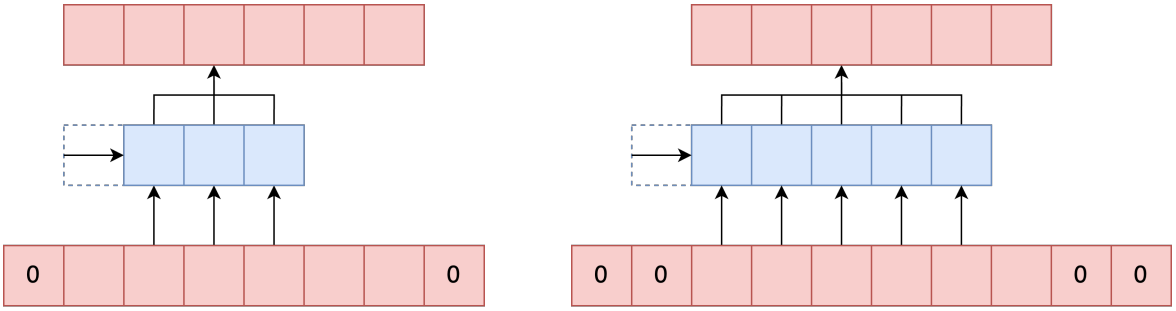
で与えられる．ここで，K はカーネルサイズ，$w_{d_{\text{in}},d_{\text{out}},k}$ は入力の d_{in} 次元目から出力の d_{out} 次元目に割り当てられたカーネルの k 番目の成分，$b_{d_{\text{out}}}$ は出力の d_{out} 次元目に割り当てられたバイアスである．上式より，1 次元畳み込み層の t 番目の出力は，t 番目を中心としたカーネルサイズの範囲分の入力から計算されることがわかる（K は奇数を想定した）．これより，畳み込み層は入力の局所的な特徴を抽出するのに適した層だと考えられる．1 次元畳み込みはテキストや音声など，データの形状が $(D \times T)$ となっている時に用いられる．

これに加えて，カーネルを 2 次元配列とすれば 2 次元畳み込み層，3 次元配列とすれば 3 次元畳み込み層となる．2 次元畳み込み層は画像など，データの形状が $(D \times H \times W)$ となっている場合に用いられる．ここで，H は高さ，W を表す．3 次元畳み込み層は動画など，データの形状が $(D \times H \times W \times T)$ となっている場合に用いられる．

畳み込み層における主要なパラメータは 3 つある．1 つ目はカーネルサイズであり，これによって考慮できる入力特徴量の範囲が定まる．2 つ目はストライドであり，これによってカーネルのシフト幅を設定できる．3 つ目はダイレーションであり，これは畳み込み演算において計算対象となる入力特徴量の間隔を表す．ダイレーションを大きくすることで，カーネルサイズが同じでも考慮できる入力特徴量の範囲を広げることが可能である．また，出力系列長 T_{out} を入力系列長 T_{in} の整数倍に保つには，上記のパラメータに対して適切なパディング長を指定する必要がある．図 3.1に，ある入出力次元間における 1 次元畳み込み層の処理を示す．赤が入出力，青がカーネルを表している．図 3.1aは，カーネルサイズを 3，ストライドとダイレーションを 1 とした場合であり，入力の両端に 1 ずつゼロパディングすることで，入出力の系列長が変わらないことを表している．図 3.1bは，カーネルサイズを 5，ストライドとダイレーションを 1 とした場合であり，カーネルサイズが大きくなることで，出力 1 フレームを計算する際に考慮される入力の範囲が図 3.1aよりも広がることを表した．図 3.1cはカーネルサイズが 3，ストライドが 2，ダイレーションが 1 の場合であり，ストライドを 2 にしたことによって出力の系列長が入力よりも短くなることを表している．ここではパディングを 1 とすることで，入力に対して出力の系列長を 1/2 にできることを表した．図 3.1dはカーネルサイズが 3，ストライドが 1，ダイレーションが 2 の場合であり，ダイレーションを 2 とすることで，カーネルに対して入力が 1 フレーム飛ばしで計算されることを示した．

3.1.3 転置畳み込み層

転置畳み込み層は，畳み込み層の逆演算に対応する層であり，主に入力のアップサンプリングに使用される．図 3.2に，ある入出力次元間における 1 次元転置畳み込み層の処理を示す．赤が入出力，青がカーネルを表している．1 次元転置畳み込み層では，t 番目の入力とカーネルの積を計算し，その結果を t 番目から $t + K$ 番目までの出力とする．ここで K はカーネルサイズである．また，複数の入力から計算された出力がオーバーラップする場合，これらは加算される．図 3.2aは，カーネルサイズを 4，ストライドを 1 とした場合の様子である．アップサンプリングを行いたい場合は，ストライドを 2 以上とすれば良い．図 3.2bにカーネルサイズを 4，ストライドを 2 とした場合を示す．この時，入力系列長が 4 であるのに対して，出力系列長が 10 まで拡大されていることがわかる．ここで，出力系列長を入力系列長の整数倍に保つためには，出力の両端の削除数を適切に設定する必要がある．上述の例では両端の削除数を 1 とすることで，出力系列長を入力系列長の 2 倍である 8 に調整できる．



(a) $(K, S, R) = (3, 1, 1)$　(b) $(K, S, R) = (5, 1, 1)$

(c) $(K, S, R) = (3, 2, 1)$　(d) $(K, S, R) = (3, 1, 2)$

図 3.1: ある入出力次元間における 1 次元畳み込み層の処理．K はカーネルサイズ，S はストライド，R はダイレーションを表し，図中の 0 はパディング部を表す．

3.1.4　活性化関数

活性化関数は，ニューラルネットワークの出力に非線形性を与えるための関数である．これにより，DNN は単純な線形変換だけでは表現できない複雑な入出力の関係を学習可能になる．以下，活性化関数への入力を $x \in \mathbb{R}$ として，代表的なものを 6 つ述べる．また，本節で取り上げる活性化関数とその 1 階導関数のグラフを図 3.3に示す．

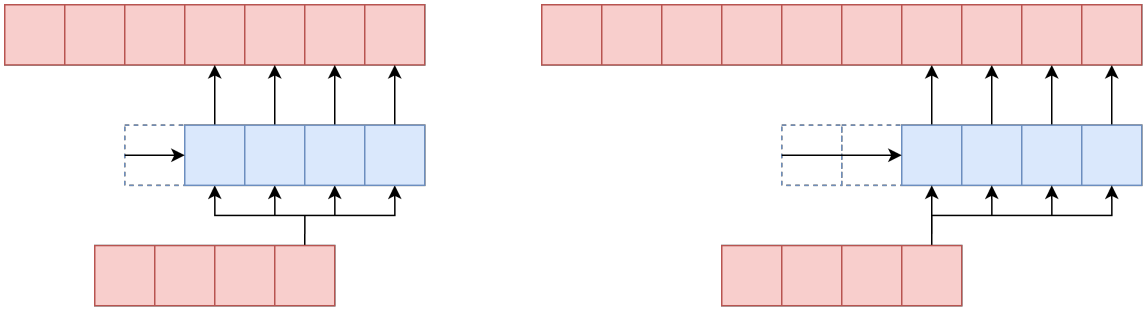
1 つ目は，シグモイド関数である．シグモイド関数は

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.4}$$

で与えられ，その 1 階導関数は

$$\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x} = \frac{\exp(-x)}{(1 + \exp(-x))^2} \tag{3.5}$$

となる．図 3.3bより，シグモイド関数の 1 階導関数の最大値は $x = 0$ における 0.25 であり，$|x|$ が大きくなるのに伴って，1 階導関数の値は小さくなることがわかる．DNN の各重みは，損失関数の勾配を利用することで更新されるから，シグモイド関数以前の層の重みにおける勾配は，シグモイド関数の 1 階導関数の値が乗算された結果となる．前述したように，シグモイド関数は 1 階導関数の値が小さくなりがちであるから，それ以前の層における勾配も小さくなり，重みの更新が進みづらくなる可能性がある．この問題を，勾配消失と呼ぶ．



(a) $(K, S) = (4, 1)$　　(b) $(K, S) = (4, 2)$

図 3.2: ある入出力次元間における 1 次元転置畳み込み層の処理．K はカーネルサイズ，S はストライドを表す．

2 つ目は，tanh 関数である．tanh は

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{3.6}$$

で与えられ，その 1 階導関数は

$$\frac{\mathrm{d}\tanh(x)}{\mathrm{d}x} = \frac{4}{(\exp(x) + \exp(-x))^2} \tag{3.7}$$

$$= \frac{1}{\cosh(x)^2} \tag{3.8}$$

となる．tanh の値域は $[-1, 1]$ となっており，図 3.3bより $|x|$ が小さいところではシグモイド関数より 1 階導関数の値が大きくなっていることがわかる．しかし，$|x|$ が大きくなればシグモイド関数と同様に 1 階導関数の値が小さく，勾配消失のリスクを抱えていることがわかる．

3 つ目は，ReLU である．ReLU は

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.9}$$

で与えられ，その 1 階導関数は

$$\frac{\mathrm{dReLU}(x)}{\mathrm{d}x} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x <= 0 \end{cases} \tag{3.10}$$

となる．ここで，ReLU は本来 $x = 0$ で微分不可能であるが，便宜上 $\mathrm{dReLU}(0)/\mathrm{d}x = 0$ とした．ReLU は入力が 0 以上であれば恒等写像として振る舞うが，0 未満であれば 0 に写す．1 階導関数は 0 あるいは 1 のみを取り，特に入力が正の値であれば常に 1 となることから，シグモイド関数や tanh よりも勾配消失が起こりづらい．ReLU は現在，標準的な活性化関数として広く用いられている．しかし，ReLU への入力が 0 未満の値を取るとき，ReLU 入力についての出力の勾配は 0 になるから，ReLU 以前の層の重みが更新されず，学習が遅くなる可能性がある．

3 つ目は，LeakyReLU[13] である．LeakyReLU は

$$\mathrm{LeakyReLU}\left(x\right) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x <= 0 \end{cases} \tag{3.11}$$

で与えられ，その 1 階導関数は

$$\frac{\mathrm{dLeakyReLU}\left(x\right)}{\mathrm{d}x} = \begin{cases} 1 & \text{if } x > 0 \\ a & \text{if } x <= 0 \end{cases} \tag{3.12}$$

となる．ここで，LeakyReLU は本来 $x = 0$ で微分不可能であるが，便宜上 $\mathrm{dLeakyReLU}\left(0\right)/\mathrm{d}x = a$ とした．ReLU と比較すると，0 未満の入力に対しても 0 でない値を出力し，1 階導関数も 0 にならない点が異なっている．これにより，重みの更新が進まなくなる ReLU の課題を解消した．

5 つ目は，PReLU[14] である．これは，LeakyReLU と似た活性化関数であるが，LeakyReLU のパラメータ a を学習可能にすることで，その他の層と合わせて最適化が可能となったことが特徴である．

6 つ目は，GELU[15] である．GELU は

$$\mathrm{GELU}\left(x\right) = x\Phi\left(x\right) \tag{3.13}$$

で与えられる．ここで，

$$\Phi\left(x\right) = P\left(X \leq x\right),\ X \sim \mathcal{N}\left(0, 1\right) \tag{3.14}$$

である．GELU の 1 階導関数は，

$$\frac{\mathrm{dGELU}\left(x\right)}{\mathrm{d}x} = \Phi\left(x\right) + \frac{x}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \tag{3.15}$$

となる．GELU は，ReLU が入力に対して 0 あるいは 1 を確定的にかける活性化関数と捉えた上で，これを入力に依存した確率的な挙動に変更したものである．

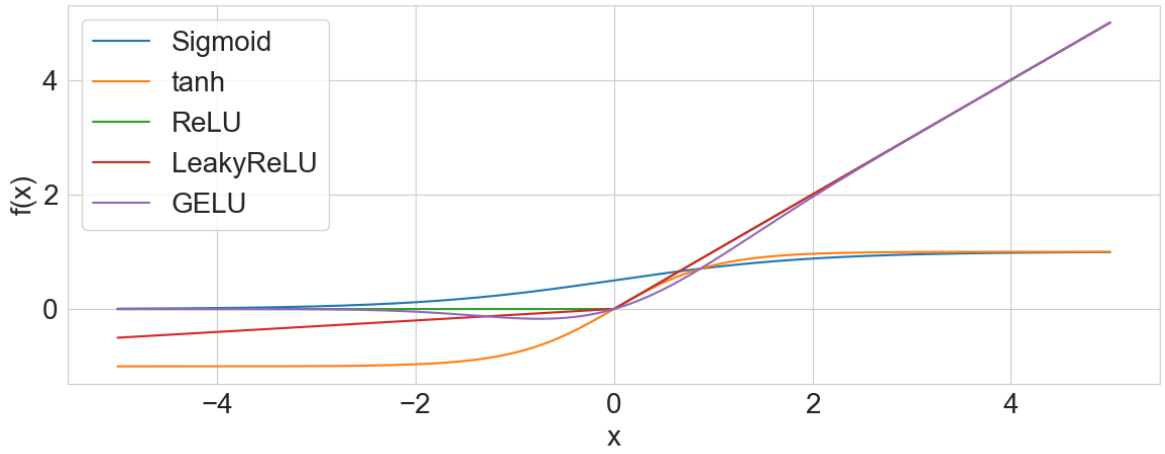
3.1.5 再帰型ニューラルネットワーク

再帰型ニューラルネットワーク（Recurrent Neural Network; RNN）は，自身の過去の出力を保持し，それをループさせる再帰的な構造を持ったネットワークである．

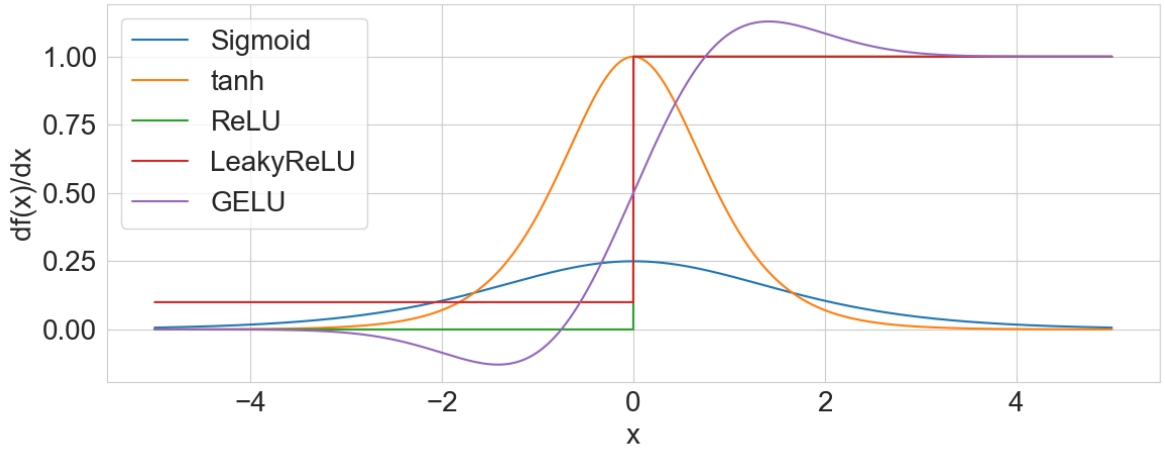
近年よく用いられる RNN として，長・短期記憶（Long Short-Time Memory; LSTM）[16] がある．LSTM は入力ゲート，忘却ゲート，出力ゲートの 3 つを持ち，これらゲートによってネットワーク内部の情報の取捨選択を行うことで，長い系列データからの学習を可能にした．LSTM のネットワーク内部で行われる計算を以下に示す．

$$\boldsymbol{f}_t = \sigma\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{f,x}\right) + \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{f,h}\right)\right) \tag{3.16}$$

$$\boldsymbol{i}_t = \sigma\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{i,x}\right) + \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{i,h}\right)\right) \tag{3.17}$$



(a) 活性化関数



(b) 活性化関数の 1 階導関数

図 3.3: 活性化関数の例

$$\tilde{\boldsymbol{c}}_t = \tanh\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{\tilde{c},x}\right) + \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{\tilde{c},h}\right)\right) \tag{3.18}$$

$$\boldsymbol{o}_t = \sigma\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{o,x}\right) + \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{o,h}\right)\right) \tag{3.19}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{3.20}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right) \tag{3.21}$$

ここで，$\boldsymbol{x}_t \in \mathbb{R}^{D_{\mathrm{in}}}$ は時刻 t の入力，$\boldsymbol{f}_t \in [0,1]^{D_{\mathrm{out}}}$ は忘却ゲートの出力，$\boldsymbol{i}_t \in [0,1]^{D_{\mathrm{out}}}$ は入力ゲートの出力，$\boldsymbol{c}_t \in [-1,1]^{D_{\mathrm{out}}}$ は時刻 t におけるセルの状態，$\boldsymbol{o}_t \in [0,1]^{D_{\mathrm{out}}}$ は出力ゲートの出力，$\boldsymbol{h}_t \in [-1,1]^{D_{\mathrm{out}}}$ は時刻 t における隠れ状態，$D_{\mathrm{in}}, D_{\mathrm{out}}$ は特徴量の次元を表す．また，$[\cdot, \ldots, \cdot]$ は入力された特徴量の次元方向の結合，$\odot$ は要素積を表す．忘却ゲート出力 $\boldsymbol{f}_t$ が前時刻のセル状態 $\boldsymbol{c}_{t-1}$ に含まれる情報の選択，入力ゲート出力 $\boldsymbol{i}_t$ が新たな入力 $\tilde{\boldsymbol{c}}_t$ に含まれる情報の選択に用いられ，$\boldsymbol{c}_t$ が決まる．その後，出力ゲート出力 $\boldsymbol{o}_t$ が $\boldsymbol{c}_t$ に含まれる情報の選択に用いられ，$\boldsymbol{h}_t$ が決まる．

また，LSTMが3つのゲートを必要とするのに対し，ゲートを2つに減らすことでネットワークの軽量化を図ったのがゲート付き回帰型ユニット（Gated Recurrent Unit; GRU）[17] である．GRUはリセットゲートと更新ゲートの2つを用いて隠れ状態を更新する．GRUのネットワーク内部で行われる計算を以下に示す．

$$\boldsymbol{z}_t = \sigma\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{z,x}\right) + \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{z,h}\right)\right) \tag{3.22}$$

$$\boldsymbol{r}_t = \sigma\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{r,x}\right) + \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{r,h}\right)\right) \tag{3.23}$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\mathrm{FC}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_{\tilde{h},x}\right) + \boldsymbol{r}_t \odot \mathrm{FC}\left(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}_{\tilde{h},h}\right)\right) \tag{3.24}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \tag{3.25}$$

ここで，$\boldsymbol{x}_t \in \mathbb{R}^{D_{\mathrm{in}}}$ が時刻 t における入力，$\boldsymbol{z}_t \in [0,1]^{D_{\mathrm{out}}}$ が更新ゲートの出力，$\boldsymbol{r}_t \in [0,1]^{D_{\mathrm{out}}}$ がリセットゲートの出力，$\boldsymbol{h}_t \in [-1,1]^{D_{\mathrm{out}}}$ が時刻 t における隠れ状態を表す．更新ゲート出力 $\boldsymbol{z}_t$ が $\boldsymbol{h}_{t-1}$ と $\tilde{\boldsymbol{h}}_t$ に含まれる情報の選択，リセットゲート出力 $\boldsymbol{r}_t$ が $\boldsymbol{h}_{t-1}$ に含まれる情報の選択に用いられる．

3.1.6 正規化層

DNNの学習過程では学習の進行に伴って重みが変化するため，その度に各層への入力の分布が変わってしまう．これは内部共変量シフトと呼ばれ，ネットワークの学習を不安定にする原因となる．これに対し，バッチ正規化（Batch Normalization）[18] が有効である．バッチ正規化は，ミニバッチ内における入力特徴量の期待値と分散を次元ごとに計算し，これらを用いて入力特徴量を次元ごとに標準化するものである．ここで，バッチサイズを N，バッチ正規化への D 次元の入力特徴量を $\boldsymbol{x}_n \in \mathbb{R}^D \ (n = 1, \ldots, N)$，出力特徴量を $\boldsymbol{y}_n \in \mathbb{R}^D \ (n = 1, \ldots, N)$ とする．このとき，各 n に対し入力特徴量 $\boldsymbol{x}_n$ の d 次元目の成分を $x_{n,d}$，出力特徴量 $\boldsymbol{y}_n$ の d 次元目の成分を $y_{n,d}$ とすると，$y_{n,d}$ は

$$\mu_d^B = \frac{1}{N} \sum_{n=1}^{N} x_{n,d} \tag{3.26}$$

$$\left(\sigma_d^B\right)^2 = \frac{1}{N} \sum_{n=1}^{N} \left(x_{n,d} - \mu_d^B\right)^2 \tag{3.27}$$

$$\tilde{x}_{n,d} = \frac{x_{n,d} - \mu_d^B}{\sqrt{\left(\sigma_d^B\right)^2 + \epsilon}} \tag{3.28}$$

$$y_{n,d} = \gamma_d \tilde{x}_{n,d} + \beta_d \tag{3.29}$$

で与えられる．ここで，γ_d, β_d は学習可能なスカラーであり，ϵ はゼロ割を避けるためのスカラーである．バッチ正規化では γ_d, β_d によって表現力を向上させており，実際

$$\gamma_d = \sqrt{\left(\sigma_d^B\right)^2 + \epsilon} \tag{3.30}$$

$$\beta_d = \mu_d^B \tag{3.31}$$

とすれば，標準化前の入力を再び得ることが可能である．学習時は，サンプルの標準化に用いる統計量とは別に，期待値の移動平均と不偏分散の移動平均を計算しておく．推論時は学習終了時に得られたこれら移動平均の値を用いるため，入力サンプルによらない挙動になる．

バッチ正規化は DNN の学習の安定化に貢献する一方，ミニバッチ全体における統計量を利用するため，バッチサイズが小さい場合はデータの分布を安定させることが難しくなる．また，テキストや音声といった系列長を持つデータを扱う場合，ミニバッチを構成するためにはゼロパディングによって系列長を揃える必要がある．この時，RNN の各ステップの出力に対しバッチ正規化を適用すると，ゼロパディングによって人為的に系列量を揃えているから，統計量が実際のデータの分布からかけ離れたものになる可能性がある．これら課題に対し，ミニバッチ内の各サンプルごとに期待値と分散を求めて標準化する，レイヤー正規化（Layer Normalization）[19] がある．バッチ正規化のときと同様の表記を用いると，$y_{n,d}$ は

$$\mu_n^L = \frac{1}{D}\sum_{d=1}^{D} x_{n,d} \tag{3.32}$$

$$\left(\sigma_n^L\right)^2 = \frac{1}{D}\sum_{d=1}^{D}\left(x_{n,d} - \mu_n^L\right)^2 \tag{3.33}$$

$$\tilde{x}_{n,d} = \frac{x_{n,d} - \mu_n^L}{\sqrt{\left(\sigma_n^L\right)^2 + \epsilon}} \tag{3.34}$$

$$y_{n,d} = \gamma_d \tilde{x}_{n,d} + \beta_d \tag{3.35}$$

で与えられる．

上述したバッチ正規化およびレイヤー正規化は，特徴量を標準化することで学習を安定させる手法であった．一方，DNN 内のある層の重みを再パラメータ化することで学習を安定させる手法として，重み正規化（Weight Normalization）[20] がある．これは，ある層の重みベクトル $\boldsymbol{w}$ を，

$$\boldsymbol{w} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} g \tag{3.36}$$

のように単位ベクトル $\boldsymbol{v}/\|\boldsymbol{v}\|_2$（ベクトルの向き）とスカラー g（ベクトルの長さ）に再パラメータ化するものである．学習時は重みの更新を $\boldsymbol{v}$ と g で別々に行う．重み正規化は，バッチ正規化やレイヤー正規化と同様に学習の安定化に役立つが，計算に入力特徴量の系列長が依存しない．そのため，例えば音声波形など系列長が非常に長くなりがちなデータを扱う場合，計算コストを下げながら同様の効果を狙える手段だと考えられる．

3.1.7 Transformer

Transformer[4] は，自己注意機構（Self-Attention）を用いて，入力系列全体に渡る依存関係を捉えることができるニューラルネットワークである．特に，再帰的な計算を必要とする RNN

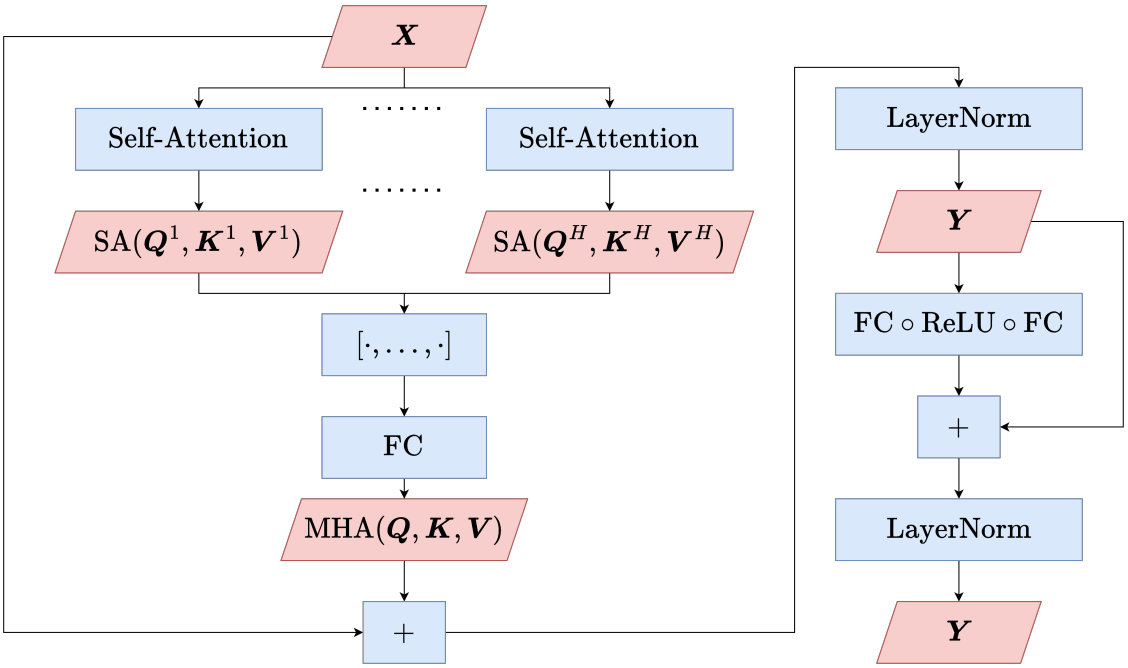


図 3.4: Transformer 層の構造（赤い平行四辺形：入出力，青い長方形：処理）

と比較して，Transformer は並列計算のみ行うため，GPU による計算の高速化が可能である．以下，入力特徴量を $\boldsymbol{X} \in \mathbb{R}^{T \times D_{\text{model}}}$ として，Transformer において行われる計算を説明する．T は系列長，D_{model} は特徴量の次元を表す．また，Transformer 層の構造を図 3.4に示す．赤の平行四辺形が入出力，青の長方形が処理を表す．

まず，Transformer 層における Self-Attention の計算の流れを述べる．ここでは，はじめにクエリ $\boldsymbol{Q} \in \mathbb{R}^{T \times D_k}$，キー $\boldsymbol{K} \in \mathbb{R}^{T \times D_k}$，バリュー $\boldsymbol{V} \in \mathbb{R}^{T \times D_v}$ の計算を行う．これは，

$$\boldsymbol{Q} = \mathrm{FC}\left(\boldsymbol{X}; \boldsymbol{\theta}_Q\right) \tag{3.37}$$

$$\boldsymbol{K} = \mathrm{FC}\left(\boldsymbol{X}; \boldsymbol{\theta}_K\right) \tag{3.38}$$

$$\boldsymbol{V} = \mathrm{FC}\left(\boldsymbol{X}; \boldsymbol{\theta}_V\right) \tag{3.39}$$

で与えられる．次に，クエリとキーを元にアテンション重みを求め，バリューに対する行列積を計算する．これは，

$$\mathrm{SA}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D_k}}\right)\boldsymbol{V} \tag{3.40}$$

で与えられる．softmax 関数は行方向に適用されるため，$\mathrm{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{D_k}\right)$ の各行ベクトルが，各クエリ $\boldsymbol{q}_t \in \mathbb{R}^{D_k}$ からキー $\boldsymbol{k}_t \in \mathbb{R}^{D_k}$ $(t = 1, \ldots, T)$ に対する注意度になっている．

ここまでが Self-Attention の計算であったが，Transformer 層では Self-Attention を複数のヘッドで並列に計算し，各ヘッドの出力を結合して最終出力を得る Multi-Head Attention が採用されている．これは，ヘッド数を H とし，各ヘッド h におけるクエリ $\boldsymbol{Q}^h \in \mathbb{R}^{T \times \frac{D_k}{H}}$，キー $\boldsymbol{K}^h \in \mathbb{R}^{T \times \frac{D_k}{H}}$，バリュー $\boldsymbol{V}^h \in \mathbb{R}^{T \times \frac{D_v}{H}}$ によって計算された $\mathrm{SA}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right)$ を用いて，

$$\mathrm{MHA}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \mathrm{FC}\left(\left[\mathrm{SA}\left(\boldsymbol{Q}^1, \boldsymbol{K}^1, \boldsymbol{V}^1\right), \ldots, \mathrm{SA}\left(\boldsymbol{Q}^H, \boldsymbol{K}^H, \boldsymbol{V}^H\right)\right]; \boldsymbol{\theta}_{\text{MHA}}\right) \tag{3.41}$$

で与えられる．Multi-Head Attention 後は，残差結合とレイヤー正規化を適用する．すなわち，この出力 $\boldsymbol{Y} \in \mathbb{R}^{T \times D_{\text{model}}}$ は，

$$\boldsymbol{Y} = \text{LayerNorm}\left(\text{MHA}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) + \boldsymbol{X}\right) \tag{3.42}$$

で与えられる．その後，全結合層を通し，再度残差結合とレイヤー正規化を適用することで Transformer 層最終出力を得る．すなわち，

$$\boldsymbol{Y} = \text{LayerNorm}\left(\text{FC}\left(\text{ReLU}\left(\text{FC}\left(\boldsymbol{Y}; \boldsymbol{\theta}_1\right)\right); \boldsymbol{\theta}_2\right) + \boldsymbol{Y}\right) \tag{3.43}$$

となる．Transformer は，Transformer 層を多層積み重ねて構成される．

最後に，Transformer では RNN と違い，並列計算によって系列全体を 1 度に処理することが可能であるが，それと引き換えに入力の順序情報を考慮することができなくなる．これに対し，Transformer では Positional Encoding によって入力に位置情報を与える．Positional Encoding は sin と cos に基づいて，

$$\text{PositionalEncoding}\left(t, d\right) = \begin{cases} \sin\left(\frac{t}{10000^{2d/D_{\text{model}}}}\right) & \text{if } d \bmod 2 = 0 \\ \cos\left(\frac{t}{10000^{2d/D_{\text{model}}}}\right) & \text{if } d \bmod 2 = 1 \end{cases} \tag{3.44}$$

で与えられる．

3.2 学習方法

本節において，$\boldsymbol{\theta}$ は DNN の全ての重みとバイアスをまとめて表す行ベクトルとする．また，簡潔さを優先して，特別な理由がない限りは重みと呼ぶ．

3.2.1 損失関数

損失関数は，DNN によって予測された結果と正解値との間の誤差を求める関数のことであり，扱う問題によって様々である．例えば，回帰問題において用いられる関数の 1 つに，MAE（Mean Absolute Error）Loss がある．D 次元の予測対象を $\boldsymbol{y} \in \mathbb{R}^D$，DNN による予測結果を $\hat{\boldsymbol{y}} \in \mathbb{R}^D$ とすると，MAE Loss は

$$L_{\text{MAE}}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) = \frac{1}{D} \sum_{d=1}^{D} \left| y_d - \hat{y}_d \right| \tag{3.45}$$

で与えられる．また，予測対象が系列長 T を持った行列 $\boldsymbol{Y} \in \mathbb{R}^{T \times D}$ の場合，DNN による予測結果を $\hat{\boldsymbol{Y}} \in \mathbb{R}^{T \times D}$ とすると，MAE Loss は

$$L_{\text{MAE}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = \frac{1}{TD} \sum_{t=1}^{T} \sum_{d=1}^{D} \left| y_{t,d} - \hat{y}_{t,d} \right| \tag{3.46}$$

で与えられる．

一方，分類問題において用いられる関数の 1 つに，Cross Entropy Loss がある．C クラス分類の問題について，予測対象を $\boldsymbol{y} \in [0,1]^C$，DNN による予測値を $\hat{\boldsymbol{y}} \in \mathbb{R}^C$ とすると，Cross Entropy Loss は

$$L_{\mathrm{CE}}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) = -\sum_{c=1}^{C} y_c \log\left(\frac{\exp\left(\hat{y}_c\right)}{\sum_{i=1}^{C}\exp\left(\hat{y}_c\right)}\right) \tag{3.47}$$

で与えられる．ここで，$\boldsymbol{y}$ はクラスに対する確率分布であり，

$$\sum_{c=1}^{C} y_c = 1 \tag{3.48}$$

を満たす．実際は，正解となるクラスのみを 1，それ以外を 0 とした One-hot ベクトルとされることが多い．また，予測値 $\hat{\boldsymbol{y}}$ は各クラスに対するスコアを表す値であり，ロジットと呼ばれる．予測対象が行列 $\boldsymbol{Y} \in [0,1]^{T\times C}$ の場合，ロジットを $\hat{\boldsymbol{Y}} \in \mathbb{R}^{T\times D}$ とすると，Cross Entropy Loss は

$$L_{\mathrm{CE}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C} y_{t,c} \log\left(\frac{\exp\left(\hat{y}_{t,c}\right)}{\sum_{i=1}^{C}\exp\left(\hat{y}_{t,i}\right)}\right) \tag{3.49}$$

で与えられる．

3.2.2 勾配降下法

勾配降下法は，損失関数の重みについての勾配を利用して，損失関数の値を最小化するように DNN を最適化するアルゴリズムである．ここで，学習データセットを $\mathcal{D}_{\mathrm{train}} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$ とする．各 n に対し，$\boldsymbol{x}_n \in \mathbb{R}^{D_{\mathrm{in}}}$ は DNN への入力，$\boldsymbol{y}_n \in \mathbb{R}^{D_{\mathrm{out}}}$ は予測対象を表す．$D_{\mathrm{in}}, D_{\mathrm{out}}$ は入出力の次元である．DNN は f とし，予測値を $\hat{\boldsymbol{y}}_n = f\left(\boldsymbol{x}_n; \boldsymbol{\theta}\right)$ とする．損失関数は L とし，$\mathcal{L}$ を

$$\mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} L\left(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i\right) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right) \tag{3.50}$$

で定義する．ここで，$\mathcal{I} \subset \{1, \ldots, N\}$ は各サンプルに対するインデックスの部分集合，$|\cdot|$ は集合の濃度を表す．$\mathcal{L}$ は学習データ $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i\in\mathcal{I}} \subset \mathcal{D}_{\mathrm{train}}$ に対する損失を，DNN の重み $\boldsymbol{\theta}$ の関数として扱うために導入した．上の表記を用いると，DNN の最適化問題は

$$\min_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \{1, \ldots, N\}\right) \tag{3.51}$$

と表される．この最適化問題に対し，勾配降下法による重み $\boldsymbol{\theta}$ の更新は，

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta\nabla\mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \mathcal{I}_\tau\right) \tag{3.52}$$

で与えられる．ここで，τ は学習におけるイテレーション，η は学習率を表す．

勾配降下法には，3 種類の方法がある [21]．1 つ目は，バッチ勾配降下法である．これは，

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta \nabla \mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \{1, \ldots, N\}\right) \tag{3.53}$$

で与えられる．すなわち，各イテレーションで学習データ全てを用いる方法である．各サンプルのノイズの影響が低減されることで安定した学習が期待できるが，計算コストが高い．2 つ目は，確率的勾配降下法である．これは，ランダムに選択された $n_\tau \in \{1, \ldots, N\}$ に対し，

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta \nabla \mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \{n_\tau\}\right) \tag{3.54}$$

で与えられる．すなわち，各イテレーションで単一サンプルのみを用いる方法である．計算コストが下がるが，各サンプルのノイズの影響が大きくなることで学習が不安定になる可能性がある．3 つ目は，ミニバッチ勾配降下法である．これは，$1 < |\mathcal{I}_\tau| < N$ を満たすランダムに選択された $\mathcal{I}_\tau \subsetneq \{1, \ldots, N\}$ に対し，

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta \nabla \mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \mathcal{I}_\tau\right) \tag{3.55}$$

で与えられる．バッチ勾配降下法と確率的勾配降下法の間をとった方法であり，DNN の学習においては一般にミニバッチ勾配降下法が用いられる．ここで，ミニバッチに含まれるサンプルの数をバッチサイズと呼ぶ.

また，確率的勾配降下法やミニバッチ勾配降下法では，サンプルを学習データセットからランダムに非復元抽出する．ここで，毎回のサンプリングされた学習データに対する処理は 1 イテレーションとカウントし，学習データセットを 1 度全て使い切ることは 1 エポックとカウントする．実際には，データセットの総サンプル数 N に対してバッチサイズを決定することで 1 エポックあたりの総イテレーション数は決まり，最大エポック数を設定して学習を回すこととなる．

3.2.3 正則化

DNN は大量のパラメータにより高い表現力を持つが，その分学習データに過剰に適合し，未知データに対する汎化性能が低いモデルとなる，過学習を引き起こす可能性がある．正則化は，このような DNN の過学習を防ぐための手段である．以下，具体的な方法を 4 つ述べる．

1 つ目は，L2 正則化である．これは，3.2.2節で定義した $\mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right)$ を，

$$\mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right) + \frac{\lambda}{2} \left\|\boldsymbol{\theta}\right\|_2^2 \tag{3.56}$$

とすることで与えられる．ここで，λ は正則化の程度を調整するパラメータである．これより，L2 正則化は損失関数の値に $\left\|\boldsymbol{\theta}\right\|_2^2$ を加算することで，重み $\boldsymbol{\theta}$ の L2 ノルムが過大になることを防ぐ方法だと言える．

2つ目は，Weight Decayである．これは，重みの更新式を

$$\boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta \nabla \mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \mathcal{I}_\tau\right) - \lambda' \boldsymbol{\theta}_{\tau-1} \tag{3.57}$$

とすることで与えられる．ここで，λ' は正則化の程度を調整するパラメータである．これより，Weight Decayは重み $\boldsymbol{\theta}$ の絶対値が過大になることを防ぐ手法だと言える．

3つ目は，Dropout[22]である．Dropoutは，学習時に特徴量の一部を0に落とす手法である．一方，推論時は恒等写像となり，学習時と挙動が変わる．Dropoutへの入力を $\boldsymbol{x} \in \mathbb{R}^D$，特徴量を0に落とす確率を p とすると，学習時のDropout出力 $\boldsymbol{y}_{\text{train}} \in \mathbb{R}^D$ および推論時のDropout出力 $\boldsymbol{y}_{\text{infer}} \in \mathbb{R}^D$ は，

$$\boldsymbol{y}_{\text{train}} = \frac{\boldsymbol{x} \odot \boldsymbol{m}}{1-p} \tag{3.58}$$

$$\boldsymbol{y}_{\text{infer}} = \boldsymbol{x} \tag{3.59}$$

で与えられる．ここで，D は特徴量の次元，$\boldsymbol{m} \in \{0,1\}^D$ は各成分 m_d が確率 $1-p$ で1，確率 p で0をとる確率変数である．式(3.58)より，学習時はDropout出力を $1/(1-p)$ 倍していることがわかる．この理由は，学習時の出力の期待値と推論時の出力を一致させるためである．実際，確率変数 $\boldsymbol{m}$ の従う確率分布上で $\boldsymbol{y}_{\text{train}}$ の期待値をとれば，

$$\mathbb{E}\left[\boldsymbol{y}_{\text{train}}\right] = \mathbb{E}\left[\frac{\boldsymbol{x} \odot \boldsymbol{m}}{1-p}\right] \tag{3.60}$$

$$= \frac{\boldsymbol{x}}{1-p} \odot \mathbb{E}\left[\boldsymbol{m}\right] \tag{3.61}$$

$$= \frac{\boldsymbol{x}}{1-p} \odot (1-p \ \cdots \ 1-p)^\top \tag{3.62}$$

$$= \boldsymbol{x} = \boldsymbol{y}_{\text{infer}} \tag{3.63}$$

となる．仮に学習時の挙動のまま推論を行えば，同じ入力を与えても出力は変化するため，複数パターンの出力の期待値を最終出力とすることで，アンサンブルモデルのように汎化性能の向上が期待される．しかし，これは計算コストの増大につながる．Dropoutではこの課題に対して，学習時の出力を $1/(1-p)$ 倍することで，推論時に1度の計算でアンサンブルモデルの効果を得られるよう設計されている．これにより，汎化性能を向上させつつ，計算コストが増加しないようになっている．

4つ目は，Early Stoppingである．Early Stoppingは，検証データに対する損失の増加を監視し，設定したエポック数だけ増加し続けた場合に学習を停止する手法である．これにより，学習データに対する過度なフィッティングを防止する．

3.2.4 最適化手法

3.2.2節において，DNNの重みが勾配降下法によって最適化されることを述べた．ここで，通常の勾配降下法に代わり，近年よく用いられる最適化手法としてAdam[23]がある．Adamの

計算過程をアルゴリズム 1に示す．ここで，$\boldsymbol{g}_\tau$ は勾配の 1 次モーメント，$\boldsymbol{m}_\tau$ が勾配の 1 次モーメントの指数移動平均，$\boldsymbol{v}_\tau$ が勾配の 2 次モーメントの指数移動平均である．$\hat{\boldsymbol{m}}_\tau$ および $\hat{\boldsymbol{v}}_\tau$ はそれぞれ，$\boldsymbol{m}_\tau$ および $\boldsymbol{v}_\tau$ の初期値が 0 であることに起因するバイアスを防ぐための計算を行った結果である．β_1, β_2 は，指数移動平均の程度を調整するパラメータである．また，9 行目のベクトルの平方根および，ベクトルとベクトルの割り算は，各成分ごとに行われる．Adam では，$\hat{\boldsymbol{m}}_\tau$ を勾配の代わりに用いることで，ミニバッチごとのノイズに対する頑健な学習を可能にする．また，$\hat{\boldsymbol{v}}_\tau$ の平方根で $\hat{\boldsymbol{m}}_\tau$ を割ることによって，勾配の大きさに対するスケーリングを行っている．ここで，Adam では正則化として L2 正則化を採用したが，Weight Decay を採用した

Algorithm 1 Adam

1: **Input:** η, β_1, β_2, λ, $\boldsymbol{\theta}_0$, $L(\boldsymbol{\theta})$
2: $\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$
3: **for** $\tau = 1$ to $\ldots$ **do**
4: $\quad \boldsymbol{g}_\tau = \nabla \mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \mathcal{I}_\tau\right) + \lambda \boldsymbol{\theta}_{\tau-1}$
5: $\quad \boldsymbol{m}_\tau = \beta_1 \boldsymbol{m}_{\tau-1} + (1-\beta_1)\, \boldsymbol{g}_\tau$
6: $\quad \boldsymbol{v}_\tau = \beta_2 \boldsymbol{v}_{\tau-1} + (1-\beta_2)\, \boldsymbol{g}_\tau \odot \boldsymbol{g}_\tau$
7: $\quad \tilde{\boldsymbol{m}}_\tau = \boldsymbol{m}_\tau / \left(1-\beta_1^\tau\right)$
8: $\quad \tilde{\boldsymbol{v}}_\tau = \boldsymbol{v}_\tau / \left(1-\beta_2^\tau\right)$
9: $\quad \boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta \tilde{\boldsymbol{m}}_\tau / \left(\sqrt{\tilde{\boldsymbol{v}}_\tau} + \epsilon\right)$
10: **end for**
11: **Return** $\boldsymbol{\theta}_\tau$

最適化手法として AdamW[24] がある．AdamW の計算過程をアルゴリズム 2に示す．正則化が L2 正則化から Weight Decay に変わった点以外は同じである．

Algorithm 2 AdamW

1: **Input:** η, β_1, β_2, λ, $\boldsymbol{\theta}_0$, $L(\boldsymbol{\theta})$
2: $\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$
3: **for** $\tau = 1$ to $\ldots$ **do**
4: $\quad \boldsymbol{g}_\tau = \nabla \mathcal{L}\left(\boldsymbol{\theta}_{\tau-1}; \mathcal{I}_\tau\right)$
5: $\quad \boldsymbol{m}_\tau = \beta_1 \boldsymbol{m}_{\tau-1} + (1-\beta_1)\, \boldsymbol{g}_\tau$
6: $\quad \boldsymbol{v}_\tau = \beta_2 \boldsymbol{v}_{\tau-1} + (1-\beta_2)\, \boldsymbol{g}_\tau \odot \boldsymbol{g}_\tau$
7: $\quad \tilde{\boldsymbol{m}}_\tau = \boldsymbol{m}_\tau / \left(1-\beta_1^\tau\right)$
8: $\quad \tilde{\boldsymbol{v}}_\tau = \boldsymbol{v}_\tau / \left(1-\beta_2^\tau\right)$
9: $\quad \boldsymbol{\theta}_\tau = \boldsymbol{\theta}_{\tau-1} - \eta \tilde{\boldsymbol{m}}_\tau / \left(\sqrt{\tilde{\boldsymbol{v}}_\tau} + \epsilon\right) - \eta \lambda \boldsymbol{\theta}$
10: **end for**
11: **Return** $\boldsymbol{\theta}_\tau$

3.2.5　学習率のスケジューリング

学習率のスケジューリングは，学習率 η の値自体を学習の進行に伴って変更するものである．これは，より安定した学習を促したり，より早く学習を収束させたりするのに役立つ手段である．以下，3 つのスケジューラを例として述べる．また，各スケジューラを用いた場合における学習率の遷移を図 3.5に示す．

1 つ目は，StepLRScheduler である．これは，初期学習率を η_0 として，エポック ν における学習率 η_ν を

$$\eta_\nu = \eta_0 \gamma^{\lfloor \nu / \nu_{\text{step}} \rfloor} \tag{3.64}$$

で与えるスケジューラである．これは，学習が ν_{step} エポック進むごとに学習率を γ 倍することで，学習率を段階的に変化させる．シンプルで分かりやすいが，学習率の変化が不連続的になる．

2 つ目は，ExponentialLRScheduler である．これは，η_ν を

$$\eta_\nu = \eta_0 \exp\left(-\gamma \nu\right) \tag{3.65}$$

で与えるスケジューラである．学習が 1 エポック進むごとに学習率を指数関数的に変化させるため，StepLRScheduler と比較して変化が連続的である．

3 つ目は，Cosine Annealing with Warmup である．これは，η_ν を

$$\eta_\nu = \begin{cases} \eta_{\min} + \left(\frac{\nu}{\nu_{\text{warmup}}}\right)\left(\eta_{\max} - \eta_{\min}\right) & \text{if } \nu < \nu_{\text{warmup}} \\ \eta_{\min} + \frac{1}{2}\left(\eta_{\max} - \eta_{\min}\right)\left(1 + \cos\left(\frac{(\nu - \nu_{\text{warmup}})\pi}{\nu_{\max} - \nu_{\text{warmup}}}\right)\right) & \text{if } \nu \geq \nu_{\text{warmup}} \end{cases} \tag{3.66}$$

で与えるスケジューラである．ここで，$\eta_{\min}$ は最小学習率，$\eta_{\max}$ は最大学習率，$\nu_{\max}$ は最大エポックである．ν_{warmup} は学習率を $\eta_{\min}$ から $\eta_{\max}$ まで線形に増加させるのにかけるエポック数を指定するパラメータである．エポック数が ν_{warmup} 以上となれば，cos 関数に従って学習率を減衰させる．Cosine Annealing with Warmup は不安定になりがちな学習初期に学習率が低い状態から開始して，徐々に学習率を大きくすることで解の十分な探索を可能にし，その後再び学習率を小さくすることで学習の収束を促すスケジューラである．

3.2.6　誤差逆伝播法

誤差逆伝播法は，DNN の各重みについての損失関数の勾配を，出力から入力へと遡る方向に計算するアルゴリズムである．ここでは例として，全結合層と活性化関数のみからなる N_{layer} 層の DNN を構築し，ミニバッチ勾配降下法によって最適化する場面を考える [25]．また，3.2.2節で定義した表記を再度用いる．

まず，$w^n_{p,q}$ を $n-1$ 層目の p 番目のニューロンから n 層目の q 番目のニューロンに割り当てられた重み，b^n_p を n 層目の p 番目のニューロンに割り当てられたバイアスとすると，n 層目の p

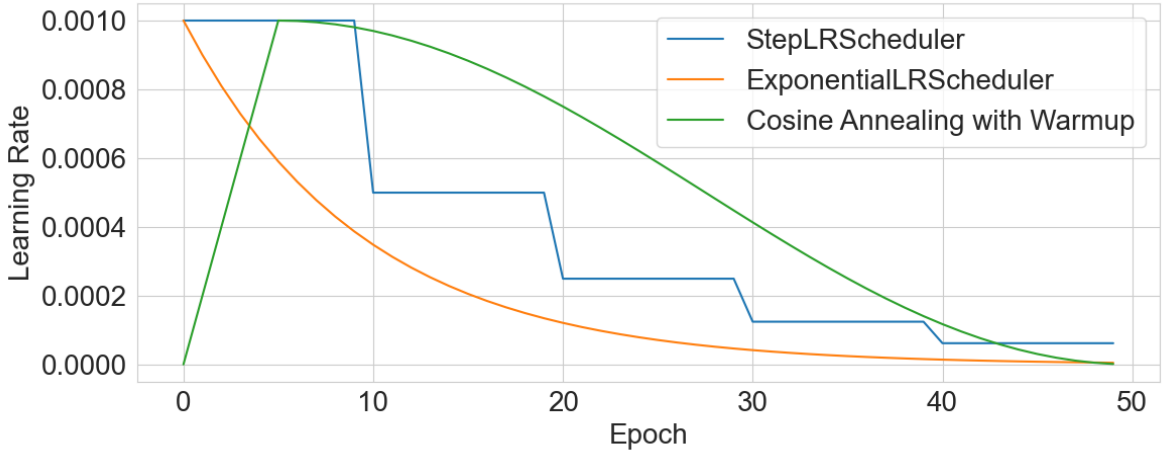


図 3.5: スケジューラによる学習率の変化

番目のニューロンにおける出力 a_p^n は，

$$a_p^n = b_p^n + \sum_{q=1}^{D_{n-1}} w_{q,p}^n o_q^{n-1} \tag{3.67}$$

で与えられる．ここで，D_{n-1} は $n-1$ 層目のニューロン数，o_q^{n-1} は $n-1$ 層目の q 番目のニューロンにおける出力 a_q^{n-1} に活性化関数 ϕ を適用した結果を表す．すなわち，

$$o_q^{n-1} = \phi\left(a_q^{n-1}\right) \tag{3.68}$$

である．例外として，各 $i \in \mathcal{I}_\tau$ に対し，$\boldsymbol{o}^0 = \boldsymbol{x}_i$ とする．また，$D_0 = D_{\text{in}}, D_{N_{\text{layer}}} = D_{\text{out}}$ とする．ここで，式 (3.67) に対し，$w_{0,p}^n = b_p^n$, $o_0^{n-1} = 1$ とおけば，

$$a_p^n = b_p^n + \sum_{q=1}^{D_{n-1}} w_{q,p}^n o_q^{n-1} = \sum_{q=0}^{D_{n-1}} w_{q,p}^n o_q^{n-1} \tag{3.69}$$

と整理できる．この時，

$$\frac{\partial \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}_\tau\right)}{\partial w_{p,q}^n} = \frac{\partial}{\partial w_{p,q}^n}\left(\frac{1}{|\mathcal{I}_\tau|}\sum_{i\in\mathcal{I}_\tau} L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)\right) \tag{3.70}$$

$$= \frac{1}{|\mathcal{I}_\tau|}\sum_{i\in\mathcal{I}_\tau} \frac{\partial L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)}{\partial w_{p,q}^n} \tag{3.71}$$

$$= \frac{1}{|\mathcal{I}_\tau|}\sum_{i\in\mathcal{I}_\tau} \frac{\partial L_i}{\partial w_{p,q}^n} \tag{3.72}$$

となる．ここで，$L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right) = L_i$ とおいた．この時，各 $n \in \{1, \ldots, N_{\text{layer}}\}$ に対し，$\partial L_i / \partial w_{p,q}^n$ は式 (3.69) を用いて，

$$\frac{\partial L_i}{\partial w_{p,q}^n} = \frac{\partial L_i}{\partial a_q^n}\frac{\partial a_q^n}{\partial w_{p,q}^n} \tag{3.73}$$

$$= \delta_q^n \frac{\partial}{\partial w_{p,q}^n} \left(\sum_{r=0}^{D_{n-1}} w_{r,q}^n o_r^{n-1} \right) \tag{3.74}$$

$$= \delta_q^n o_p^{n-1} \tag{3.75}$$

となる．ここで，$\partial L_i / \partial a_q^n = \delta_q^n$ とおいた．このとき，最終層（$n = N_{\text{layer}}$）の重みの場合，$f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right) = \boldsymbol{o}^{N_{\text{layer}}}$ だから，

$$\frac{\partial L_i}{\partial w_{p,q}^{N_{\text{layer}}}} = \delta_q^{N_{\text{layer}}} o_p^{N_{\text{layer}}-1} \tag{3.76}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L_i}{\partial a_q^{N_{\text{layer}}}} \tag{3.77}$$

$$= o_p^{N_{\text{layer}}-1} \sum_{r=1}^{D_{N_{\text{layer}}}} \frac{\partial L_i}{\partial o_r^{N_{\text{layer}}}} \frac{\partial o_r^{N_{\text{layer}}}}{\partial a_q^{N_{\text{layer}}}} \tag{3.78}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L_i}{\partial o_q^{N_{\text{layer}}}} \frac{\partial o_q^{N_{\text{layer}}}}{\partial a_q^{N_{\text{layer}}}} \tag{3.79}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L_i}{\partial o_q^{N_{\text{layer}}}} \phi' \left(a_q^{N_{\text{layer}}} \right) \tag{3.80}$$

となる．これは，入力から出力を計算する順伝搬で得られた値のみに依存するから，直ちに計算可能であることがわかる．一方，最終層以外（$1 \leq n < N_{\text{layer}}$）の重みの場合，

$$\frac{\partial L_i}{\partial w_{p,q}^n} = \delta_q^n o_p^{n-1} \tag{3.81}$$

$$= o_p^{n-1} \frac{\partial L_i}{\partial a_q^n} \tag{3.82}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \frac{\partial L_i}{\partial a_r^{n+1}} \frac{\partial a_r^{n+1}}{\partial a_q^n} \tag{3.83}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} \frac{\partial}{\partial a_q^n} \left(\sum_{s=0}^{D_n} w_{s,r}^{n+1} o_s^n \right) \tag{3.84}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} \frac{\partial}{\partial a_q^n} \left(\sum_{s=0}^{D_n} w_{s,r}^{n+1} \phi\left(a_s^n\right) \right) \tag{3.85}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} w_{q,r}^{n+1} \phi'\left(a_q^n\right) \tag{3.86}$$

$$= o_p^{n-1} \phi'\left(a_q^n\right) \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} w_{q,r}^{n+1} \tag{3.87}$$

となる．これは，順伝搬時には計算されない δ_r^{n+1} に依存しているから，$n+1$ 層目についての勾配計算を先に行う必要があることがわかる．従って，最終層のみ直ちに勾配を計算可能であり，それ以外の層は自身の次の層に依存しているから，出力から入力へと DNN を遡る方向に計算する，誤差逆伝播法が効率の良いアルゴリズムだと言える．

3.2.7　学習の安定化

DNN の学習は勾配降下法によって行われるが，ここで勾配が大きくなりすぎると重みの更新幅が過剰に大きくなり，学習が不安定になる可能性がある．これに対して，Gradient Clipping が有効である．これは，

$$\nabla\mathcal{L}\left(\boldsymbol{\theta};\mathcal{I}\right) \leftarrow \frac{c}{\max\left\{\left\|\nabla\mathcal{L}\left(\boldsymbol{\theta};\mathcal{I}\right)\right\|_2, c\right\}}\nabla\mathcal{L}\left(\boldsymbol{\theta};\mathcal{I}\right) \tag{3.88}$$

で与えられる．ここで，c は勾配の L2 ノルムに対する閾値である．$\mathcal{L}\left(\boldsymbol{\theta};\mathcal{I}\right)$ は 3.2.2節で定義した関数を再度用いた．

また，近年は数億単位のパラメータを持つ大規模な DNN も提案されており，こういった規模間の DNN を構築して学習する場合，それ相応のメモリが必要になる．マシンのスペックに対し，バッチサイズを十分小さくすれば基本的に学習は可能であるが，これは各データのノイズの影響が強くなるため，学習を不安定にする要因となる．これに対し，Gradient Accumulation が有効である．Gradient Accumulation は，小さなバッチサイズで計算した勾配を複数イテレーションに渡って累積し，設定したイテレーション数ごとに重みの更新を行う手法である．累積される勾配を $\boldsymbol{g}_{\mathrm{accum}}$ とすると，この更新は

$$\boldsymbol{g}_{\mathrm{accum}} \leftarrow \boldsymbol{g}_{\mathrm{accum}} + \nabla\mathcal{L}\left(\boldsymbol{\theta}_{\tau-1};\mathcal{I}_{\tau}\right) \tag{3.89}$$

で与えられる．ここで，設定した累積回数を N_{accum} とすると，重み $\boldsymbol{\theta}$ の更新は

$$\boldsymbol{\theta}_{\tau} = \boldsymbol{\theta}_{\tau-1} - \frac{\eta}{N_{\mathrm{accum}}}\boldsymbol{g}_{\mathrm{accum}} \tag{3.90}$$

で与えられる．N_{accum} 回分の勾配を累積した分，重みを更新する際には $1/N_{\mathrm{accum}}$ 倍して平均をとることで，実質的に N_{accum} 倍のバッチサイズにおける学習が可能になる．また，重み更新後は累積した勾配を 0 にリセットして，次の N_{accum} 回の累積に備える．

4　動画音声合成モデルの検討

4.1　音声合成法

4.1.1　全体像

本実験で用いる学習データセット $\mathcal{D}_{\text{train}}$ を

$$\mathcal{D}_{\text{train}} = \left\{ \left(\boldsymbol{X}_n^{\text{video}}, \boldsymbol{y}_n^{\text{sp-wf}}, \boldsymbol{x}_n^{\text{spk-emb}}, \boldsymbol{Y}_n^{\text{mel}}, \boldsymbol{Y}_n^{\text{HuB-int}}, \boldsymbol{Y}_n^{\text{HuB-disc}} \right) \right\}_{n=1}^{N} \tag{4.1}$$

とする．各 n に対し，$\boldsymbol{X}_n^{\text{video}} \in \mathbb{R}^{T_n^{\text{video}} \times W \times H}$ は口唇動画，$\boldsymbol{y}_n^{\text{sp-wf}} \in \mathbb{R}^{T_n^{\text{sp-wf}}}$ は原音声の音声波形，$\boldsymbol{x}_n^{\text{spk-emb}} \in \mathbb{R}^{D^{\text{spk-emb}}}$ は話者ベクトル，$\boldsymbol{Y}_n^{\text{mel}} \in \mathbb{R}^{T_n^{\text{mel}} \times D^{\text{mel}}}$ はメルスペクトログラム，$\boldsymbol{Y}_n^{\text{HuB-int}} \in \mathbb{R}^{T_n^{\text{HuB}} \times D^{\text{HuB-int}}}$ は HuBERT 中間特徴量，$\boldsymbol{Y}_n^{\text{HuB-disc}} \in \{0,1\}^{T_n^{\text{HuB}} \times C}$ は HuBERT 離散特徴量とする．T に添字をつけて各サンプル n ごとの各特徴量の系列長，D に添字をつけて各特徴量の次元，W, H で動画の幅と高さ，C でクラス数を表した．以下，新たに定義される変数も同様の表記でデータの形状を表した．

ここで，話者ベクトル $\boldsymbol{x}_n^{\text{spk-emb}}$ は，話者識別モデル [26] を利用して得られた音声の話者性を反映するベクトル（d-vector）である．話者識別モデルは，LSTM と全結合層から構成され，softmax 関数を用いた Generalized End-to-End Loss を用いて学習される．これは，各話者ごとに複数の発話から得られた話者ベクトルの平均を重心とし，各話者ベクトルについて同一話者の重心とのコサイン類似度を最大化し，異なる話者の重心とのコサイン類似度を最小化するように設計された損失関数である．よって，学習されたモデルによって得られる話者ベクトルは，話者間での識別性を有したベクトルとなる．本実験ではこれをモデルへの入力とすることで，動画から音声を予測する際に話者性を正しく反映させることを狙った．

また，HuBERT 中間特徴量と HuBERT 離散特徴量は，HuBERT を利用して得られる特徴量である．HuBERT は，1 次元畳み込み層を中心とした畳み込みエンコーダと Transformer から構成され，Masked Prediction によって学習される．これは，音声波形を畳み込みエンコーダに通すことで特徴量に変換し，その一部区間をマスクしてから Transformer に入力して，マスクされた区間の教師ラベルを予測させる学習方法である．損失関数には，マスクされた区間に限定した Cross Entropy Loss が用いられる．ここで，教師ラベルは学習の過程で更新されるため，学習は 2 段階に分かれる．1 段階目は，MFCC を k-means 法によってクラスタリングした結果を教師ラベルとする．2 段階目は，1 段階目の学習済みモデルの Transformer における中間特徴量を k-means 法によってクラスタリングした結果を教師ラベルとする．比較的単純な MFCC ベースのラベルから，DNN によって得られるより複雑なラベルへと予測対象を更新して学習難易度を高めることで，モデルの表現力を向上させる仕組みとなっている．HuBERT は，マスクされた入力から教師ラベルを予測する学習を行うため，特に音声の文脈的構造を学習可能だと考えられる．ここで，HuBERT における畳み込みエンコーダまでを $f_{\text{HuB-conv}}$，HuBERT における Transformer を $f_{\text{HuB-trans}}$，k-means 法を $f_{\text{k-means}}$，インデックス系列を One-hot ベクトルに変換する関数を one-hot とすると，HuBERT 中間特徴量 $\boldsymbol{Y}_n^{\text{HuB-int}}$ および HuBERT 離散特徴

量 $\boldsymbol{Y}_n^{\text{HuB-disc}}$ は，

$$\boldsymbol{Y}_n^{\text{HuB-int}} = f_{\text{HuB-conv}}\left(\boldsymbol{y}_n^{\text{sp-wf}}; \boldsymbol{\theta}_{\text{HuB-conv}}\right) \tag{4.2}$$

$$\boldsymbol{Y}_n^{\text{HuB-disc}} = \text{one-hot}\left(f_{\text{k-means}}\left(f_{\text{HuB-trans}}\left(\boldsymbol{Y}_n^{\text{HuB-int}}; \boldsymbol{\theta}_{\text{HuB-trans}}\right)\right)\right) \tag{4.3}$$

で与えられる．ここで，$\boldsymbol{\theta}_{\text{HuB-conv}}, \boldsymbol{\theta}_{\text{HuB-trans}}$ は HuBERT の事前学習済み重みを表す．Masked Prediction においてマスク対象となっていたのは，ここで定めた HuBERT 中間特徴量にあたる．

提案手法を図 4.1に示す．赤の平行四辺形が入出力，青の長方形が処理を表す．提案手法は，ネットワーク A，ネットワーク B，ボコーダの 3 つからなる．まず，ネットワーク A を f_{A} とすると，f_{A} の行う処理は，

$$\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}, \hat{\boldsymbol{Y}}_n^{\text{mel-A}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}} = f_{\text{A}}\left(\boldsymbol{X}_n^{\text{video}}, \boldsymbol{x}_n^{\text{spk-emb}}; \boldsymbol{\theta}_{\text{A}}\right) \tag{4.4}$$

と表される．ここで，$\hat{\boldsymbol{Y}}_n^{\text{HuB-int}} \in \mathbb{R}^{T_n^{\text{HuB}} \times D^{\text{HuB-int}}}$ は予測 HuBERT 中間特徴量，$\hat{\boldsymbol{Y}}_n^{\text{mel-A}} \in \mathbb{R}^{T_n^{\text{mel}} \times D^{\text{mel}}}$ はネットワーク A の予測メルスペクトログラム，$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}} \in \mathbb{R}^{T_n^{\text{HuB}} \times C}$ はネットワーク A の HuBERT 離散特徴量に対するロジットを表す．次に，ネットワーク B を f_{B} とすると，f_{B} の行う処理は，

$$\hat{\boldsymbol{Y}}_n^{\text{mel-B}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}} = f_{\text{B}}\left(\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}, \boldsymbol{x}_n^{\text{spk-emb}}; \boldsymbol{\theta}_{\text{B}}\right) \tag{4.5}$$

と表される．ここで，$\hat{\boldsymbol{Y}}_n^{\text{mel-B}} \in \mathbb{R}^{T_n^{\text{mel}} \times D^{\text{mel}}}$ はネットワーク B の予測メルスペクトログラム，$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}} \in \mathbb{R}^{T_n^{\text{HuB}} \times C}$ はネットワーク B の HuBERT 離散特徴量に対するロジットを表す．最後に，音声波形を生成するボコーダを f_{voc} とすると，f_{voc} の行う処理は，

$$\hat{\boldsymbol{y}}_n^{\text{sp-wf}} = f_{\text{voc}}\left(\hat{\boldsymbol{Y}}_n^{\text{mel-B}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}; \boldsymbol{\theta}_{\text{voc}}\right) \tag{4.6}$$

と表される．まとめると，提案手法は口唇動画と話者ベクトルを入力とし，ネットワーク A とネットワーク B によって中間表現を獲得後，これをボコーダに入力することで音声波形を生成するものである．

4.1.2 ネットワーク A

ネットワーク A を図 4.2aに示す．赤の平行四辺形が入出力，青の長方形が処理を表す．ネットワーク A では，まず，口唇動画 $\boldsymbol{X}_n^{\text{video}}$ を AVHuBERT に通すことで，特徴量 $\boldsymbol{H}_n^{\text{A}} \in \mathbb{R}^{T_n^{\text{video}} \times D^{\text{A}}}$ に変換する．これは，

$$\boldsymbol{H}_n^{\text{A}} = f_{\text{AVHuB}}\left(\boldsymbol{X}_n^{\text{video}}; \boldsymbol{\theta}_{\text{A-AVHuB}}\right) \tag{4.7}$$

と表される．$\boldsymbol{\theta}_{\text{A-AVHuB}}$ は，AVHuBERT の事前学習済み重みで初期化した．AVHuBERT は，動画入力に対する 3 次元畳み込み層と 2 次元畳み込み層を中心とした ResNet，メルスペクトログラム入力に対する全結合層，これらの出力を次元方向に結合した特徴量を入力とする Transformer から構成され，HuBERT と同様に Masked Prediction によって学習される．AVHuBERT では，動画およびメルスペクトログラムに対してマスクが適用され，マスクされた区間の教師ラベル

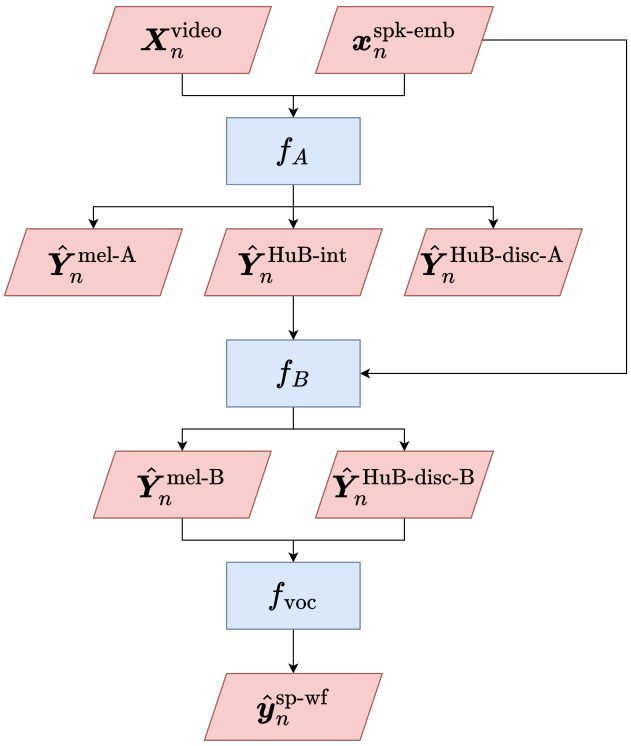


図 4.1: 提案手法の全体像（赤い平行四辺形：入出力，青い長方形：処理）

を予測させる．損失関数には，マスクされた区間に限定した Cross Entropy Loss が用いられる．また，ResNet から出力される動画特徴量と，全結合層から出力される音声特徴量を次元方向に結合する前には，Modality Dropout が行われる．これは，各フレームにおいて両方のモダリティを入力とするか，どちらか一方のモダリティのみを入力とする（他方を 0 に置き換える）かを確率的に選択するものである．これにより，どちらか一方のモダリティのみに依存しないようにモデルを学習させることが可能となるため，特定のタスクに fine-tuning する際には，Masked Prediction を行う場合のように動画と音声の両方を入力とせず，どちらか一方のモダリティのみを入力としても破綻しないよう設計されている．また，AVHuBERT でも，HuBERT と同様に教師ラベルの更新を段階的に行う．1 段階目は MFCC を k-means 法によってクラスタリングした結果を教師ラベルとし，2 から 5 段階目は前回の学習済みモデルの Transformer における中間特徴量を k-means 法によってクラスタリングした結果を教師ラベルとする．比較的単純な MFCC ベースのラベルから，DNN によって得られるより複雑なラベルへと予測対象を更新して学習難易度を高めることで，モデルの表現力を向上させる仕組みとなっている．AVHuBERT は，HuBERT と同様に動画および音声の文脈的構造を学習可能だと考えられるが，さらに 2 つのモダリティが組み合わさることで，これらの対応関係なども合わせたより複雑な関係性が学習可能だと考えられる．動画音声合成では，先行研究において AVHuBERT を転移学習することの有効性がすでに示されているため，本実験においても採用した．

次に，$\boldsymbol{H}_n^{\text{A}}$ の各時刻 t におけるベクトルに対し，話者ベクトル $\boldsymbol{x}_n^{\text{spk-emb}}$ を次元方向に結合して

から，全結合層によって次元を再度圧縮し，元の次元に戻す．これは，

$$\boldsymbol{H}_n^{\mathrm{A}} = \mathrm{FC}\left(\left[\boldsymbol{H}_n^{\mathrm{A}}, \boldsymbol{x}_n^{\text{spk-emb}}\mathbf{1}^\top\right]; \boldsymbol{\theta}_{\text{A-fc-spk}}\right) \tag{4.8}$$

と表される．ここで，$\mathbf{1} \in \{1\}^{T_n^{\text{video}}}$ は全成分が 1 のベクトルである．次に，話者ベクトルが統合された特徴量に対する後処理として，f_{post} を適用する．これは，

$$\boldsymbol{H}_n^{\mathrm{A}} = f_{\text{post}}\left(\boldsymbol{H}_n^{\mathrm{A}}; \boldsymbol{\theta}_{\text{A-post}}\right) \tag{4.9}$$

と表される．ここで，後処理層 f_{post} の構造を図 4.3に示す．赤の平行四辺形が入出力，青の長方形が処理を表す．内部特徴量は全て $\boldsymbol{H}_n$ で表した．f_{post} は，1 次元畳み込み層を中心とした ConvBlock および，これに残差結合を組み合わせた ResBlock から構成される．f_{post} は，話者ベクトル $\boldsymbol{x}_n^{\text{spk-emb}}$ が次元方向に結合された特徴量 $\boldsymbol{H}_n^{\mathrm{A}}$ に対する話者性を考慮した変換を行うために導入した．なぜなら，f_{AVHuB} は Masked Prediction によって事前学習されたモデルだから，動画の文脈的な構造を考慮するのに適していると考えられる一方で，音声合成に必要となる話者性は発話に依存しない，動画の文脈的な構造とは別の性質を持った情報だと考えたからである．

最後に，$\boldsymbol{H}_n^{\mathrm{A}}$ を全結合層を通して変換することで，予測対象である HuBERT 中間特徴量，メルスペクトログラム，HuBERT 離散特徴量に対するロジットを得る．これは，

$$\hat{\boldsymbol{Y}}_n^{\text{HuB-int}} = \mathrm{FC}\left(\boldsymbol{H}_n^{\mathrm{A}}; \boldsymbol{\theta}_{\text{A-fc-HuB-int}}\right) \tag{4.10}$$

$$\hat{\boldsymbol{Y}}_n^{\text{mel-A}} = \mathrm{FC}\left(\boldsymbol{H}_n^{\mathrm{A}}; \boldsymbol{\theta}_{\text{A-fc-mel}}\right) \tag{4.11}$$

$$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}} = \mathrm{FC}\left(\boldsymbol{H}_n^{\mathrm{A}}; \boldsymbol{\theta}_{\text{A-fc-HuB-Disc}}\right) \tag{4.12}$$

と表される．$\boldsymbol{\theta}_{\text{A-fc-spk}}, \boldsymbol{\theta}_{\text{A-post}}, \boldsymbol{\theta}_{\text{A-fc-HuB-int}}, \boldsymbol{\theta}_{\text{A-fc-mel}}, \boldsymbol{\theta}_{\text{A-fc-HuB-Disc}}$ は，すべてランダムに初期化した．

ネットワーク A の役割は，続くネットワーク B の入力である HuBERT 中間特徴量を予測することである．これに対し，メルスペクトログラムと HuBERT 離散特徴量の予測を同時に行った理由は，先行研究においてマルチタスク学習の有効性が確認されており，HuBERT 中間特徴量の予測においても有効ではないかと考えたからである．

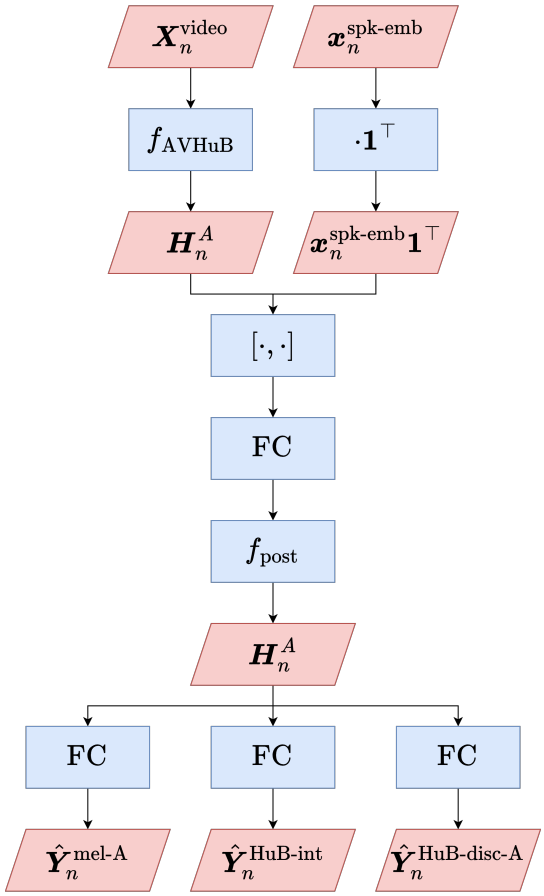
4.1.3 ネットワーク B

ネットワーク B を図 4.2bに示す．赤の平行四辺形が入出力，青の長方形が処理を表す．ネットワーク B では，まず，ネットワーク A で得られた予測 HuBERT 中間特徴量 $\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}$ を $f_{\text{HuB-trans}}$ に通すことで，特徴量 $\boldsymbol{H}_n^{\mathrm{B}} \in \mathbb{R}^{T_n^{\text{HuB}} \times D^{\mathrm{B}}}$ に変換する．これは，

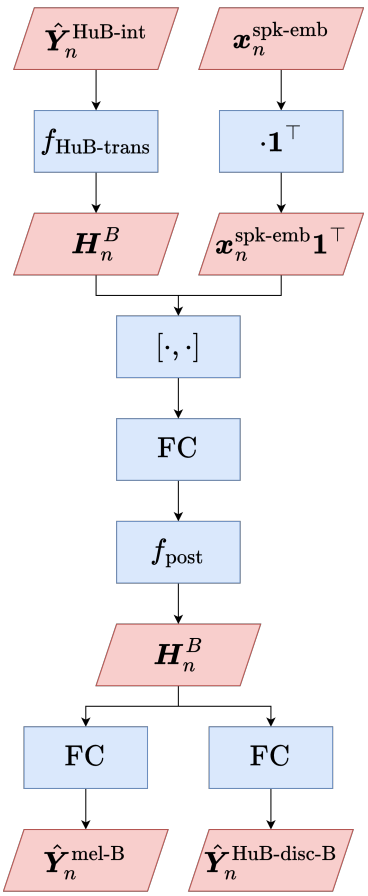
$$\boldsymbol{H}_n^{\mathrm{B}} = f_{\text{HuB-trans}}\left(\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}; \boldsymbol{\theta}_{\text{B-HuB-trans}}\right) \tag{4.13}$$

と表される．$\boldsymbol{\theta}_{\text{B-HuB-trans}}$ は，HuBERT の事前学習済み重みで初期化する場合と，ランダム初期化する場合の 2 つを検討した．以下，ネットワーク A と処理は同様であるため，数式のみ記載する．

$$\boldsymbol{H}_n^{\mathrm{B}} = \mathrm{FC}\left(\left[\boldsymbol{H}_n^{\mathrm{B}}, \boldsymbol{x}_n^{\text{spk-emb}}\mathbf{1}^\top\right]; \boldsymbol{\theta}_{\text{B-fc-spk}}\right) \tag{4.14}$$



(a) ネットワーク A



(b) ネットワーク B

図 4.2: ネットワーク A とネットワーク B の構造（赤い平行四辺形：入出力，青い長方形：処理）

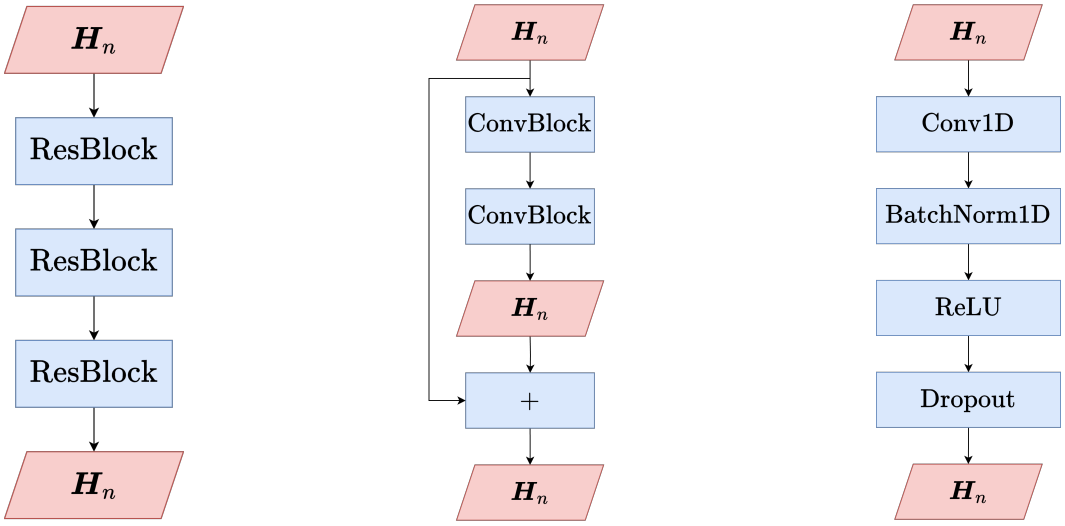
$$\boldsymbol{H}_n^{\text{B}} = f_{\text{post}}\left(\boldsymbol{H}_n^{\text{B}}; \boldsymbol{\theta}_{\text{B-post}}\right) \tag{4.15}$$

$$\hat{\boldsymbol{Y}}_n^{\text{mel-B}} = \text{FC}\left(\boldsymbol{H}_n^{\text{B}}; \boldsymbol{\theta}_{\text{B-fc-mel}}\right) \tag{4.16}$$

$$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}} = \text{FC}\left(\boldsymbol{H}_n^{\text{B}}; \boldsymbol{\theta}_{\text{B-fc-HuB-disc}}\right) \tag{4.17}$$

$\boldsymbol{\theta}_{\text{B-fc-spk}}, \boldsymbol{\theta}_{\text{B-post}}, \boldsymbol{\theta}_{\text{B-fc-mel}}, \boldsymbol{\theta}_{\text{B-fc-HuB-disc}}$ は，すべてランダムに初期化した．

ネットワーク B の役割は，メルスペクトログラムと HuBERT 離散特徴量の予測である．このネットワークの設計には，HuBERT Transformer が持つ自己教師あり学習の特性を活用する意図がある．HuBERT の Masked Prediction では，畳み込みエンコーダからの出力にマスクを適用し，Transformer を通じてマスクされたフレームの教師ラベルを予測する．よって，Transformer は不完全な入力をもとに教師ラベルを予測する必要があるため，全体の文脈を考慮して情報を補完する力を持つよう最適化されると考えられる．さらに，HuBERT は音声データのみで学習可能であり，動画音声データの制約を受けず，より大規模なデータセットによる事前学習が可能である．以上の特性に基づき，ネットワーク A の不完全な予測結果を，大量の音声データを元に学習された文脈を考慮する力によって補完することで，最終予測値であるメルスペクトログラムと HuBERT 離散特徴量に対する予測精度改善を狙った．



(a) 全体　(b) ResBlock　(c) ConvBlock

図 4.3: 後処理層 f_{post} の構造（赤い平行四辺形：入出力，青い長方形：処理）

4.1.4　ボコーダ

ボコーダを図 4.4に示す．赤の平行四辺形が入出力，青の長方形が処理を表す．本実験で用いるボコーダは，今回ベースラインとする先行研究 [8] で提案された Multi-input Vocoder を参考にしたものである．まず，ネットワーク B で得られた予測メルスペクトログラム $\hat{\boldsymbol{Y}}_n^{\text{mel-B}}$ と HuBERT 離散特徴量に対するロジット $\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}$ を前処理層に通し，扱いやすい形状に変換する．これは，

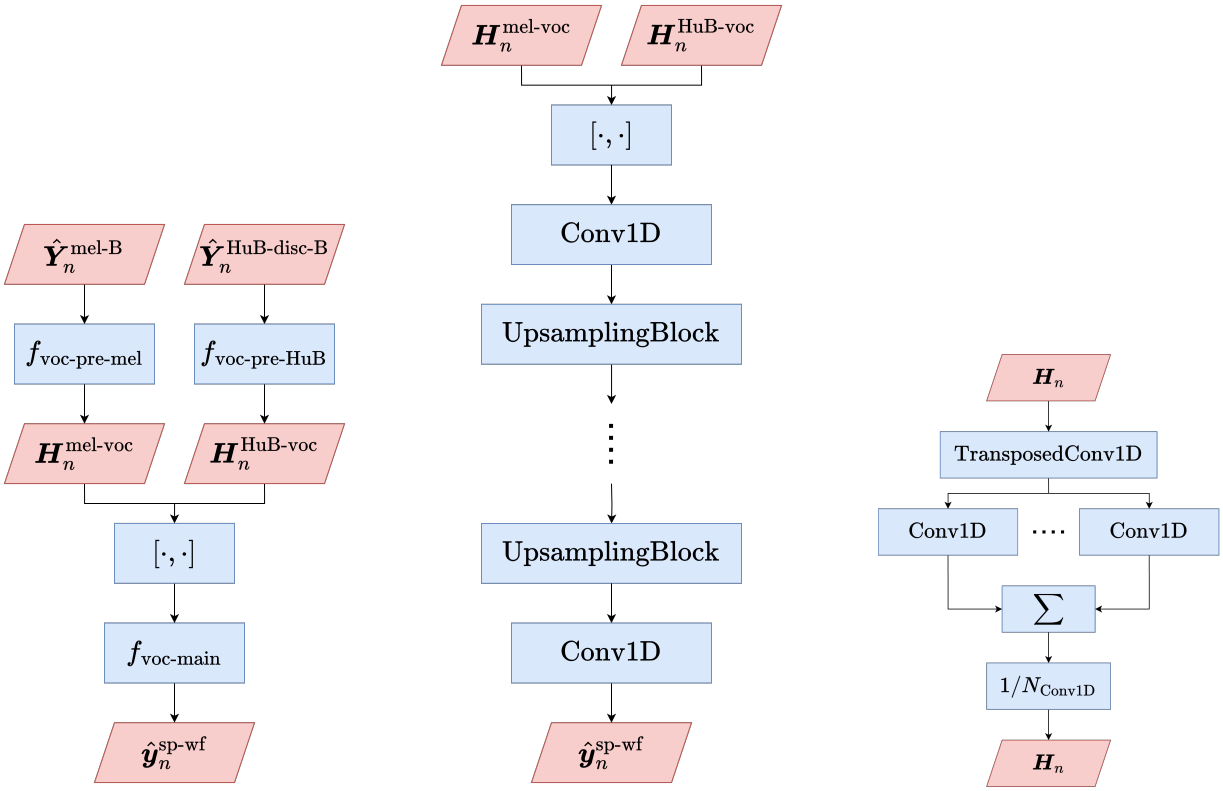
$$\boldsymbol{H}_n^{\text{mel-voc}} = f_{\text{voc-pre-mel}}\left(\hat{\boldsymbol{Y}}_n^{\text{mel-B}}; \boldsymbol{\theta}_{\text{voc-pre-mel}}\right) \tag{4.18}$$

$$\boldsymbol{H}_n^{\text{HuB-voc}} = f_{\text{voc-pre-HuB}}\left(\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}; \boldsymbol{\theta}_{\text{voc-pre-HuB}}\right) \tag{4.19}$$

と表される．$\boldsymbol{H}_n^{\text{mel-voc}} \in \mathbb{R}^{T_n^{\text{HuB}} \times D^{\text{mel-voc}}}$ は，$\hat{\boldsymbol{Y}}_n^{\text{mel-B}}$ の時間方向に隣接したベクトルを次元方向に結合することで $\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}$ と系列長を揃え，その後全結合層を適用した特徴量である．一方，$\boldsymbol{H}_n^{\text{HuB-voc}} \in \mathbb{R}^{T_n^{\text{HuB}} \times D^{\text{HuB-voc}}}$ は，$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}$ に対して argmax 関数を適用した後，各時刻 t において選択されたインデックスをベクトルに変換した特徴量である．次に，$\boldsymbol{H}_n^{\text{mel-voc}}, \boldsymbol{H}_n^{\text{HuB-voc}}$ を入力として，音声波形を生成する．これは，

$$\hat{\boldsymbol{y}}_n^{\text{sp-wf}} = f_{\text{voc-main}}\left(\left[\boldsymbol{H}_n^{\text{mel-voc}}, \boldsymbol{H}_n^{\text{HuB-voc}}\right]; \boldsymbol{\theta}_{\text{voc-main}}\right) \tag{4.20}$$

と表される．$f_{\text{voc-main}}$ は，図 4.4bに示すように，1 次元畳み込み層と UpsamplingBlock から構成され，これは HiFi-GAN[27] の Generator と同様である．各 UpsamplingBlock では，初めに 1 次元転置畳み込み層を通すことで時間方向のアップサンプリングを行い，その後複数の 1 次元畳み込み層から特徴抽出を行った結果を平均して出力する．図 4.4cにおいて，内部特徴量は $\boldsymbol{H}_n$ で表した．また，N_{Conv1D} は，UpsamplingBlock に並列に設けられている 1 次元畳み込み層の数を表す．



(a) 全体　　(b) $f_{\text{voc-main}}$　　(c) UpsamplingBlock

図 4.4: ボコーダの構造（赤い平行四辺形：入出力，青い長方形：処理）

4.1.5　損失関数

まず，ネットワーク A の学習に用いる損失関数 L_A は，

$$\begin{aligned} &L_A\left(\boldsymbol{Y}_n^{\text{HuB-int}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-int}}, \boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-A}}, \boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}}\right) \\ &=\lambda_{\text{HuB-int}} L_{\text{MAE}}\left(\boldsymbol{Y}_n^{\text{HuB-int}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-int}}\right) + \lambda_{\text{mel}} L_{\text{MAE}}\left(\boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-A}}\right) \\ &\quad + \lambda_{\text{HuB-disc}} L_{\text{CE}}\left(\boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}}\right) \end{aligned} \tag{4.21}$$

で与えられる．すなわち，HuBERT 中間特徴量についての MAE Loss，メルスペクトログラムについての MAE Loss，HuBERT 離散特徴量についての Cross Entropy Loss の重み付け和である．

次に，ネットワーク B の学習に用いる損失関数 L_B は，

$$\begin{aligned} &L_B\left(\boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-B}}, \boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}\right) \\ &=\lambda_{\text{mel}} L_{\text{MAE}}\left(\boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-B}}\right) + \lambda_{\text{HuB-disc}} L_{\text{CE}}\left(\boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}\right) \end{aligned} \tag{4.22}$$

で与えられる．すなわち，メルスペクトログラムについての MAE Loss，HuBERT 離散特徴量についての Cross Entropy Loss の重み付け和である．

表 4.1: 利用したデータセットの各分割における発話数

	学習	検証	テスト
動画音声データセット	1598	200	212
Hi-Fi-Captain	37714	200	200
JVS	10398	1299	1300

最後に，ボコーダの学習に用いる損失関数について，これは HiFi-GAN と同様に，音声波形をメルスペクトログラムに変換して計算される MAE Loss（元論文では L1 Loss の期待値として定義される）と，Multi-scale Discriminator（MSD）および Multi-period Discriminator（MPD）を用いて計算される GAN Loss，Feature Matching Loss の重み付け和とした．

4.2 実験方法

4.2.1 利用したデータセット

動画音声データセットには，先行研究 [28, 29] で収録されたもののうち，ATR 音素バランス文 [30] を読み上げたデータを利用した．このデータセットは，男女 2 人ずつから収録された合計 4 人分のデータから構成される．分割は，A から H セットを学習データ，I セットを検証データ，J セットをテストデータとした．各分割における発話数を表 4.1に示す．

ボコーダの学習には，Hi-Fi-Captain[31] と JVS[32] を利用した．Hi-Fi-Captain は，日本語話者 2 名と英語話者 2 名からなるデータセットであるが，本実験では日本語話者 2 名分のデータのみを利用した．分割は，train-parallel および train-non-parallel を学習データ，val を検証データ，eval をテストデータとした．各分割における発話数を表 4.1に示す．JVS は 100 人の日本語話者からなるデータであり，各話者に対して 1 から 100 まで番号が割り振られている．読み上げ音声の parallel100 および nonpara30 と，裏声の falset10，囁き声の whisper10 が含まれるが，本研究では parallel100 と nonpara30 のみ利用した．分割は，1 から 80 番の話者を学習データ，81 番から 90 番の話者を検証データ，91 番から 100 番までの話者をテストデータとした．各分割における発話数を表 4.1に示す．

4.2.2 データの前処理

動画データは 60 fps で収録されたものを ffmpeg により 25 fps に変換して用いた．その後，手法 [33] により動画に対してランドマーク検出を適用した[1]．このランドマークを利用することで口元のみを切り取り，画像サイズを $(96, 96)$ にリサイズした．モデル入力時は動画をグレースケールに変換し，各フレームに対する正規化および標準化を適用した．全体として，今回は AVHuBERT の転移学習を行うため，そこでの前処理に合わせている．学習時のデータ拡張は，

[1] https://github.com/1adrianb/face-alignment

ランダムクロップ，左右反転，Time Masking を適用した．ランダムクロップは，$(96, 96)$ で与えられる画像から $(88, 88)$ をランダムに切り取る処理である．検証およびテスト時は，必ず画像中央を切り取るよう実装した．左右反転は，50%の確率で左右が反転されるよう実装した．Time Masking は，動画 1 s あたり 0 s から 0.5 s の間でランダムに停止区間を定め，その区間における動画の時間方向平均値を計算し，区間内のすべてのフレームをこの平均値で置換した．これにより，動画が一時停止されるような効果が得られる．

音声データは，サンプリング周波数を 16 kHz にして用いた．窓長 25 ms のハニング窓を用いてシフト幅 10 ms で STFT を適用することで，フレームレート 100 Hz のスペクトログラムに変換し，パワースペクトログラムに対して 80 次のメルフィルタバンクを適用した後，対数スケールに変換することで対数メルスペクトログラムを得た．また，Hi-Fi-Captain と JVS には，ボコーダの学習安定化のため，無音区間のトリミング（500 ms 継続する −40 dBFS 以下の区間を 100 ms までカット）を適用した．

モデルへの入力とした話者ベクトルは，各話者に対し学習データの中から 100 発話をランダムサンプリングし，各発話に対して得られたベクトルの平均値を用いた．これを学習・検証・テストで一貫して用いるため，検証データやテストデータには非依存な値となっている．

HuBERT 離散特徴量の計算に利用する HuBERT Transformer 層出力は，8 層目出力を利用した．HuBERT の層ごとの特徴量について，音素の One-hot ベクトルおよび単語の One-hot ベクトルとの相関を projection weighted Canonical Correlation Analysis（PWCCA）[34] によって調べた先行研究 [35] より，8 層目出力がそのどちらとも相関が高いことが示されている．本実験では，HuBERT 離散特徴量は言語的な情報を持つものとして扱いたかったため，この層からの出力を利用した．k-means 法のクラスタ数は 100 とし，動画音声データセットの学習用データを利用して学習した後，全データセットに対してクラスタリングを実施した．また，学習時はゼロパディングされる区間のためにクラスを 1 つ追加したため，合計 101 クラスとして扱った．

4.2.3 本実験で利用した事前学習済みモデルについて

話者ベクトルの計算に用いた話者識別モデルの事前学習済み重みには，YouTube から構築された，合計 352 時間の動画音声データセットである VoxCeleb1[36]，同じく YouTube から構築された，合計 2442 時間の多言語動画音声データセットである VoxCeleb2，パブリックドメインとなった本に対する読み上げ音声データから構築された，合計 960 時間からなる英語音声データセットである LibriSpeech[37] を用いて学習されたものを利用した[2]．学習データセットの詳細な記述は，このモデルを利用して複数話者 TTS を検討した，先行研究 [38] の GitHub リポジトリに示されている[3]．話者識別モデルでは，まず 16 kHz の音声波形を，窓長 25 ms のハニング窓で，シフト幅 10 ms として 40 次元のメルスペクトログラムに変換した後，これを 1.6 s ごとに 0.77 s のオーバーラップを持つよう分割する．その後，各区間ごとにモデルに入力してベクトルに変換し，区間ごとの出力ベクトルを平均して正規化することで，最終出力を得る．モ

[2] https://github.com/resemble-ai/Resemblyzer

[3] https://github.com/CorentinJ/Real-Time-Voice-Cloning/wiki/Training

デルは 3 層の LSTM と全結合層からなり，最終的に得られる話者ベクトルの次元は 256 次元である．

AVHuBERT の事前学習済み重みには，TED と TEDx から構築された，合計 433 時間の英語動画音声データセットである LRS3 と，VoxCeleb2 のうち 1326 時間分の英語データを利用し，データ拡張として音声データに対するノイズ付与を行った上で学習されたモデル [39] の重みを利用した[4]．脚注のリンク先では,「Model: Noise-Augmented AV-HuBERT Base」,「Pretraining Data: LRS3 + VoxCeleb2 (En)」,「Finetuning Data: No finetuning」と示されている．データ拡張におけるノイズには，MUSAN データセット [40] に含まれる音楽やバブルノイズおよび，LRS3 に含まれる音声データが用いられている．AVHuBERT は 3 次元畳み込み層と 2 次元畳み込み層を中心とした ResNet と，12 層の Transformer 層から構成される．ResNet によって動画からの局所的な特徴の抽出および空間情報の圧縮が行われ，その後 Transformer を適用することで，時系列全体を考慮した特徴抽出が行われる．最終的に得られる特徴量は 768 次元 25 Hz である．ノイズ付与の場合を用いた理由について，音声と動画の両方を入力とする Audio-Visual Speech Recognition（AVSR）の先行研究 [41] では，音声から発話内容を予測する方が容易であるために，学習時にモデルが動画情報を考慮しない可能性があることを言及し，あえて音声データにノイズを付与してタスクを難しくすることで，動画情報の考慮を促す学習方法が提案されている．AVHuBERT は事前学習時に音声と動画の両方を入力とするが，本研究では動画側のみを利用するため，ノイズ付与によって動画情報の考慮が促進された場合の重みが適するのではないかと考えた．

HuBERT の事前学習済み重みには，テレビ番組の録画データから構築された，合計 19000 時間の日本語音声データセットである ReazonSpeech[42] によって学習されたモデル [43] を利用した[5]．今回用いる動画音声データセットが日本語であることから，このモデルが検討対象に適すると判断した．HuBERT は 1 次元畳み込み層を中心とした畳み込みエンコーダと，12 層の Transformer 層から構成される．畳み込みエンコーダによって音声波形の系列長を削減しつつ次元を上げた後，Transformer を適用することで，時系列全体を考慮した特徴抽出を行う．最終的に得られる特徴量は 768 次元 50 Hz である．

4.2.4 本実験で独自に構築したモデルについて

本実験で独自に構築したモデルは，ネットワーク A と B における f_{post} と，ボコーダ f_{voc} である．ボコーダについては HiFi-GAN の Generator と大枠は同様であるが，扱う音声波形のサンプリング周波数の違いや入力特徴量の違いから本実験専用に実装を改めたため，その詳細を述べる．

f_{post} について，ConvBlock における畳み込み層の次元は 768 次元とし，カーネルサイズは 3 とした．Dropout は $p = 0.1$ として用いた．ConvBlock から構成される ResBlock は 3 層積み重ねた．

[4] https://github.com/facebookresearch/av_hubert

[5] https://huggingface.co/rinna/japanese-hubert-base

表 4.2: $f_{\text{voc-main}}$ の各 UpsamplingBlock におけるパラメータと，各層における出力特徴量の形状

	転置畳み込み層 (K, S)	畳み込み層 (K, R)	出力特徴量の形状 (D, T)
1	$(11, 5)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(512, 250)$
2	$(8, 4)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(256, 1000)$
3	$(4, 2)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(128, 2000)$
4	$(4, 2)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(64, 4000)$
5	$(4, 2)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(32, 8000)$
6	$(4, 2)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(16, 16000)$

ボコーダについて，メルスペクトログラムの前処理層 $f_{\text{voc-pre-mel}}$ では，入力される 80 次元 100 Hz のメルスペクトログラムに対し，時間方向に隣接した 2 フレームを次元方向に結合することで 160 次元 50 Hz の特徴量に変換し，全結合層を適用して 128 次元まで次元を削減した．HuBERT 離散特徴量の前処理層 $f_{\text{voc-pre-HuB}}$ では，初めに各時刻 t におけるロジットに対して argmax を適用することで，最もスコアの高いクラスを選択した．ここで，学習時など原音声から計算されたインデックス系列が初めから入力される場合は，この処理をスキップした．その後，各時刻 t におけるインデックスを 128 次元のベクトルに変換することで，128 次元 50 Hz の特徴量を得た．最後に，前処理後の 2 つの特徴量を次元方向に結合することで，256 次元 50 Hz の特徴量を得た．次に，メインの処理層 $f_{\text{voc-main}}$ では，初めにカーネルサイズ 7 の畳み込み層により，次元を 1024 次元まで拡大した．これに対し，転置畳み込み層による時間方向のアップサンプリングと，複数種類の畳み込み層による特徴抽出を繰り返し行うことで，16 kHz の特徴量を獲得した．各 UpsamplingBlock におけるパラメータと，各層における出力特徴量の形状を表 4.2に示す．ここで，K はカーネルサイズ，S はストライド，R はダイレーション，D は特徴量の次元，T は系列長である．出力特徴量の形状については，入力が 1 s 分の場合を例として記載した．畳み込み層についてはカーネルサイズとダイレーションを集合として表記しているが，実際はこれらの直積の元，すなわち $(3, 1)$ や $(3, 3)$，$(3, 5)$ をパラメータとする層が存在することを表す．よって，転置畳み込み層 1 層に対し，その後の特徴量抽出は 15 種類の異なるカーネルサイズ，ダイレーションを設定した畳み込み層によって行ったこととなる．最後に，16 次元 16 kHz の特徴量を畳み込み層に通して 1 次元まで次元を削減することで，16 kHz の音声波形を得た．

4.2.5 学習方法

ネットワーク A について，最適化手法は AdamW[24] を利用し，$\beta_1 = 0.9, \beta_2 = 0.98, \lambda = 0.01$ とした．スケジューラは Cosine Annealing with Warmup を利用し，$\eta_{\min} = 1.0 \times 10^{-6}, \eta_{\max} = 1.0 \times 10^{-3}, \nu_{\text{warmup}} = 5, \nu_{\max} = 50$ とした．バッチサイズは 4 とし，8 イテレーションに 1 回重みを更新するよう Gradient Accumulation を適用した．モデルに入力する動画の秒数は 10 s を上限とし，10 s を超える場合はランダムにトリミング，10 s 未満の場合はゼロパディングした．勾

配のノルムは3.0を上限としてクリッピングした．10エポック連続して検証データに対する損失関数L_Aの値が小さくならない場合には学習を中断するようにし（Early Stopping），学習終了時には検証データに対する損失が最も小さかったエポックにおけるチェックポイントを保存して，これをテストデータに対する評価に用いた．L_Aの重み係数は，$\lambda_{\text{mel}} = 1.0, \lambda_{\text{HuB-int}} = 1.0$に固定した上で，$\lambda_{\text{HuB-disc}}$を0.0001, 0.001, 0.01, 0.1, 1.0の5段階でグリッドサーチした．

ネットワークBについて，ここではネットワークAの重みは固定した．最適化手法はAdamWを利用し，$\beta_1 = 0.9, \beta_2 = 0.98, \lambda = 0.01$とした．スケジューラはCosine Annealing with Warmupを利用し，$\eta_{\min} = 1.0 \times 10^{-6}, \eta_{\max} = 5.0 \times 10^{-4}, \nu_{\text{warmup}} = 5, \nu_{\max} = 50$とした．$\eta_{\max}$をネットワークAから半減したのは，学習の安定化のためである．その他の設定はネットワークAと同様で，Early Stoppingにおいて監視したのは検証データに対するL_Bの値である．L_Bの重み係数は，$\lambda_{\text{mel}} = 1.0$に固定した上で，$\lambda_{\text{HuB-disc}}$を0.0001, 0.001, 0.01, 0.1, 1.0の5段階でグリッドサーチした．ただし，ネットワークBにおいて用いるネットワークAの学習済み重みは，等しい$\lambda_{\text{HuB-disc}}$で学習されたものとした．例えば，ネットワークBにおいて$\lambda_{\text{HuB-disc}} = 0.0001$で学習させる時，ネットワークAも$\lambda_{\text{HuB-disc}} = 0.0001$で学習されたものを用いた．

ボコーダについて，ここではじめにHi-Fi-Captainのみを用いて学習させ，その後JVSによって再学習した．最適化手法はAdamWを利用し，$\beta_1 = 0.8, \beta_2 = 0.99, \lambda = 1.0 \times 10^{-5}$とした．スケジューラはExponentialLRSchedulerを利用し，$\eta_0 = 2.0 \times 10^{-4}, \gamma = 0.99$とした．また，最大エポック数は30とした．バッチサイズは16とした．モデルへの入力は1 sを上限とし，1 sを超える場合はランダムにトリミング，1 s未満の場合はゼロパディングした．ここではEarly Stoppingは適用せず，学習終了時に検証データに対する損失（メルスペクトログラムに対するMAE Loss）が最も小さかったエポックにおけるチェックポイントを保存し，これをテストデータに対する評価に用いた．

実装に用いた深層学習ライブラリはPyTorchおよびPyTorch Lightningである．GPUにはNVIDIA RTX A4000を利用し，計算の高速化のためAutomatic Mixed Precisionを適用した．

4.2.6 客観評価

合成音声の客観評価には，2種類の指標を用いた．1つ目は，音声認識の結果から算出した単語誤り率（Word Error Rate; WER）である．WERの計算方法について，まず，正解文字列s_1と音声認識モデルによる予測文字列s_2に対し，レーベンシュタイン距離によってその差分を測る．レーベンシュタイン距離は，2つの文字列を一致させるために必要なトークンの挿入数I，削除数D，置換数Rの和の最小値として定義される．WERは，レーベンシュタイン距離を測ることによって得られたI, D, Rを利用し，

$$\text{WER}\,(s_1, s_2) = \frac{I + D + R}{|s_1|} \tag{4.23}$$

で与えられる．ここで，$|s_1|$は正解文字列s_1のトークン数を表す．実際には，音声認識モデルにWhisper[44]を利用し[6]，出力される漢字仮名交じり文に対してMeCabを用いて分かち書き

[6] https://github.com/openai/whisper

を行った上で，jiwer というライブラリを用いて算出した．Whisper は Large モデルを利用し，MeCab の辞書には unidic を利用した．WER の値は 0%以上であり，この値が低いほど音声認識の誤りが少ないため，より聞き取りやすい音声であると判断した．

2 つ目は，話者ベクトルから計算したコサイン類似度である．モデルに入力する話者ベクトルを計算するのに用いた話者識別モデルを利用して，各サンプルごとに合成音声の話者ベクトルと原音声の話者ベクトルを計算し，これらのコサイン類似度を計算した．今回構築するモデルは 4 人の話者に対応するモデルとなるため，原音声に似た声質の合成音声が得られているかをこの指標で評価した．値は-1 以上 1 以下であり，高いほど原音声と似た合成音声だと判断した．

4.2.7 主観評価

合成音声の主観評価では，音声の明瞭性と類似性の 2 点を評価した．今回はクラウドワークスというクラウドソーシングサービスおよび，自作の実験用 Web サイトを利用してオンラインで実験を実施した．被験者の条件は，

1. 日本語話者である方
2. 聴覚に異常のない方
3. イヤホンあるいはヘッドホンでの実施が可能な方
4. 静かな環境での実施が可能な方
5. 以前に行った主観評価実験に参加されなかった方

とした．実験項目は，

1. アンケート
2. 練習試行（明瞭性）
3. 本番試行（明瞭性）
4. 練習試行（類似性）
5. 本番試行（類似性）

である．

1 つ目のアンケートでは，被験者についての基本的な統計を取ることを目的として，性別・年齢・実験に利用した音響機器について回答してもらった．性別は，男性，女性，無回答の 3 つからの選択式とした．年齢は，被験者の方に直接数値を入力してもらう形式とした．実験に利用した音響機器は，イヤホン，ヘッドホンの 2 つからの選択式とした．

2 つ目の練習試行（明瞭性）および 3 つ目の本番試行（明瞭性）では，音声の明瞭性の評価を実施した．初めに練習試行で実験内容を把握してもらい，その後本番施行を行う流れとした．練習施行は何度でも実施可能とし，本番試行は 1 回のみ実施可能とした．評価項目について，明瞭性は「話者の意図した発話内容を，1 回の発話でどの程度聞き取ることができたか」を評価するものとした．実際の評価プロセスは以下の 3 段階で構成した．

1. 提示された音声サンプルを 1 回再生し，発話内容を聞き取る．

2. 「発話内容を表示」ボタンを押し，本来の発話内容を表示する．
3. 聴取者が想定していた発話内容と本来の発話内容を照らし合わせ，音声の聞き取りやすさを 5 段階評価する．

5 段階評価の回答項目は以下のようにした．

1. 全く聞き取れなかった
2. ほとんど聞き取れなかった
3. ある程度聞き取れた
4. ほとんど聞き取れた
5. 完全に聞き取れた

実験に利用した音声サンプルについて，練習試行では検証データである ATR 音素バランス文の I セット，本番試行ではテストデータである ATR 音素バランス文の J セットを用いた．被験者ごとの評価サンプルの割り当て方法をアルゴリズム 3に示す．sentences は文章のリスト，methods は手法のリスト，speakers は話者のリスト，N_{p} は被験者総数である．各被験者について，初めに sentences と methods をランダムにシャッフルし，それから順に sentence, method, speaker を決めて，対応した音声サンプルを選択した．この方法では，2 つのことに注意した．1 つ目は，各被験者がユニークな発話内容を評価することである．各音声サンプルの評価の際に本来の発話内容が分かるため，これを知った上で同じ発話内容のサンプルを評価すると，音声自体の明瞭性に関わらず発話内容がわかってしまい，評価に影響を与える可能性がある．これを避けるため，評価サンプルは全て異なる発話内容にした．2 つ目は，各手法がなるべく均等な回数出現することである．今回の実験は手法の比較を目的としたため，各被験者がすべての手法をなるべく均等な回数評価できるようにした．また，評価時に音声サンプルを 1 回だけ聞けるようにしたことについて，代用音声をコミュニケーションツールとして利用する場面を想定したとき，会話において何度も聞き返されることは利用者のストレスになると考えられる．よって，本実験では 1 回の発話で意図した発話内容をどの程度聞き取ってもらえるかを聞き取りやすさとして評価したいと考え，この制限を設けた．

4 つ目の練習試行（類似性）および 5 つ目の本番試行（類似性）では，評価対象の音声と同一話者の原音声の類似性の評価を実施した．初めに練習試行で実験内容を把握してもらい，その後本番施行を行う流れとした．練習施行は何度でも実施可能とし，本番試行は 1 回のみ実施可能とした．評価項目について，類似性は「評価対象の音声が同一話者の原音声とどれくらい似ているか」を評価するものとした．実際の評価プロセスは以下の 2 段階で構成した．

1. 評価対象の音声と原音声を聞き比べる．
2. 評価対象の音声が原音声にどれくらい似ていたかを 5 段階評価する．

5 段階評価の回答項目は以下のようにした．

1. 全く似ていなかった
2. あまり似ていなかった

Algorithm 3 被験者に対する評価サンプル割り当て方法（明瞭性）

Input: sentences, methods, speakers, N_{p}
allAssignments = {}
for participantId = 0 to $N_{\mathrm{p}} - 1$ **do**
    assignments = []
    Randomly shuffle sentences
    Randomly shuffle methods
    for $i = 0$ to len (sentences) $- 1$ **do**
        sentence = sentences $[i]$
        method = methods $[i \bmod \mathrm{len}\,(\mathrm{methods})]$
        speaker = Randomly select from speakers
        Append [sentence, method, speaker] to assignments
    end for
    allAssignments [participantId] = assignments
end for
return allAssignments

3. やや似ていた
4. かなり似ていた
5. 同じ話者に聞こえた

実験に利用した音声サンプルは明瞭性評価と同じである．被験者ごとの評価サンプルの割り当て方法をアルゴリズム 4に示す．明瞭性実験と概ね同様であるが，評価サンプル用の発話文章である sentenceEval に対して，比較対象となる原音声用の発話文章である sentenceGT が追加された点が異なる．類似性評価は発話内容に依存しない評価だと考えて，sentenceGT はランダムに選択した．また，類似性評価では音声サンプルを何度でも聞けるようにした．明瞭性におけるコミュニケーションを想定した評価と異なり，単に原音声とどの程度似ているかを評価したいと考えたからである．ここで，類似性評価を明瞭性評価の後に行うようにした理由は，類似性評価と明瞭性評価に用いるデータが同一だったからである．類似性評価は発話内容に依存しない評価だと考えて，明瞭性評価を完了し，発話内容を知った上で評価しても問題ないと判断した．

また，オンラインでの評価は効率よく数多くの方に評価していただけるという点でメリットがあるが，オフラインでの評価と比較して実験環境を制御することが難しく，評価品質が低下する恐れがある．これに対して，本実験では先行研究 [45] を参考に，評価サンプル中にダミー音声を混入させることで対策を講じた．ダミー音声は本研究で得られた合成音声とは無関係に，gTTS というライブラリを用いて生成したサンプルである．明瞭性評価では

これはダミー音声です．明瞭性は「3: ある程度聞き取れた」を選択してください．

Algorithm 4 被験者に対する評価サンプル割り当て方法（類似性）

Input: sentences, methods, speakers, N_{p}
allAssignments = {}
for participantId = 0 to $N_{\mathrm{p}} - 1$ **do**
    assignments = []
    Randomly shuffle sentences
    Randomly shuffle methods
    for $i = 0$ to len (sentences) $- 1$ **do**
        sentenceEval = sentences [i]
        method = methods [i mod len (methods)]
        speaker = Randomly select from speakers
        sentenceGT = Randomly select from sentences
        Append [sentenceEval, method, speaker, sentenceGT] to assignments
    end for
    allAssignments [participantId] = assignments
end for
return allAssignments

のような発話内容の音声を，類似性評価では

> これはダミー音声です．類似性は「3: やや似ていた」を選択してください．

のような発話内容の音声を提示した．「3: ある程度聞き取れた」や「3: やや似ていた」の部分は，5 段階評価の回答項目からランダムに選択した．被験者には，ダミー音声が提示された場合はその音声自体の明瞭性や類似性とは無関係に，必ず音声で指定された評価値を選択するよう伝えた．練習試行では 1 つ，本番試行では 2 つ混入させ，本番試行でダミー音声に対する評価値を誤った被験者については，適当に回答した恐れがあると判断し，最終的な統計処理には用いなかった．

総被験者数は 75 人とし，謝礼は実験 1 回あたり 40 分程度要すると考えて，750 円とした．

4.3 結果

4.3.1 客観評価 1: ベースラインと提案手法の比較

本節では，ベースラインと提案手法の比較を行う．比較手法は，以下の 4 つである．

1. ベースライン
2. ネットワーク A
3. ネットワーク B（Randomized）

4. ネットワークB（Pretrained）

ベースラインは，提案手法におけるネットワークAで，メルスペクトログラムとHuBERT離散特徴量のみを予測する場合であり，損失関数はL_Bを用いた．ネットワークAは提案手法自体ではないが，その構成要素となっているため客観評価指標を確認した．ネットワークB（Randomized）は，HuBERT Transformerをランダム初期化した場合，ネットワークB（Pretrained）は，事前学習済み重みで初期化した場合である．これらを比較することで，HuBERT Transformerの転移学習の有効性を調べた．

まず，損失関数の重み係数$\lambda_{\text{HuB-disc}}$による客観評価指標の変化を表4.3に示す．各手法の客観評価指標ごとに最良値を下線で示し，各手法ごとに最適だと判断した$\lambda_{\text{HuB-disc}}$の行を太字で示している．ベースラインでは，$\lambda_{\text{HuB-disc}}$が0.001のときにWERが最も低く，0.01のときに話者類似度が最も高くなった．現状WERの高さが特に課題であるため，今回は0.001が最適だと判断した．ネットワークAでは，提案手法の構成要素であるため最適な手法の選択は行わなかった．ベースラインとの違いはHuBERT中間特徴量についてのMAE Lossが損失に含まれることであるが，客観評価指標の値はベースラインと概ね同様であった．ネットワークB（Randomized）では，$\lambda_{\text{HuB-disc}}$が0.1の時にWERが最も低く，0.0001の時に話者類似度が最も高くなった．ここではWERの低さを優先し，0.1が最適だと判断した．ネットワークB（Pretrained）では，$\lambda_{\text{HuB-disc}}$が0.1の時にWERが最も低く，0.01の時に話者類似度が最も高くなった．今回はWERが低いことを優先して，0.1が最適だと判断した．

以上の結果をもとに，最適な$\lambda_{\text{HuB-disc}}$における手法ごとの比較を行った結果を表4.4に示す．分析合成は，原音声から計算したメルスペクトログラムとHuBERT離散特徴量をボコーダへの入力とした合成音声であり，本実験において合成音声により達成され得る上限を表す．ベースライン，ネットワークB（Randomized），ネットワークB（Pretrained）は，それぞれ最適な$\lambda_{\text{HuB-disc}}$だと判断したものを載せている．また，ベースライン，ネットワークB（Randomized），ネットワークB（Pretrained）の中で，最良値を下線で示した．これより，ネットワークB（Randomized）はベースラインよりもWERが9.3%低く，話者類似度が0.004高いことが分かる．一方，ネットワークB（Pretrained）はベースラインよりもWERが10.4%低いが，話者類似度も0.058低いことが分かる．以上より，客観評価指標においては，$\lambda_{\text{HuB-disc}} = 0.1$としたネットワークB（Randomized）が最も優れていたと考えられる．

4.3.2 客観評価 2: 提案手法のさらなる検討

4.3.1節において良好な結果を示したのはネットワークB（Randomized）であったが，これはHuBERTの転移学習による改善を狙った本実験の仮説に反する結果であった．これに対し，本節では良好な結果を示したネットワークB（Randomized）に焦点を当てたさらなる検討の結果を述べる．

まず，入力特徴量について検討した．4.3.1節では事前学習済み重みを用いることの有効性を調べる目的があったため，入力特徴量はHuBERTの事前学習時に揃えて，HuBERT中間特徴量と

表 4.3: 損失関数の重み係数 $\lambda_{\text{HuB-disc}}$ による客観評価指標の変化

手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ベースライン	0.0001	57.3	0.833
ベースライン	**0.001**	**54.6**	**0.836**
ベースライン	0.01	56.2	0.837
ベースライン	0.1	58.2	0.784
ベースライン	1	59.0	0.685
ネットワーク A	0.0001	55.4	0.841
ネットワーク A	0.001	55.1	0.842
ネットワーク A	0.01	53.4	0.843
ネットワーク A	0.1	54.6	0.809
ネットワーク A	1	58.7	0.698
ネットワーク B（Randomized）	0.0001	60.6	0.852
ネットワーク B（Randomized）	0.001	56.7	0.829
ネットワーク B（Randomized）	0.01	54.4	0.847
ネットワーク B（Randomized）	**0.1**	**45.3**	**0.840**
ネットワーク B（Randomized）	1	45.5	0.712
ネットワーク B（Pretrained）	0.0001	60.1	0.841
ネットワーク B（Pretrained）	0.001	57.1	0.839
ネットワーク B（Pretrained）	0.01	56.8	0.860
ネットワーク B（Pretrained）	**0.1**	**44.2**	**0.778**
ネットワーク B（Pretrained）	1	48.1	0.685

した．しかし，ネットワーク B（Randomized）は事前学習済み重みを利用しないため，HuBERT 中間特徴量を入力とすることが意味をなしているか不明である．これに対し，ネットワーク A からマルチタスク学習によって同時に予測される，メルスペクトログラムと HuBERT 離散特徴量に対するロジットを組み合わせて入力する場合を比較した．メルスペクトログラムは，時間方向に隣接した 2 フレームを次元方向に結合することで 160 次元 50 Hz の特徴量に変換し，HuBERT 離散特徴量に対するロジットは 101 次元 50 Hz の特徴量としてそのまま用いた．これらを次元方向に結合することで 261 次元 50 Hz の特徴量を構成し，全結合層を通して HuBERT 中間特徴量と同じ 768 次元に変換することで，HuBERT Transformer への入力とした．結果を表 4.5に示す．下線は掲載した手法全てにおける最良値を示す．新たに検討した手法はネットワーク B（Randomized・Mel-HuB）とし，$\lambda_{\text{HuB-disc}} = 0.001$ の場合が最適だと判断した．これより，最適なネットワーク B（Randomized・Mel-HuB）はネットワーク B（Randomized）よりも話者類似度は 0.005 高いが，WER も 10.1%高いことが分かる．従って，HuBERT 中間特徴量は，メルスペクトログラムと HuBERT 離散特徴量のロジットから構成した特徴量と比較

表 4.4: 最適な $\lambda_{\text{HuB-disc}}$ における手法ごとの比較

手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ベースライン	0.001	54.6	0.836
ネットワーク B（Randomized）	0.1	45.3	<u>0.840</u>
ネットワーク B（Pretrained）	0.1	<u>44.2</u>	0.778
分析合成	-	3.7	0.956
原音声	-	3.7	1.000

して，ネットワーク B の予測精度改善につながるより良い入力特徴量であったと考えられる．

次に，HuBERT 中間特徴量をネットワーク B に与えている，ネットワーク A の学習方法を検討した．ここでは，これまでネットワーク A で採用していた HuBERT 中間特徴量，メルスペクトログラム，HuBERT 離散特徴量を予測するマルチタスク学習に対し，HuBERT 中間特徴量のみを予測するシングルタスク学習の場合を比較した．結果を表 4.6に示す．下線は掲載した手法全てにおける最良値を示す．新たに検討した手法はネットワーク B（Randomized・A-SingleTask）とし，$\lambda_{\text{HuB-disc}} = 0.1$ の場合が最適だと判断した．これより，最適なネットワーク B（Randomized・A-SingleTask）はネットワーク B（Randomized）よりも WER が 2.8%低く，話者類似度は 0.007 高いことがわかる．従って，ネットワーク A では HuBERT 中間特徴量，メルスペクトログラム，HuBERT 離散特徴量を予測するマルチタスク学習を行うよりも，HuBERT 中間特徴量のみを予測するシングルタスク学習を行う方が，ネットワーク B に対してより良い入力特徴量を与えられるのだと考えられる．

最後に，これまでネットワーク A とネットワーク B を 2 段階で学習させたのに対し，ネットワーク A からネットワーク B までを 1 度に学習させる方法を検討した．これにより，HuBERT 中間特徴量に表現力を制限させないことで，さらなる精度改善を狙った．ここでは，ネットワーク B の出力に対する損失関数 L_B のみを用いて学習させ，ネットワーク A の出力は単にネットワーク B への入力とした．ネットワーク A およびネットワーク B（Randomized）を未学習の状態で初期化した場合の結果を表 4.7に示す．下線は掲載した手法全てにおける最良値を示す．

表 4.5: HuBERT Transformer への入力特徴量を変化させた場合の比較

手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ネットワーク B（Randomized・Mel-HuB）	0.0001	59.5	0.840
ネットワーク B（Randomized・Mel-HuB）	**0.001**	**55.4**	**<u>0.845</u>**
ネットワーク B（Randomized・Mel-HuB）	0.01	56.2	0.803
ネットワーク B（Randomized・Mel-HuB）	0.1	57.7	0.795
ネットワーク B（Randomized・Mel-HuB）	1	58.1	0.711
ネットワーク B（Randomized）	0.1	<u>45.3</u>	0.840

表 4.6: ネットワーク A におけるマルチタスク学習の有無による比較

手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ネットワーク B（Randomized・A-SingleTask）	0.0001	54.0	<u>0.867</u>
ネットワーク B（Randomized・A-SingleTask）	0.001	52.2	0.865
ネットワーク B（Randomized・A-SingleTask）	0.01	51.8	0.843
ネットワーク B（Randomized・A-SingleTask）	**0.1**	**<u>42.5</u>**	**0.847**
ネットワーク B（Randomized・A-SingleTask）	1	43.0	0.768
ネットワーク B（Randomized）	0.1	45.3	0.840

表 4.7: ネットワーク A からネットワーク B までを未学習の状態で初期化し，1 度に学習させた場合の比較

手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ネットワーク E2E（Randomized）	0.0001	101.0	0.543
ネットワーク E2E（Randomized）	0.001	121.9	0.551
ネットワーク E2E（Randomized）	0.01	101.1	0.507
ネットワーク E2E（Randomized）	**0.1**	**62.0**	**0.767**
ネットワーク E2E（Randomized）	1	71.1	0.643
ネットワーク B（Randomized・A-SingleTask）	0.1	<u>42.5</u>	<u>0.847</u>

新たに検討した手法はネットワーク E2E（Randomized）とし，$\lambda_{\text{HuB-disc}} = 0.1$ が最適だと判断した．これより，最適なネットワーク E2E（Randomized）はネットワーク B（Randomized・A-SingleTask）よりも WER が 20.0%高く，話者類似度は 0.080 低いことがわかる．提案手法全体を未学習の状態から 1 度に学習させることは今回の条件下では困難であったため，ネットワーク A とネットワーク B に分けた 2 段階学習を行う場合に劣る結果となった．これに対し，ここまでの検討で最も良好な結果を示した，ネットワーク B（Randomized・A-SingleTask）かつ $\lambda_{\text{HuB-disc}} = 0.1$ の場合における学習済み重みでネットワーク A とネットワーク B を初期化して，ネットワーク全体を再学習させた場合の結果を表 4.8に示す．新たに検討した手法はネットワーク E2E（Pretrained）とし，ここでは $\lambda_{\text{HuB-disc}}$ の値は 0.1 のままで固定して，スケジューラである Cosine Annealing with Warmup の最大学習率 $\eta_{\max}$ を変更することでチューニングした．最適値は $\eta_{\max} = 5.0 \times 10^{-5}$ とした．これより，最適なネットワーク E2E（Pretrained）はネットワーク B（Randomized・A-SingleTask）よりも話者類似度が 0.006 高いが，WER も 1.0%高いことがわかる．以上より，今回の条件下では，2 段階学習によって得られた最適なモデルを初期値として学習の安定化を図っても，ネットワーク全体を再学習することは効果的でなかったと考えられる．

表 4.8: ネットワーク B（Randomized・A-SingleTask）かつ $\lambda_{\text{HuB-disc}} = 0.1$ の場合における学習済み重みでネットワーク A とネットワーク B を初期化し，1 度に学習させた場合の比較

手法	$\eta_{\max}$	WER [%]	話者類似度
ネットワーク E2E（Pretrained）	5.0×10^{-4}	44.6	0.851
ネットワーク E2E（Pretrained）	$\mathbf{5.0 \times 10^{-5}}$	**43.5**	**0.853**
ネットワーク E2E（Pretrained）	5.0×10^{-6}	43.6	0.853
ネットワーク B（Randomized・A-SingleTask）	5.0×10^{-4}	42.5	0.847

4.3.3 主観評価

主観評価実験では，4.3.1節および 4.3.2節の検討結果を踏まえ，以下の 5 種類の音声を評価対象とした．

1. ベースライン
2. ネットワーク B（Randomized）
3. ネットワーク B（Randomized・A-SingleTask）
4. 分析合成
5. 原音声

ここで，ネットワーク B（Randomized）およびネットワーク B（Randomized・A-SingleTask）が，今回の実験を通して選択した提案手法の代表である．4.2.7節で述べたように，主観評価では音声の明瞭性と類似性の 5 段階評価を実施した．ダミー音声に対する評価値を誤った被験者は存在しなかったため，75 人分のデータを統計処理に用いた．

被験者に対するアンケート結果について述べる．年齢層は 21 歳から 62 歳に渡り，箱ひげ図にてその分布を示した（図 4.5）．性別の内訳は男性 32 名，女性 43 名であり，実験に利用した音響機器の内訳はヘッドホンが 19 名，イヤホンが 56 名であった．

明瞭性と類似性の評価値について，手法ごとの標本平均と 95%信頼区間を表 4.9に示す．また，各手法の組み合わせについて，平均値の差の検定（Welch の t 検定）を行った結果を表 4.10,

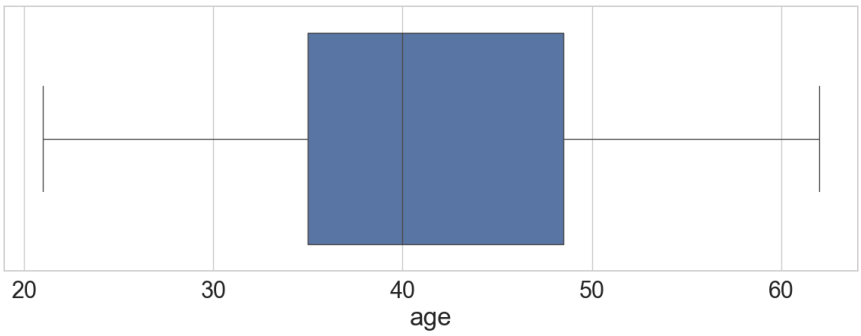


図 4.5: 主観評価実験における被験者の年齢層

表 4.9: 主観評価実験の結果より計算した標本平均と 95%信頼区間

手法	$\lambda_{\text{HuB-disc}}$	明瞭性	類似性
ベースライン	0.001	2.37 ± 0.07	2.92 ± 0.08
ネットワーク B（Randomized）	0.1	2.76 ± 0.07	3.04 ± 0.08
ネットワーク B（Randomized・A-SingleTask）	0.1	2.82 ± 0.08	3.18 ± 0.08
分析合成	-	4.75 ± 0.04	4.31 ± 0.07
原音声	-	4.87 ± 0.03	4.71 ± 0.05

4.11に示す．表における (i, j) 成分は，i 行目の手法に対する評価値の母平均を μ_i，j 列目の手法に対する評価値の母平均を μ_j とするとき，帰無仮説を $\mu_i = \mu_j$，対立仮説を $\mu_i > \mu_j$ とする片側検定で計算された p 値に対し，Holm–Bonferroni 法による多重比較のための補正を行った結果である．ここで，表の文字列は以下の意味である．

1. GT: 原音声（Ground Truth）
2. AbS: 分析合成（Analysis by Synthesis）
3. B（Rand, A-S）: ネットワーク B（Randomized・A-SingleTask）
4. B（Rand）: ネットワーク B（Randomized）
5. Baseline: ベースライン

本実験における有意水準は 5%とし，有意差があった場合を $*$ で示した．

まず，表 4.10より，提案したネットワーク B（Randomized）およびネットワーク B（Randomized・A-SingleTask）は，ベースラインよりも明瞭性の評価値が有意に高いことがわかる．一方，ネットワーク B（Randomized）とネットワーク B（Randomized・A-SingleTask）の間に有意差がないことから，ネットワーク A の学習方法の違いは明瞭性に顕著な違いをもたらさなかったと考えられる．次に，表 4.11より，提案したネットワーク B（Randomized）およびネットワーク B（Randomized・A-SingleTask）は，ベースラインよりも類似性の評価値が有意に高いことがわかる．また，ここではネットワーク B（Randomized・A-SingleTask）がネットワーク B（Randomized）よりも類似性の評価値が有意に高いこともわかる．従って，明瞭性と類似性の評価結果を合わせると，ネットワーク B（Randomized・A-SingleTask）が最も優れた動画音声合成モデルであったと考えられる．

また，良好な結果を示したネットワーク B（Randomized・A-SingleTask）であっても，分析合成や原音声との間には依然として大きな差があることがわかった．これより，本研究の目的であったベースラインからの改善は達成されたものの，原音声に匹敵する合成音声には至っていないと言える．従って，今後さらなるネットワークの改善が必要だと考える．

表 4.10: 主観評価実験の結果より計算した平均値の差の検定における p 値（明瞭性）

	AbS	B（Rand, A-S）	B（Rand）	Baseline
GT	$1.22 \times 10^{-5*}$	$1.85 \times 10^{-261*}$	$6.81 \times 10^{-289*}$	0^{*}
AbS	-	$9.67 \times 10^{-243*}$	$6.72 \times 10^{-270*}$	0^{*}
B（Rand, A-S）	-	-	1.23×10^{-1}	$1.98 \times 10^{-16*}$
B（Rand）	-	-	-	$2.68 \times 10^{-13*}$

表 4.11: 主観評価実験の結果より計算した平均値の差の検定における p 値（類似性）

	AbS	B（Rand, A-S）	B（Rand）	Baseline
GT	$3.48 \times 10^{-18*}$	$1.41 \times 10^{-159*}$	$1.78 \times 10^{-173*}$	$1.28 \times 10^{-205*}$
AbS	-	$9.32 \times 10^{-83*}$	$5.20 \times 10^{-97*}$	$1.00 \times 10^{-119*}$
B（Rand, A-S）	-	-	$2.04 \times 10^{-2*}$	$1.31 \times 10^{-5*}$
B（Rand）	-	-	-	$2.13 \times 10^{-2*}$

4.4 考察

4.4.1 ネットワーク B（Randomized）について

まず，ネットワーク B（Randomized）は客観評価において，ネットワーク B（Randomized・Mel-HuB）を上回る性能を示した．これより，ネットワーク B への入力として，HuBERT 中間特徴量はメルスペクトログラムと HuBERT 離散特徴量に対するロジットを組み合わせた特徴量よりも優れていたと考えられる．HuBERT 中間特徴量は，音声波形を入力とする畳み込みエンコーダの出力であり，Masked Prediction において Transformer の入力となる．従って，畳み込みエンコーダは音声波形の系列長を圧縮しつつ，文脈構造を考慮するための情報を効率的に抽出するよう最適化されると考えられる．よって，ネットワーク B（Randomized）における HuBERT Transformer はランダム初期化されてはいたものの，HuBERT 中間特徴量自体が文脈的構造の考慮に適した情報を含んでいるために，メルスペクトログラムと HuBERT 離散特徴量に対するロジットを入力特徴量とする場合よりも高い性能に達したと考えられる．

次に，ネットワーク B（Randomized・A-SingleTask）は客観評価と主観評価の両面において，ネットワーク B（Randomized）を上回る性能を示した．これより，HuBERT 中間特徴量を予測するネットワーク A は，メルスペクトログラムと HuBERT 離散特徴量を同時に予測するマルチタスク学習を行わず，HuBERT 中間特徴量のみを予測するよう学習した方が，ネットワーク B に対するより良い入力特徴量を与えられると考えられる．マルチタスク学習では，複数タスクの損失を考慮させることが相互作用的に良い影響を及ぼす場合もあるが，シングルタスクの場合より悪い局所最適解に収束する場合もある [46]．本実験条件下では，HuBERT 中間特徴量の損失に対し，メルスペクトログラムおよび HuBERT 離散特徴量の損失の干渉が悪影響を与えたと考えられる．

最後に，ネットワークE2E（Pretrained）は客観評価において，ネットワークB（Randomized・A-SingleTask）を上回ることはできなかった．これより，HuBERT中間特徴量を継ぎ目として2段階で学習したことは，ネットワーク全体としての表現力を損なうものではなかったと考えられる．HuBERT中間特徴量は元々DNNの中間表現であったため，2つのネットワーク間の継ぎ目として利用しても，十分な表現力を維持できたと考えられる．

4.4.2 ネットワークB（Pretrained）について

ネットワークB（Pretrained）は客観評価において，ネットワークB（Randomized）に劣る結果となった．これに対し，ネットワークB（Pretrained）が有効性を示さなかった理由は2つ考えられる．1つ目は，損失関数の重み係数$\lambda_{\text{HuB-disc}}$の適切な値が，グリッドサーチした範囲に存在しなかったことである．ネットワークB（Randomized）では適切な値が含まれていたが，グリッドサーチの候補は決め打ちであるから，ネットワークB（Randomized）にとって適した値を探索すれば，高い性能を発揮した可能性がある．2つ目は，事前学習時との入力特徴量のギャップである．本実験では，HuBERTの事前学習に合わせてHuBERT中間特徴量をTransformerへの入力とした．しかし，事前学習時に用いられる，マスクベクトルおよび原音声から得られる特徴量が存在しなかったことがギャップとなり，転移学習の妨げになった可能性がある．

4.5 今後の課題

本実験を通して得られた今後の課題は2つある．

1つ目は，損失関数の重み係数$\lambda_{\text{HuB-disc}}$におけるグリッドサーチである．本実験では5種類の値によるグリッドサーチを行い，客観評価の結果から，検討した手法の性能は$\lambda_{\text{HuB-disc}}$に大きく左右されることがわかった．これに対し，各タスクに対する損失関数の勾配のノルムを求め，損失の下がり具合によって勾配のノルムを調整するように重み係数を最適化するGradNorm[47]や，損失の下がり具合のみを用いて重み係数を変化させるDynamic Weight Averaging[48]がある．これらのように，重み係数を動的に調節し，グリッドサーチよりも柔軟なチューニングを可能にすることで，モデルのさらなる精度改善および，実験時間の短縮が期待できる．

2つ目は，HuBERT Transformerの転移学習方法である．本実験では，事前学習時との入力特徴量のギャップが大きかったことが課題の1つとして考えられる．これに対し，学習初期にあえて原音声から得られる特徴量を混入させて事前学習時の条件に近づけ，徐々に混入率を下げてネットワークAによる予測結果のみを入力とするようスケジューリングする方法が考えられる．混入率を下げて事前学習時から徐々に離していくことは，学習難易度が徐々に高まっていくことに相当する．このような方法はカリキュラム学習と呼ばれ，難易度を下げた場合において収束した局所最適解が，その後難易度を上げた場合における良い初期値となることで，初めから難易度の高い学習を行うよりも良い局所最適解に収束させられる可能性がある[49]．

5　結論

本研究では,「AVHuBERT を利用したメルスペクトログラムと HuBERT 離散特徴量を予測対象とするマルチタスク学習手法」をベースラインとして採用し，この手法を上回る新たなモデルを提案することを目的とした．これに対し，HuBERT の転移学習による改善を狙った，ネットワーク A，ネットワーク B，ボコーダの 3 つから構成されるモデルを提案した．その結果，客観評価と主観評価の両面において，提案手法がベースラインを上回る明瞭性および類似性を達成したことが示された．一方，いまだ分析合成や原音声との差は大きいことがわかり，さらなる改善が必要であることも明らかとなった．今後の課題として，損失関数の重み係数に対するグリッドサーチからの脱却や，HuBERT の転移学習方法の検討が挙げられる．

謝辞

本研究に取り組む上で熱心に指導してくださり，計算環境の整備やクラウドワークスの活用など，新しい取り組みにも前向きにご協力くださった鏑木先生に心より感謝申し上げます.

また，日頃から多くの相談に乗ってくださり，研究に対する助言をいただいた藤田さんをはじめ，ゼミや中間発表において貴重なご意見をくださった河原先生，吉永先生，そして研究室の皆様にも厚く御礼申し上げます.

参考文献

[1] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

[2] KR Prajwal, Rudrabha Mukhopadhyay, Vinay P Namboodiri, and CV Jawahar. Learning individual speaking styles for accurate lip to speech synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13796–13805, 2020.

[3] Minsu Kim, Joanna Hong, and Yong Man Ro. Lip to speech synthesis with visual context attentional gan. *Advances in Neural Information Processing Systems*, Vol. 34, pp. 2758–2770, 2021.

[4] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[5] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[6] Rodrigo Mira, Alexandros Haliassos, Stavros Petridis, Björn W Schuller, and Maja Pantic. Svts: scalable video-to-speech synthesis. *arXiv preprint arXiv:2205.02058*, 2022.

[7] Minsu Kim, Joanna Hong, and Yong Man Ro. Lip-to-speech synthesis in the wild with multi-task learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

[8] Jeongsoo Choi, Minsu Kim, and Yong Man Ro. Intelligible lip-to-speech synthesis with speech units. *arXiv preprint arXiv:2305.19603*, 2023.

[9] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, Vol. 29, pp. 3451–3460, 2021.

[10] Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. Learning audio-visual speech representation by masked multimodal cluster prediction. *arXiv preprint arXiv:2201.02184*, 2022.

[11] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. Lrs3-ted: a large-scale dataset for visual speech recognition. *arXiv preprint arXiv:1809.00496*, 2018.

[12] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30, p. 3. Atlanta, GA, 2013.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[16] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[17] Kyunghyun Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[18] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[19] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[20] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, Vol. 29, , 2016.

[21] Jiawei Zhang. Gradient descent based optimization algorithms for deep learning models training. *arXiv preprint arXiv:1903.03614*, 2019.

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[23] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[25] Catherine F Higham and Desmond J Higham. Deep learning: An introduction for applied mathematicians. *Siam review*, Vol. 61, No. 4, pp. 860–891, 2019.

[26] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4879–4883. IEEE, 2018.

[27] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, Vol. 33, pp. 17022–17033, 2020.

[28] 田口史郎. "深層学習を用いたデータ駆動型調音・音声間変換に関する研究". 九州大学大学院芸術工学府芸術工学専攻 博士論文, 2021.

[29] 江崎蓮. "深層学習を用いた口唇動画・音声変換に関する調査". 九州大学大学院芸術工学府芸術工学専攻 修士論文, 2022.

[30] Yoshinori Sagisaka, Kazuya Takeda, M Abel, Shigeru Katagiri, Tetsuo Umeda, and Hisao Kuwabara. A large-scale japanese speech database. In *ICSLP*, pp. 1089–1092, 1990.

[31] T Okamoto, Y Shiga, and H Kawai. Hi-fi-captain: High-fidelity and high-capacity conversational speech synthesis corpus developed by nict, 2023.

[32] Shinnosuke Takamichi, Kentaro Mitsui, Yuki Saito, Tomoki Koriyama, Naoko Tanji, and Hiroshi Saruwatari. Jvs corpus: free japanese multi-speaker voice corpus. *arXiv preprint arXiv:1908.06248*, 2019.

[33] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *Proceedings of the IEEE international conference on computer vision*, pp. 1021–1030, 2017.

[34] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, Vol. 31, , 2018.

[35] Ankita Pasad, Bowen Shi, and Karen Livescu. Comparative layer-wise analysis of self-supervised speech models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

[36] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. Voxceleb: Large-scale speaker verification in the wild. *Computer Speech & Language*, Vol. 60, p. 101027, 2020.

[37] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

[38] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, Vol. 31, , 2018.

[39] Bowen Shi, Wei-Ning Hsu, and Abdelrahman Mohamed. Robust self-supervised audio-visual speech recognition. *arXiv preprint arXiv:2201.01763*, 2022.

[40] David Snyder, Guoguo Chen, and Daniel Povey. Musan: A music, speech, and noise corpus. *arXiv preprint arXiv:1510.08484*, 2015.

[41] Triantafyllos Afouras, Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Deep audio-visual speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 44, No. 12, pp. 8717–8727, 2018.

[42] Yue Yin, Daijiro Mori, and Seiji Fujimoto. Reazonspeech: A free and massive corpus for japanese asr. In *Annual meetings of the Association for Natural Language Processing*, 2023.

[43] Kei Sawada, Tianyu Zhao, Makoto Shing, Kentaro Mitsui, Akio Kaga, Yukiya Hono, Toshiaki Wakatsuki, and Koh Mitsuda. Release of pre-trained models for the Japanese language. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 13898–13905, 5 2024. Available at: https://arxiv.org/abs/2404.01657.

[44] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pp. 28492–28518. PMLR, 2023.

[45] Ambika Kirkland, Shivam Mehta, Harm Lameris, Gustav Eje Henter, Eva Székely, and Joakim Gustafson. Stuck in the mos pit: A critical analysis of mos test methodology in tts evaluation. In *12th Speech Synthesis Workshop (SSW) 2023*, 2023.

[46] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[47] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

[48] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1871–1880, 2019.

[49] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 44, No. 9, pp. 4555–4576, 2021.