2024年度後期　修士論文

深層学習による口唇音声変換に関する研究

A Study on the Conversion from Lip-Video to Speech Using a Deep Learning Technique

2025年月日

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

目 次

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が，自然で聞き取りやすい音声を生成することが難しかったり，習得に訓練を必要としたり，器具の交換のための定期的な手術を必要とするといった課題がある．これに対して本研究では，新たな代用音声手法として，深層学習を活用して口唇の動きと音声波形の関係性を学習することで，口唇の動きから音声を合成する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．

近年の動画音声合成では，LRS3[1] や VoxCeleb2[2] などの大規模データセットを活用して構築された，自己教師あり学習（Self Supervised Learning; SSL）モデルを FineTuning することの有効性が示されている．特に，近年の動画音声合成においては AVHuBERT[3] というモデルが動画からの特徴抽出器として採用される場合が多い．AVHuBERT は，動画と音声の対応関係を Masked Prediction という自己教師あり学習方法により捉えたモデルであり，Modality Dropout という工夫により，自己教師あり学習時には動画と音声の両方を入力とするものの，FineTuning 時には動画あるいは音声のみを入力とできる柔軟性を持ち合わせている．これに加え，従来の動画音声においては動画からの予測対象としてメルスペクトログラムが選択されることが多かったが，近年ではこれにテキストや，音声 SSL モデルである HuBERT を利用して得られた離散特徴量を合わせて予測対象とする，マルチタスク学習の有効性が示されている [4, 5]．

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が困難である点を課題とする．この課題に対し，近年高い精度を達成した手法 [5] における，「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，この性能を上回る新たなモデルを提案することを目的とする．

1.3 本論文の構成

1.4 本論文で用いる演算子

記号	説明
$\odot$	要素積
$[\cdot, \ldots, \cdot]$	特徴量の次元方向の結合
$(\boldsymbol{x})_N$	ベクトル $\boldsymbol{x}$ を N 回列方向に並べる
one-hot	インデックス系列を One-hot ベクトルに変換
argmax	特徴量の次元方向に、最大の値を取るインデックスに変換

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

音声の時間波形に対して，周波数領域での情報を得るためにはフーリエ変換（Fourier Transform）を使用する．特に，音声は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ことが多い．この時，音声はアナログ信号ではなく，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たデジタル信号として扱われる．このような場合，離散信号に対してのフーリエ変換である離散フーリエ変換（discrete Fourier transform; DFT）が用いられる．また，信号の系列長をゼロパディングして 2 の冪乗の長さに調整することで，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 $X[k]$ とする．ここで，n はサンプルのインデックス，k は周波数インデックスである．$X[k]$ は複素数であり，以下のように極座標表示することができる．

$$X[k] = \mathrm{Re}\left(X[k]\right) + j\mathrm{Im}\left(X[k]\right) \tag{2.1}$$

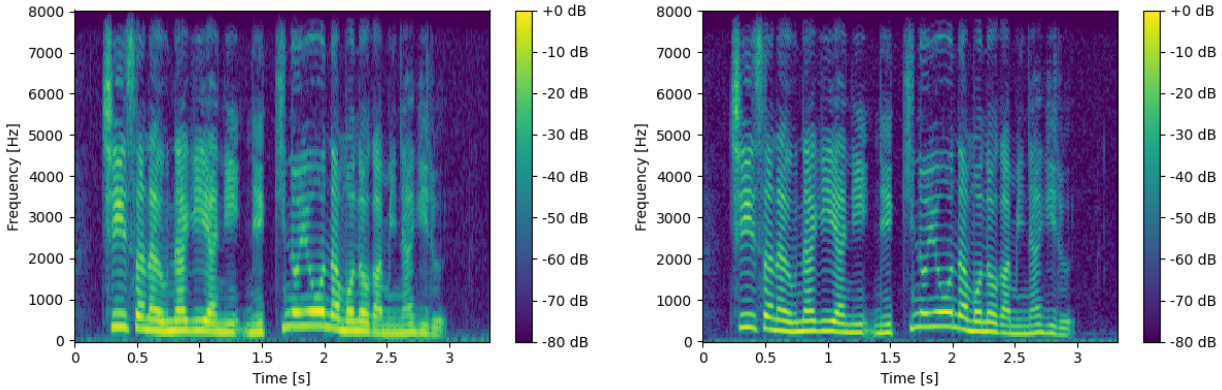
$$= |X[k]|e^{j\angle \mathrm{X[k]}} \tag{2.2}$$

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

$$|X[k]| = \sqrt{\mathrm{Re}\left(X[k]\right)^2 + \mathrm{Im}\left(X[k]\right)^2} \tag{2.3}$$

$$\angle \mathrm{X[k]} = \tan^{-1}\frac{\mathrm{Im}\left(X[k]\right)}{\mathrm{Re}\left(X[k]\right)} \tag{2.4}$$

と表される．また，$|X[k]|^2$ 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 には時間分解能と



(a) 窓長 12.5ms，シフト幅 5ms

(b) 窓長 25ms，シフト幅 10ms

(c) 窓長 50ms，シフト幅 20ms

(d) 窓長 100ms，シフト幅 40ms

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

周波数分解能の間に不確定性が存在し，両者の間にトレード・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 j，周波数インデックスを k として $X[j,k]$ と表すと，$X[j,k]$ 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，$|X[j,k]|$ を振幅スペクトログラム，$\angle \mathrm{X[j,k]}$ を位相スペクトログラム，$|X[j,k]|^2$ をパワースペクトログラムと呼ぶ.「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 100ms 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 12.5ms 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とトレード・オフであり，窓長 25ms や 50ms が程よいパラメータであることがわかる．

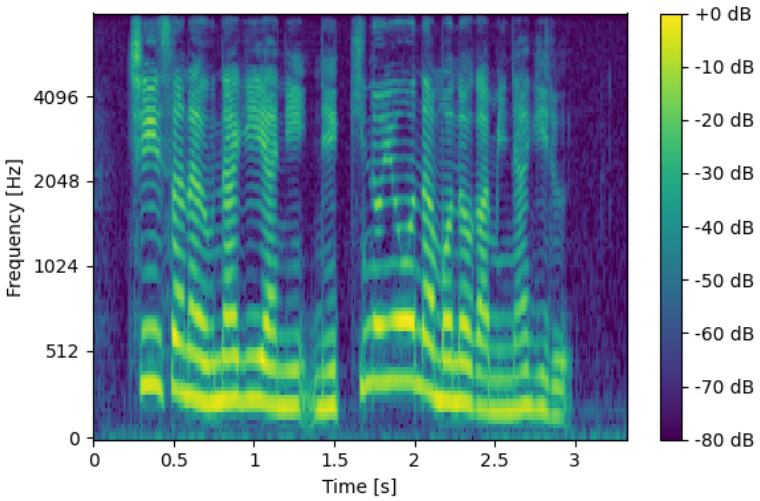


図 2.2: 「小さな鰻屋に，熱気のようなものがみなぎる」と発話した音声に対する対数メルスペクトログラム

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を人間の聴感特性を考慮したメル尺度に変換することによって得られる．周波数軸をメル尺度に変換する際，以下の式を用いる．

$$\mathrm{Mel}\,(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.5}$$

メル尺度は，1000 Hz，40 dB の純音を 1000 mel とする比率尺度である．メル尺度を用いることにより，低い周波数ほど細かく，高い周波数ほど荒い特徴量になる．メルスペクトログラムは，パワースペクトログラムに対してメルフィルタバンクを適用することによって得られる．メルフィルタバンクの数は任意に決定できるパラメータであり，メルスペクトログラムの周波数方向の次元はこれに一致する．音声合成においては，音声のサンプリング周波数を 16 kHz とするとき，メルフィルタバンクの数を 80 とし，8000 Hz までの帯域に対して適用することが多い．「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し，窓関数にハニング窓を用い，窓長 25ms，シフト幅 10ms としてパワースペクトログラムを計算した上で，80 次元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 2.2に示す．

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，音声認識や音声合成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，

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{3.1}$$

で与えられる．ここで，$\boldsymbol{W} \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$ は重み，$\boldsymbol{b} \in \mathbb{R}^{D_{\text{out}}}$ はバイアスである．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．

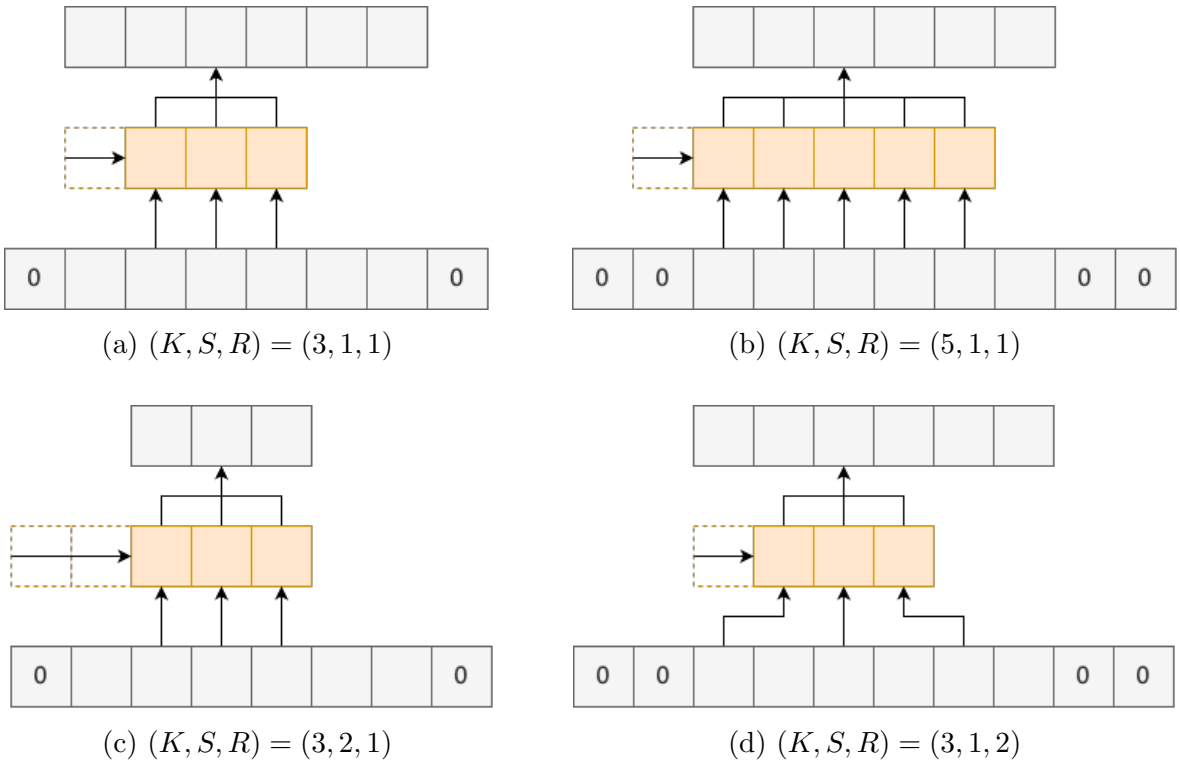
3.1.2　畳み込み層

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．一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．このとき，$y_{d_{\text{out}},t}$ は，

$$y_{d_{\text{out}},t} = b_{d_{\text{out}}} + \sum_{d_{\text{in}}=1}^{D_{\text{in}}} \sum_{k=1}^{K} x_{d_{\text{in}},t-\left\lfloor \frac{K}{2} \right\rfloor+k} w_{d_{\text{in}},d_{\text{out}},k} \tag{3.2}$$

で与えられる．ここで，$K \in \mathbb{N}$ はカーネルサイズ，$w_{d_{\text{in}},d_{\text{out}},k} \in \mathbb{R}$ 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，$b_{d_{\text{out}}} \in \mathbb{R}$ 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，一次元畳み込み層の 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．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．一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．ここで，D は次元，T は系列長である．

これに加えて，カーネルを二次元配列とすれば二次元畳み込み層，三次元配列とすれば三次元畳み込み層となる．二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 は幅に相当する．三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．



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

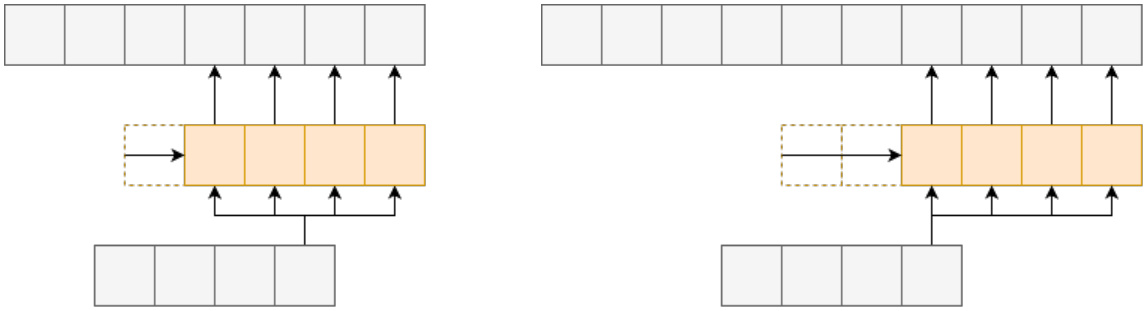
(c) $(K, S, R) = (3, 2, 1)$　(d) $(K, S, R) = (3, 1, 2)$

図 3.1: ある次元における一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．

畳み込み層における主要なパラメータは三つある．一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．二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．三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．例えば，カーネルサイズを 3，ストライドとダイレーションを 1 とした場合には，入力の両端に 1 ずつゼロパディングすれば良い．図 3.1に，ある次元における一次元畳み込み層の処理を示す．

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一次元転置畳み込み層の処理を示す．一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



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

図 3.2: ある次元における一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．

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．

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，代表的なものを六つ述べる．また，本節で取り上げる活性化関数とその一階導関数のグラフを図 3.3に示す．

一つ目は，シグモイド関数である．シグモイド関数は

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.3}$$

で与えられ，その一階導関数は

$$\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x} = \frac{\exp(-x)}{(1 + \exp(-x))^2} \tag{3.4}$$

となる．図 3.3bより，シグモイド関数の一階導関数の最大値は $x = 0$ における 0.25 であり，$|x - 0|$ が大きくなるのに伴ってそ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，シグモイド関数の一階導関数の値が乗算された結果となる．前述したように，シグモイド関数は一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．

二つ目は，tanh 関数である．tanh は

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{3.5}$$

で与えられ，その一階導関数は

$$\frac{\mathrm{d}\tanh(x)}{\mathrm{d}x} = \frac{4}{(\exp(x)+\exp(-x))^2} \tag{3.6}$$

$$= \frac{1}{\cosh(x)^2} \tag{3.7}$$

となる．tanh の値域は $[-1,1]$ となっており，図 3.3b より $|x-0|$ が小さいところではシグモイド関数より一階導関数の値が大きくなっていることがわかる．しかし，$|x-0|$ が大きくなればシグモイド関数と同様に一回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．

三つ目は，ReLU である．ReLU は

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.8}$$

で与えられ，その一階導関数は

$$\frac{\mathrm{dReLU}(x)}{\mathrm{d}x} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x <= 0 \end{cases} \tag{3.9}$$

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．一階導関数は 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微分係数は 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．

四つ目は，LeakyReLU[6] である．LeakyReLU は

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x <= 0 \end{cases} \tag{3.10}$$

で与えられ，その一階導関数は

$$\frac{\mathrm{dLeakyReLU}(x)}{\mathrm{d}x} = \begin{cases} 1 & \text{if } x > 0 \\ a & \text{if } x <= 0 \end{cases} \tag{3.11}$$

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 $\mathrm{dLeakyReLU}(0)/\mathrm{d}x = a$ 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，一階導関数も 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．

五つ目は，PReLU[7] 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．

六つ目は，GELU[8] である．GELU は

$$\mathrm{GELU}\left(x\right) = x\Phi\left(x\right) \tag{3.12}$$

で与えられる．ここで，

$$\Phi\left(x\right) = P\left(X \leq x\right),\ X \sim \mathcal{N}\left(0, 1\right) \tag{3.13}$$

である．GELU の一階導関数は，

$$\frac{\mathrm{dGELU}\left(x\right)}{\mathrm{d}x} = \Phi\left(x\right) + \frac{x}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \tag{3.14}$$

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．実際，$m \sim \mathrm{Bernoulli}\left(\Phi\left(x\right)\right)$ とすると，

$$\begin{aligned}\mathrm{GELU}\left(x\right) &= x\Phi\left(x\right) &(3.15)\\ &= 1x \cdot \Phi\left(x\right) + 0x \cdot \left(1 - \Phi\left(x\right)\right) &(3.16)\\ &= \mathbb{E}\left[mx\right] &(3.17)\end{aligned}$$

となり，GELU の出力は確率的なバイナリマスク m を入力 x にかけた，mx の期待値に等しいことがわかる．

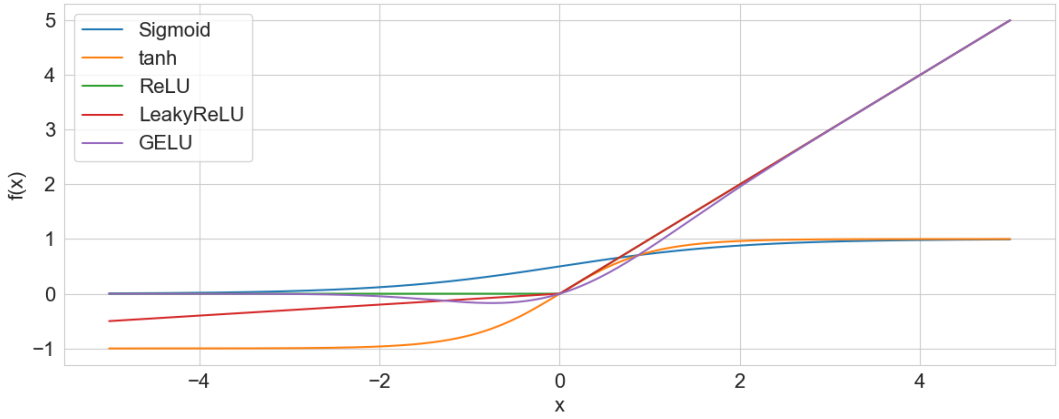
3.1.5 再帰型ニューラルネットワーク

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．

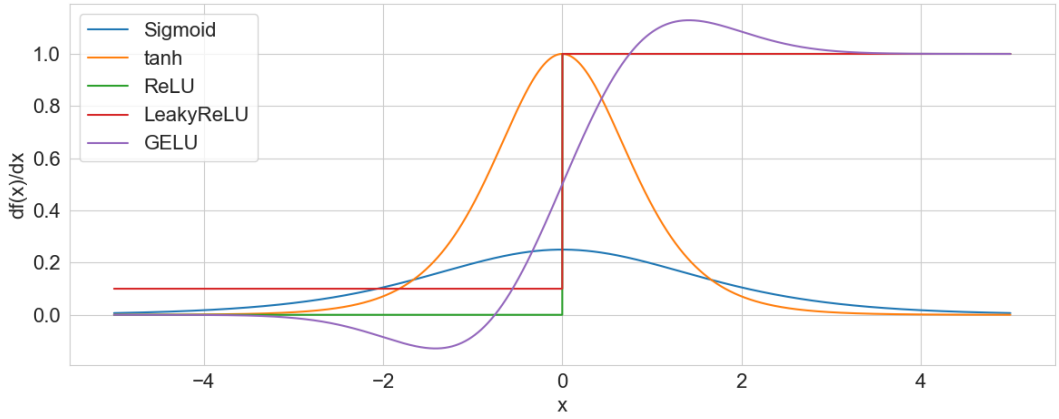
近年よく用いられる RNN として，長・短期記憶（Long Short-Time Memory; LSTM）[9] がある．LSTM は入力ゲート，忘却ゲート，出力ゲートの 3 つを持ち，これらゲートによってネットワーク内部の情報の取捨選択を行うことで，長い系列データからの学習を可能にした．LSTM のネットワーク内部で行われる計算を以下に示す．

$$\begin{aligned}\boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_f\left[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}\right] + \boldsymbol{b}_f\right) &(3.18)\\ \boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_i\left[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}\right] + \boldsymbol{b}_i\right) &(3.19)\\ \tilde{\boldsymbol{c}}_t &= \tanh\left(\boldsymbol{W}_h\left[\boldsymbol{x}_t, \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\right]\right) &(3.20)\\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t &(3.21)\\ \boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_o\left[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}\right] + \boldsymbol{b}_o\right) &(3.22)\\ \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right) &(3.23)\end{aligned}$$

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，$\boldsymbol{c}_t \in [-1,1]^{D_{\mathrm{out}}}$ が時刻 t におけるセルの状態，$\boldsymbol{o}_t \in [0,1]^{D_{\mathrm{out}}}$ が出力ゲートの出



(a) 活性化関数



(b) 活性化関数の一階導関数

図 3.3: 活性化関数の例

力，$\boldsymbol{h}_t \in [-1,1]^{D_{\text{out}}}$ が時刻 t における隠れ状態を表す．$\boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_h, \boldsymbol{W}_o \in \mathbb{R}^{D_{\text{out}} \times (D_{\text{in}}+D_{\text{out}})}$ は重み，$\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_h, \boldsymbol{b}_o \in \mathbb{R}^{D_{\text{out}}}$ はバイアスである．忘却ゲート出力 $\boldsymbol{f}_t$ が前時刻のセル状態 $\boldsymbol{c}_t$ 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）[10] 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．

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_z \left[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}\right] + \boldsymbol{b}_z\right) \tag{3.24}$$

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_r \left[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}\right] + \boldsymbol{b}_r\right) \tag{3.25}$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{W}_h \left[\boldsymbol{x}_t, \boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}\right] + \boldsymbol{b}_h\right) \tag{3.26}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \tag{3.27}$$

ここで，$\boldsymbol{x}_t \in \mathbb{R}^{D_{\text{in}}}$ が時刻 t における入力，$\boldsymbol{z}_t \in [0,1]^{D_{\text{out}}}$ が更新ゲートの出力，$\boldsymbol{r}_t \in [0,1]^{D_{\text{out}}}$ がリセットゲートの出力，$\boldsymbol{h}_t \in [-1,1]^{D_{\text{out}}}$ が時刻 t における隠れ状態を表す．また，$\boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h \in \mathbb{R}^{D_{\text{out}} \times (D_{\text{in}}+D_{\text{out}})}$ は重み，$\boldsymbol{b}_z, \boldsymbol{b}_r, \boldsymbol{b}_h \in \mathbb{R}^{D_{\text{out}}}$ はバイアスである．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）[11] 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

$$\mu_d^B = \frac{1}{N} \sum_{n=1}^{N} x_{n,d} \tag{3.28}$$

$$\left(\sigma_d^B\right)^2 = \frac{1}{N} \sum_{n=1}^{N} \left(x_{n,d} - \mu_d^B\right)^2 \tag{3.29}$$

$$\tilde{x}_{n,d} = \frac{x_{n,d} - \mu_d^B}{\sqrt{\left(\sigma_d^B\right)^2 + \epsilon}} \tag{3.30}$$

$$y_{n,d} = \gamma_d \tilde{x}_{n,d} + \beta_d \tag{3.31}$$

で与えられる．ここで，$\gamma_d, \beta_d \in \mathbb{R}$ は学習可能なスカラーであり，$\epsilon \in \mathbb{R}$ 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

$$\gamma_d = \sqrt{\left(\sigma_d^B\right)^2 + \epsilon} \tag{3.32}$$

$$\beta_d = \mu_d^B \tag{3.33}$$

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ことで，確率的な挙動を持たないよう工夫されている．

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）[12] 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

$$\mu_n^L = \frac{1}{D}\sum_{d=1}^{D} x_{n,d} \tag{3.34}$$

$$\left(\sigma_n^L\right)^2 = \frac{1}{D}\sum_{d=1}^{D}\left(x_{n,d} - \mu_n^L\right)^2 \tag{3.35}$$

$$\tilde{x}_{n,d} = \frac{x_{n,d} - \mu_n^L}{\sqrt{\left(\sigma_n^L\right)^2 + \epsilon}} \tag{3.36}$$

$$y_{n,d} = \gamma_d \tilde{x}_{n,d} + \beta_d \tag{3.37}$$

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）[13] がある．これは，ある層の重みベクトル $\boldsymbol{w}$ を，

$$\boldsymbol{w} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} g \tag{3.38}$$

のように単位ベクトル $\boldsymbol{v}/\|\boldsymbol{v}\|$（ベクトルの向き）とスカラー g（ベクトルの大きさ）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[14] 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 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．また，Transformer 層の構造を図 3.4に示す．

まず，Transformer 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，

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q + \left((\boldsymbol{b}_Q)_T\right)^\top \tag{3.39}$$

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K + \left((\boldsymbol{b}_K)_T\right)^\top \tag{3.40}$$

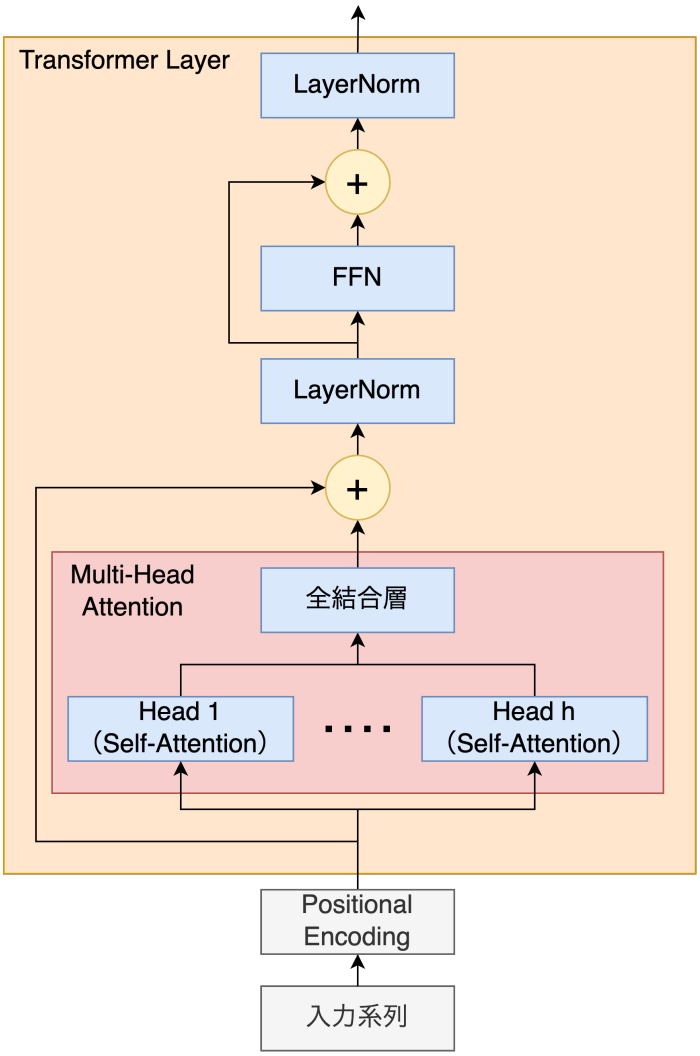


図 3.4: Transformer 層の構造

$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V + ((\boldsymbol{b}_V)_T)^\top \tag{3.41}$$

で与えられる．ここで，$\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{D_{\text{model}} \times D_k}, \boldsymbol{W}_V \in \mathbb{R}^{D_{\text{model}} \times D_v}$ は重み，$\boldsymbol{b}_Q, \boldsymbol{b}_K \in \mathbb{R}^{D_k}, \boldsymbol{b}_V \in \mathbb{R}^{D_v}$ はバイアスである．また，$(\boldsymbol{b}_Q)_T \in \mathbb{R}^{D_k \times T}$ は，列ベクトル $\boldsymbol{b}_Q$ を T 回列方向に並べる操作を表す．

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，

$$\text{Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D_k}}\right)\boldsymbol{V} \tag{3.42}$$

で与えられる．softmax 関数は行方向に適用されるため，$\text{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{D_k}\right)$ 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 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．ヘッド数を H とすると，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 $\boldsymbol{V}^h \in \mathbb{R}^{T \times \frac{D_v}{H}}$ の計算は,

$$\boldsymbol{Q}^h = \boldsymbol{X}\boldsymbol{W}_Q^h + \left(\left(\boldsymbol{b}_Q^h\right)_T\right)^\top \tag{3.43}$$

$$\boldsymbol{K}^h = \boldsymbol{X}\boldsymbol{W}_K^h + \left(\left(\boldsymbol{b}_K^h\right)_T\right)^\top \tag{3.44}$$

$$\boldsymbol{V}^h = \boldsymbol{X}\boldsymbol{W}_V^h + \left(\left(\boldsymbol{b}_V^h\right)_T\right)^\top \tag{3.45}$$

で与えられる．ここで，$\boldsymbol{W}_Q^h, \boldsymbol{W}_K^h \in \mathbb{R}^{D_{\text{model}} \times \frac{D_k}{H}}$，$\boldsymbol{W}_V^h \in \mathbb{R}^{D_{\text{model}} \times \frac{D_v}{H}}$ は重み，$\boldsymbol{b}_Q^h, \boldsymbol{b}_K^h \in \mathbb{R}^{\frac{D_k}{H}}, \boldsymbol{b}_V^h \in \mathbb{R}^{\frac{D_v}{H}}$ はバイアスである．この後の処理も Self-Attention と同様に,

$$\text{Attention}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right) = \text{softmax}\left(\frac{\boldsymbol{Q}^h\left(\boldsymbol{K}^h\right)^\top}{\sqrt{D_k}/H}\right)\boldsymbol{V}^h \tag{3.46}$$

となる．ここで得られた各ヘッドからの出力を結合し，さらに全結合層を通すことで Multi-Head Attetion の出力が得られる．すなわち,

$$\text{MultiHead}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \left[\text{Attention}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right), \ldots, \text{Attention}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right)\right]\boldsymbol{W}_o + \left(\left(\boldsymbol{b}_o\right)_T\right)^\top \tag{3.47}$$

と表される．ここで，$\boldsymbol{W}_o \in \mathbb{R}^{D_v \times D_{\text{model}}}$ は重み，$\boldsymbol{b}_o \in \mathbb{R}^{D_{\text{model}}}$ はバイアスである．ヘッドを分割して複数パターンの Self-Attention を可能にすることで，より複雑な入出力の関係を学習できると考えられる．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,

$$\boldsymbol{Y} = \text{LayerNorm}\left(\text{MultiHead}r\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} + \boldsymbol{X}\right) \tag{3.48}$$

で与えられる．その後，全結合層を通し，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,

$$\boldsymbol{Y} = \text{LayerNorm}\left(\left(\text{ReLU}\left(\boldsymbol{Y}\boldsymbol{W}_1 + \left(\left(\boldsymbol{b}_1\right)_T\right)^\top\right)\boldsymbol{W}_2 + \left(\left(\boldsymbol{b}_2\right)_T\right)^\top\right) + \boldsymbol{Y}\right) \tag{3.49}$$

となる．ここで，$\boldsymbol{W}_1 \in \mathbb{R}^{D_{\text{model}} \times D_{\text{fc}}}$，$\boldsymbol{W}_2 \in \mathbb{R}^{D_{\text{fc}} \times D_{\text{model}}}$ は重み，$\boldsymbol{b}_1 \in \mathbb{R}^{D_{\text{fc}}}$，$\boldsymbol{b}_2 \in \mathbb{R}^{D_{\text{model}}}$ はバイアスである．D_{fc} は D_{model} の 4 倍とされることが多い．Transformer 全体は，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一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,

$$\text{PositionalEncoding}\left(t, d\right) = \begin{cases} \sin\left(\frac{t}{10000^{2d/D_{\text{model}}}}\right) & \text{if } d \bmod 2 = 0 \\ \cos\left(\frac{t}{10000^{2d/D_{\text{model}}}}\right) & \text{if } d \bmod 2 = 1 \end{cases} \tag{3.50}$$

で与えられる．

3.2 学習方法

本節において，$\boldsymbol{\theta} \in \mathbb{R}^{N_{\text{model}}}$ はDNNの重みとバイアスをまとめて表す変数とする．また，文章中では簡潔さを優先し，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によって推定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一つに，MAE（Mean Absolute Error）Lossがある．推定対象を $\boldsymbol{y} \in \mathbb{R}^D$，DNNによる推定結果を $\hat{\boldsymbol{y}} \in \mathbb{R}^D$ とすると，MAE Lossは

$$L_{\text{MAE}}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) = \frac{1}{D}\sum_{d=1}^{D}\left|y_d - \hat{y}_d\right| \tag{3.51}$$

で与えられる．また，推定対象が行列 $\boldsymbol{Y} \in \mathbb{R}^{D\times T}$ の場合，DNNによる推定結果を $\hat{\boldsymbol{Y}} \in \mathbb{R}^{D\times T}$ とすると，

$$L_{\text{MAE}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = \frac{1}{DT}\sum_{d=1}^{D}\sum_{t=1}^{T}\left|y_{d,t} - \hat{y}_{d,t}\right| \tag{3.52}$$

で与えられる.

一方，分類問題において用いられる関数の一つに，Cross Entropy Lossがある．C クラス分類の問題について，推定対象を $\boldsymbol{y} \in [0,1]^C$，DNNによる推定結果を $\hat{\boldsymbol{y}} \in \mathbb{R}^C$ とすると，Cross Entropy Lossは

$$L_{\text{CE}}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) = -\sum_{c=1}^{C} y_c \log\frac{\exp\left(\hat{y}_c\right)}{\sum_{i=1}^{C}\exp\left(\hat{y}_c\right)} \tag{3.53}$$

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，

$$\sum_{c=1}^{C} y_c = 1 \tag{3.54}$$

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，推定対象が行列 $\boldsymbol{Y} \in [0,1]^C$ の場合，DNNによる推定結果を $\hat{\boldsymbol{Y}} \in \mathbb{R}^{C\times T}$ とすると，

$$L_{\text{CE}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C} y_{c,t} \log\frac{\exp\left(\hat{y}_{c,t}\right)}{\sum_{i=1}^{C}\exp\left(\hat{y}_{c,t}\right)} \tag{3.55}$$

で与えられる.

3.2.2 勾配降下法

勾配降下法（Gradient Descent）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 $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$ とする．各 n に対し，$\boldsymbol{x}_n \in \mathbb{R}^{D_{\text{in}}}$ は DNN への入力, $\boldsymbol{y}_n \in \mathbb{R}^{D_{\text{out}}}$ は予測対象を表す．DNN は $f : \mathbb{R}^{D_{\text{in}}} \to \mathbb{R}^{D_{\text{out}}}$ で表し，予測値を $\hat{\boldsymbol{y}}_n = f(\boldsymbol{x}_n; \boldsymbol{\theta})$ とする．損失関数は $L : \mathbb{R}^{D_{\text{out}}} \times \mathbb{R}^{D_{\text{out}}} \to \mathbb{R}$ とし，$\mathcal{L} : \mathbb{R}^{N_{\text{model}}} \to \mathbb{R}$ を，

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} L(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} L(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})) \tag{3.56}$$

で定義する．ここで，$\mathcal{I} \subset \{1, \dots, N\}$ は各サンプルに対するインデックスの部分集合，$|\cdot|$ は集合の元の数を表す．すなわち，$\mathcal{L}$ は学習データ $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i \in I} \subset \mathcal{D}_{\text{train}}$ に対する損失を，DNN の重み $\boldsymbol{\theta}$ の関数として扱うものである．上の表記を用いれば，DNN の最適化問題は

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{N_{\text{model}}}} \mathcal{L}(\boldsymbol{\theta}; \{1, \dots, N\}) \tag{3.57}$$

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，

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}) \tag{3.58}$$

で与えられる．ここで，$\tau^{\text{iter}} \in \mathbb{N}$ は学習におけるイテレーション，$\eta \in [0, \infty)$ は学習率を表す．

勾配降下法には，三種類の方法がある [15]．一つ目は，バッチ勾配降下法である．これは，

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \{1, \dots, N\}) \tag{3.59}$$

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．二つ目は，確率的勾配降下法である．これは，ランダムに選択された $n_{\tau^{\text{iter}}} \in \{1, \dots, N\}$ に対し，

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \{n_{\tau^{\text{iter}}}\}) \tag{3.60}$$

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．三つ目は，ミニバッチ勾配降下法である．これは，$1 < |\mathcal{I}_{\tau^{\text{iter}}}| < N$ を満たすランダムに選択された $\mathcal{I}_{\tau^{\text{iter}}} \subsetneq \{1, \dots, N\}$ に対し，

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}) \tag{3.61}$$

で与えられる．これより，各イテレーションで二つ以上のサンプルからなる学習データセットの真部分集合を用いる方法だと言える．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一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，具体的な方法を四つ述べる．

一つ目は，L2 正則化である．これは，$\mathcal{L}(\boldsymbol{\theta};\mathcal{I})$ を

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{I}) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right)\right) + \frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2 \tag{3.62}$$

とすることで与えられる．ここで，$\lambda \in [0,\infty)$ 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 $\|\boldsymbol{\theta}\|_2^2$ 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．

二つ目は，Weight Decay である．これは，重みの更新式を

$$\boldsymbol{\theta}_{\tau^{\mathrm{iter}}} = \boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1};\mathcal{I}_{\tau^{\mathrm{iter}}}\right) - \lambda'\boldsymbol{\theta} \tag{3.63}$$

とすることで与えられる．ここで，$\lambda' \in [0,\infty)$ 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．

三つ目は，Dropout[16] 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 $\boldsymbol{x} \in \mathbb{R}^{D_{\mathrm{in}}}$，特徴量を 0 に落とす確率を $p \in (0,1)$ とすると，学習時の Dropout 出力 $\boldsymbol{y}_{\mathrm{train}} \in \mathbb{R}^{D_{\mathrm{out}}}$ および推論時の Dropout 出力 $\boldsymbol{y}_{\mathrm{infer}} \in \mathbb{R}^{D_{\mathrm{out}}}$ は，

$$\boldsymbol{y}_{\mathrm{train}} = \frac{\boldsymbol{x}\odot\boldsymbol{m}}{1-p} \tag{3.64}$$

$$\boldsymbol{y}_{\mathrm{infer}} = \boldsymbol{x} \tag{3.65}$$

で与えられる．ここで，$\boldsymbol{m} \in \{0,1\}^{D_{\mathrm{in}}}$ 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 (3.64) 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 $\boldsymbol{y}_{\mathrm{train}}$ の期待値をとれば，

$$\mathbb{E}\left[\boldsymbol{y}_{\mathrm{train}}\right] = \mathbb{E}\left[\frac{\boldsymbol{x}\odot\boldsymbol{m}}{1-p}\right] \tag{3.66}$$

$$= \frac{\boldsymbol{x}}{1-p}\odot\mathbb{E}\left[\boldsymbol{m}\right] \tag{3.67}$$

$$= \frac{\boldsymbol{x}}{1-p}\odot(1-p\ \cdots\ 1-p)^{\top} \tag{3.68}$$

$$= \boldsymbol{x} = \boldsymbol{y}_{\mathrm{infer}} \tag{3.69}$$

となる．Dropout は学習時に一部のニューロンを落とすことで，実質的に異なるネットワークを学習させていると考えることができる．よって，学習時の出力の期待値に推論時の出力を一

致させることは，異なるネットワークから得られた出力の期待値をとり，アンサンブルモデルとして推論時の出力を得ていると解釈できる．

四つ目は，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[17] がある．Adam の計算過程をアルゴリズム 1に示す．ここで，$\boldsymbol{g}_{\tau^{\mathrm{iter}}} \in \mathbb{R}^{N_{\mathrm{model}}}$ は勾配の一次モーメント，$\boldsymbol{m}_{\tau^{\mathrm{iter}}} \in \mathbb{R}^{N_{\mathrm{model}}}$ が勾配の一次モーメントの指数移動平均，$\boldsymbol{v}_{\tau^{\mathrm{iter}}} \in \mathbb{R}^{N_{\mathrm{model}}}$ が勾配の二次モーメントの指数移動平均である．$\hat{\boldsymbol{m}}_{\tau^{\mathrm{iter}}}$ および $\hat{\boldsymbol{v}}_{\tau^{\mathrm{iter}}}$ はそれぞれ，$\boldsymbol{m}_{\tau^{\mathrm{iter}}}$ および $\boldsymbol{v}_{\tau^{\mathrm{iter}}}$ 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．また，$\beta_1, \beta_2 \in [0, 1]$ が，指数移動平均の程度を調整するパラメータである．ここで，Adam では正則化として L2 正則化を採用したが，

Algorithm 1 Adam

Input: η, β_1, β_2, λ, $\boldsymbol{\theta}_0$, $L(\boldsymbol{\theta})$
$\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$
for $\tau^{\mathrm{iter}} = 1$ to $\ldots$ **do**
  $\boldsymbol{g}_{\tau^{\mathrm{iter}}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1}; \mathcal{I}_{\tau^{\mathrm{iter}}}\right) + \lambda \boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1}$
  $\boldsymbol{m}_{\tau^{\mathrm{iter}}} = \beta_1 \boldsymbol{m}_{\tau^{\mathrm{iter}}-1} + (1-\beta_1)\, \boldsymbol{g}_{\tau^{\mathrm{iter}}}$
  $\boldsymbol{v}_{\tau^{\mathrm{iter}}} = \beta_2 \boldsymbol{v}_{\tau^{\mathrm{iter}}-1} + (1-\beta_2)\, \boldsymbol{g}_{\tau^{\mathrm{iter}}} \odot \boldsymbol{g}_{\tau^{\mathrm{iter}}}$
  $\tilde{\boldsymbol{m}}_{\tau^{\mathrm{iter}}} = \boldsymbol{m}_{\tau^{\mathrm{iter}}} / \left(1 - \beta_1^{\tau^{\mathrm{iter}}}\right)$
  $\tilde{\boldsymbol{v}}_{\tau^{\mathrm{iter}}} = \boldsymbol{v}_{\tau^{\mathrm{iter}}} / \left(1 - \beta_2^{\tau^{\mathrm{iter}}}\right)$
  $\boldsymbol{\theta}_{\tau^{\mathrm{iter}}} = \boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1} - \eta \tilde{\boldsymbol{m}}_{\tau^{\mathrm{iter}}} / \left(\sqrt{\tilde{\boldsymbol{v}}_{\tau^{\mathrm{iter}}}} + \epsilon\right)$
end for
Return $\boldsymbol{\theta}_{\tau^{\mathrm{iter}}}$

Weight Decay を採用した最適化手法として AdamW[18] 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．

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，三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．

Algorithm 2 AdamW

Input: η, β_1, β_2, λ, $\boldsymbol{\theta}_0$, $L(\boldsymbol{\theta})$
$\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$
for $\tau^{\text{iter}} = 1$ to $\ldots$ **do**
  $\boldsymbol{g}_{\tau^{\text{iter}}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}\right)$
  $\boldsymbol{m}_{\tau^{\text{iter}}} = \beta_1 \boldsymbol{m}_{\tau^{\text{iter}}-1} + (1-\beta_1)\, \boldsymbol{g}_{\tau^{\text{iter}}}$
  $\boldsymbol{v}_{\tau^{\text{iter}}} = \beta_2 \boldsymbol{v}_{\tau^{\text{iter}}-1} + (1-\beta_2)\, \boldsymbol{g}_{\tau^{\text{iter}}} \odot \boldsymbol{g}_{\tau^{\text{iter}}}$
  $\tilde{\boldsymbol{m}}_{\tau^{\text{iter}}} = \boldsymbol{m}_{\tau^{\text{iter}}} / \left(1 - \beta_1^{\tau^{\text{iter}}}\right)$
  $\tilde{\boldsymbol{v}}_{\tau^{\text{iter}}} = \boldsymbol{v}_{\tau^{\text{iter}}} / \left(1 - \beta_2^{\tau^{\text{iter}}}\right)$
  $\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \tilde{\boldsymbol{m}}_{\tau^{\text{iter}}} / \left(\sqrt{\tilde{\boldsymbol{v}}_{\tau^{\text{iter}}}} + \epsilon\right) - \lambda \boldsymbol{\theta}$
end for
Return $\boldsymbol{\theta}_{\tau^{\text{iter}}}$

一つ目は，StepLRScheduler である．これは，初期学習率を η_0 として，エポック τ^{epoch} における学習率 $\eta_{\tau^{\text{epoch}}}$ を

$$\eta_{\tau^{\text{epoch}}} = \eta_0 \gamma^{\left\lfloor \tau^{\text{epoch}} / \text{stepSize} \right\rfloor} \tag{3.70}$$

で与えるスケジューラである．これは，学習が stepSize 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．

二つ目は，ExponentialLRScheduler である．これは，$\eta_{\tau^{\text{epoch}}}$ を

$$\eta_{\tau^{\text{epoch}}} = \eta_0 \exp\left(-\gamma \tau^{\text{epoch}}\right) \tag{3.71}$$

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．

三つ目は，Cosine Annealing with Warmup である．これは，$\eta_{\tau^{\text{epoch}}}$ を

$$\eta_{\tau^{\text{epoch}}} = \begin{cases} \eta_{\min} + \left(\frac{\tau^{\text{epoch}}}{\text{warmupSteps}}\right)(\eta_{\max} - \eta_{\min}) & \text{if } \tau^{\text{epoch}} < \text{warmupSteps} \\ \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{\left(\tau^{\text{epoch}} - \text{warmupSteps}\right)\pi}{\tau_{\max}^{\text{epoch}} - \text{warmupSteps}}\right)\right) & \text{if } \tau^{\text{epoch}} \geq \text{warmupSteps} \end{cases} \tag{3.72}$$

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，$\tau_{\max}^{\text{epoch}}$ は最大エポックである．warmupSteps 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 warmupSteps 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．

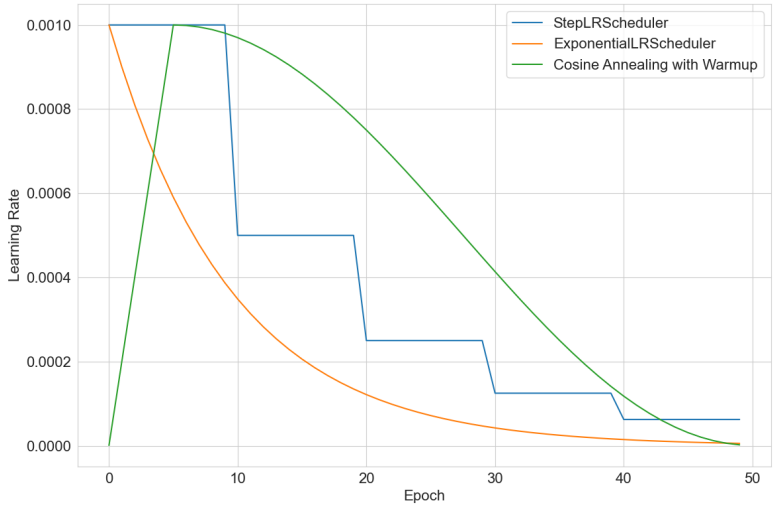


図 3.5: スケジューラによる学習率の変化

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 [19].

まず，$w_{p,q}^n \in \mathbb{R}$ 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，$b_p^n \in \mathbb{R}$ 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 番目のニューロンにおける出力 $a_p^n \in \mathbb{R}$ は，

$$a_p^n = b_p^n + \sum_{q=1}^{D_{n-1}} w_{q,p}^n o_q^{n-1} \tag{3.73}$$

で与えられる．ここで，D_{n-1} は $n-1$ 層目の全結合層の次元（総ニューロン数），$o_q^{n-1} \in \mathbb{R}$ 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，

$$o_q^{n-1} = \phi\left(a_q^{n-1}\right) \tag{3.74}$$

である．例外として，各 $i \in \mathcal{I}$ 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 (3.73) に対し，$w_{0,p}^n = b_p^n$，$o_0^{n-1} = 1$ とおけば，

$$a_p^n = b_p^n + \sum_{q=1}^{D_{n-1}} w_{q,p}^n o_q^{n-1} = \sum_{q=0}^{D_{n-1}} w_{q,p}^n o_q^{n-1} \tag{3.75}$$

と整理できる．この時，

$$\frac{\partial \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right)}{\partial w_{p,q}^n} = \frac{\partial}{\partial w_{p,q}^n}\left(\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)\right) \tag{3.76}$$

$$= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\partial L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)}{\partial w_{p,q}^n} \tag{3.77}$$

$$= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\partial L_i}{\partial w_{p,q}^n} \tag{3.78}$$

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 (3.75) を用いて，

$$\frac{\partial L_i}{\partial w_{p,q}^n} = \frac{\partial L_i}{\partial a_q^n} \frac{\partial a_q^n}{\partial w_{p,q}^n} \tag{3.79}$$

$$= \delta_q^n \frac{\partial}{\partial w_{p,q}^n} \left(\sum_{r=0}^{D_{n-1}} w_{r,q}^n o_r^{n-1} \right) \tag{3.80}$$

$$= \delta_q^n o_p^{n-1} \tag{3.81}$$

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，

$$\frac{\partial L_i}{\partial w_{p,q}^{N_{\text{layer}}}} = \delta_q^{N_{\text{layer}}} o_p^{N_{\text{layer}}-1} \tag{3.82}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L_i}{\partial a_q^{N_{\text{layer}}}} \tag{3.83}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)}{\partial a_q^{N_{\text{layer}}}} \tag{3.84}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L\left(\boldsymbol{y}_i, \phi\left(\boldsymbol{a}^{N_{\text{layer}}}\right)\right)}{\partial a_q^{N_{\text{layer}}}} \tag{3.85}$$

$$= o_p^{N_{\text{layer}}-1} \left(\nabla_{\phi\left(\boldsymbol{a}^{N_{\text{layer}}}\right)} L\left(\boldsymbol{y}_i, \phi\left(\boldsymbol{a}^{N_{\text{layer}}}\right)\right) \right)^\top \frac{\partial \phi\left(\boldsymbol{a}^{N_{\text{layer}}}\right)}{\partial a_q^{N_{\text{layer}}}} \tag{3.86}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L\left(\boldsymbol{y}_i, \phi\left(\boldsymbol{a}^{N_{\text{layer}}}\right)\right)}{\partial \phi\left(a_q^{N_{\text{layer}}}\right)} \phi'\left(a_q^{N_{\text{layer}}}\right) \tag{3.87}$$

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，

$$\frac{\partial L_i}{\partial w_{p,q}^n} = \delta_q^n o_p^{n-1} \tag{3.88}$$

$$= o_p^{n-1} \frac{\partial L_i}{\partial a_q^n} \tag{3.89}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \frac{\partial L_i}{\partial a_r^{n+1}} \frac{\partial a_r^{n+1}}{\partial a_q^n} \tag{3.90}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} \frac{\partial}{\partial a_q^n} \left(\sum_{s=0}^{D_n} w_{s,r}^{n+1} o_s^n \right) \tag{3.91}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} \frac{\partial}{\partial a_q^n} \left(\sum_{s=0}^{D_n} w_{s,r}^{n+1} \phi\left(a_s^n\right) \right) \tag{3.92}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} w_{q,r}^{n+1} \phi'\left(a_q^n\right) \tag{3.93}$$

$$= o_p^{n-1} \phi'\left(a_q^n\right) \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} w_{q,r}^{n+1} \tag{3.94}$$

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，

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right) \leftarrow \frac{c}{\max\{\left\|\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right)\right\|_2, c\}} \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right) \tag{3.95}$$

で与えられる．ここで，$c \in (0, \infty)$ は勾配の L2 ノルムに対する閾値である．

また，近年は数億単位のパラメータを持つ大規模なモデルも提案されており，こういった規模間のモデル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 $\boldsymbol{g}_{\text{accum}}$ とすると，この更新は

$$\boldsymbol{g}_{\text{accum}} \leftarrow \boldsymbol{g}_{\text{accum}} + \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}\right) \tag{3.96}$$

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

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \frac{\eta}{N_{\text{accum}}} \boldsymbol{g}_{\text{accum}} \tag{3.97}$$

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 $1/N_{\text{accum}}$ 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．

3.2.8 自己教師あり学習

近年，音声や動画を用いる分野では，自己教師あり学習を事前に行ったモデルを特定の問題にFineTuningする転移学習の有効性が確認されている．自己教師あり学習とは、教師ラベルのないデータから特徴を学習する手法であり、データ自体を利用して擬似的な教師ラベルを生成し、教師あり学習を行う点が特徴である。このアプローチは教師なし学習と類似しているが、教師ラベルを生成して利用する点で教師あり学習に近いといえる。ここでは特に，本研究で用いる自己教師あり学習モデルであるHuBERT[20]，AVHuBERT[3]で行われている，Masked Predictionという自己教師あり学習方法について述べる．

Masked Predictionでは，データの一部をマスクした上でモデルに入力し，マスクされた領域をモデルに予測させることで学習を行う．入力データを $\boldsymbol{X} \in \mathbb{R}^{T\times D}$ とし，このうちマスクされるインデックスの部分集合を $\mathcal{M} \subset \{1, \dots T\}$ とする．この時，マスクされた入力 $\boldsymbol{X}^{\text{masked}}$ の各時刻 t における値 $\boldsymbol{x}_t^{\text{masked}}$ は，

$$\boldsymbol{x}_t^{\text{masked}} = \begin{cases} \boldsymbol{x}_t & \text{if } t \notin \mathcal{M} \\ \boldsymbol{m} & \text{if } t \in \mathcal{M} \end{cases} \tag{3.98}$$

である．$\boldsymbol{m} \in \mathbb{R}^D$ はマスク専用のベクトルである．ここで，HuBERTでは音声データ，AVHuBERTでは動画データと音声データを扱うが，いずれもクラスタリングによって連続特徴量を離散化し，教師ラベルを作成する．クラスタ数 C のクラスタリングによって得られる教師ラベルを $\boldsymbol{Y} \in \{0, 1\}^{T\times C}$ とする．各時刻 t における教師ラベル $\boldsymbol{y}_t$ は，正解クラスが1，それ以外が0となったOne-hotベクトルである．この時，モデルの予測値 $\hat{\boldsymbol{Y}} \in [0, 1]^{T\times C}$ に対する損失は，

$$L_{\text{CE-masked}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = -\frac{1}{|\mathcal{M}|}\sum_{t\in\mathcal{M}}\sum_{c=1}^{C} y_{t,c}\log\left(\hat{y}_{t,c}\right) \tag{3.99}$$

で与えられる．すなわち，マスクされた位置に限定したCross Entropy Lossである．この損失関数の値を最小化するためには，未知の情報を正しく穴埋めできるように観測できる情報から特徴抽出を行う必要があるから，最適化されたDNNは音声や動画における文脈的な構造を学習していると考えられる．また，音声認識やVSRでは予測対象がテキストになるため，音声データおよび動画データに対するテキストアノテーションを行う必要がある．これに対し，Masked Predictionはテキストを必要としない学習方法であるから，より多くのリソースを用いた学習が可能になるという利点がある．

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

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

とする．各 n に対し，$\boldsymbol{X}_n^{\text{video}} \in \mathbb{R}^{D^{\text{video}} \times T_n^{\text{video}} \times W^{\text{video}} \times H^{\text{video}}}$ 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，$\boldsymbol{x}_{s_n}^{\text{spk-emb}} \in \mathbb{R}^{D^{\text{spk-emb}}}$ は話者ベクトル，$\boldsymbol{Y}_n^{\text{mel}} \in \mathbb{R}^{D^{\text{mel}} \times T_n^{\text{mel}}}$ はメルスペクトログラム，$\boldsymbol{Y}_n^{\text{HuB-int}} \in \mathbb{R}^{D^{\text{HuB-int}} \times T_n^{\text{HuB}}}$ は HuBERT 中間特徴量，$\boldsymbol{Y}_n^{\text{HuB-disc}} \in \{0,1\}^{C \times T_n^{\text{HuB}}}$ は HuBERT 離散特徴量とする．ここで，話者ベクトル $\boldsymbol{x}_{s_n}^{\text{spk-emb}}$ は，話者識別モデル [21] 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）であり，

$$\boldsymbol{x}_{s_n}^{\text{spk-emb}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} f_{\text{spk}} \left(\boldsymbol{y}_i^{\text{sp-wf}}; \boldsymbol{\theta}_{\text{spk}} \right) \tag{4.2}$$

で与えられる．ここで，$\boldsymbol{\theta}_{\text{spk}}$ は事前学習済み重みを表す．$\mathcal{I}$ は学習データセット $\mathcal{D}_{\text{train}}$ において、話者 s_n と話者が同一であるデータのインデックス集合から、ランダムに $N_{\text{spk-emb}}$ 個のインデックスを抽出した部分集合とする。すなわち、

$$\mathcal{I} \subset \{ i \mid i \in \{1, \ldots, N\},\ s_i = s_n \}, \quad |\mathcal{I}| = N_{\text{spk-emb}} \tag{4.3}$$

である。次に，HuBERT 中間特徴量と HuBERT 離散特徴量は，HuBERT における畳み込みエンコーダまで（Transformer 層以前）を $f_{\text{HuB-conv}}$，HuBERT における Transformer 層（$f_{\text{HuB-conv}}$ 以後）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，

$$\boldsymbol{Y}_n^{\text{HuB-int}} = f_{\text{HuB-conv}} \left(\boldsymbol{y}_n^{\text{sp-wf}}; \boldsymbol{\theta}_{\text{HuB-conv}} \right) \tag{4.4}$$

$$\boldsymbol{Y}_n^{\text{HuB-disc}} = \text{one-hot} \left(f_{\text{k-means}} \left(f_{\text{HuB-trans}} \left(\boldsymbol{Y}_n^{\text{HuB-int}}; \boldsymbol{\theta}_{\text{HuB-trans}} \right) \right) \right) \tag{4.5}$$

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．HuBERT を利用した特徴量計算の流れを図 4.1に示す。

提案手法を図 4.2に示す．提案手法は，ネットワーク A，ネットワーク B，ボコーダの三つからなる．まず，ネットワーク A を f_{A} とすると，f_{A} の行う処理は，

$$\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}, \hat{\boldsymbol{Y}}_n^{\text{mel-A}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}} = f_{\text{A}} \left(\boldsymbol{X}_n^{\text{video}}, \boldsymbol{x}_{s_n}^{\text{spk-emb}}; \boldsymbol{\theta}_{\text{A}} \right) \tag{4.6}$$

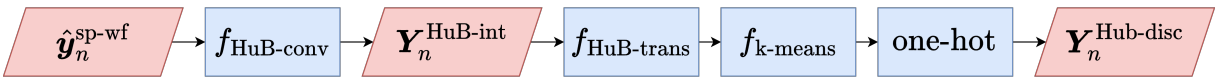


図 4.1: HuBERT を利用した特徴量計算の流れ

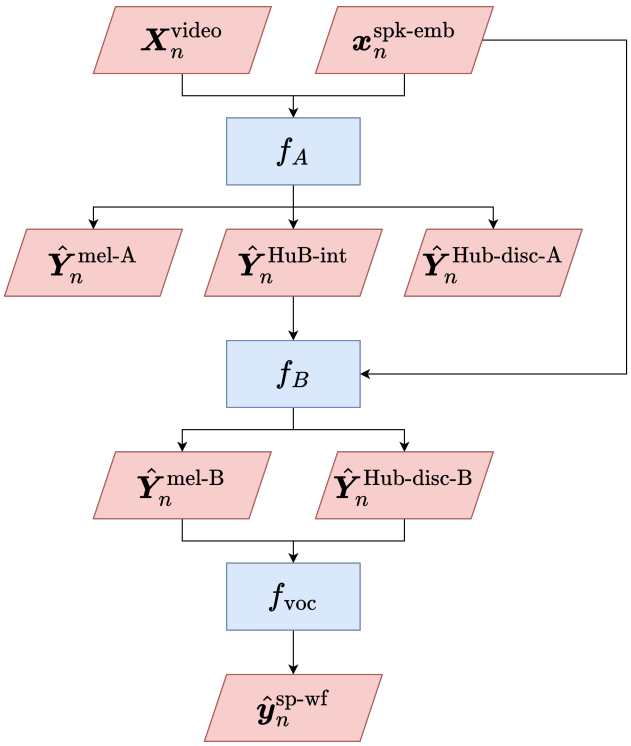


図 4.2: 提案手法の全体像

と表される．ここで，$\hat{\boldsymbol{Y}}_n^{\text{HuB-int}} \in \mathbb{R}^{D^{\text{HuB-int}} \times T_n^{\text{HuB}}}$ は予測 HuBERT 中間特徴量，$\hat{\boldsymbol{Y}}_n^{\text{mel-A}} \in \mathbb{R}^{D^{\text{mel}} \times T_n^{\text{mel}}}$ はネットワーク A の予測メルスペクトログラム，$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}} \in \mathbb{R}^{C \times T_n^{\text{HuB}}}$ 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，

$$\hat{\boldsymbol{Y}}_n^{\text{mel-B}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}} = f_{\text{B}}\left(\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}, \boldsymbol{x}_{s_n}^{\text{spk-emb}}; \boldsymbol{\theta}_{\text{B}}\right) \tag{4.7}$$

と表される．ここで，$\hat{\boldsymbol{Y}}_n^{\text{mel-B}} \in \mathbb{R}^{D^{\text{mel}} \times T_n^{\text{mel}}}$ はネットワーク B の予測メルスペクトログラム，$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}} \in \mathbb{R}^{C \times T_n^{\text{HuB}}}$ 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，

$$\hat{\boldsymbol{y}}_n^{\text{sp-wf}} = f_{\text{voc}}\left(\hat{\boldsymbol{Y}}_n^{\text{mel-B}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}; \boldsymbol{\theta}_{\text{voc}}\right) \tag{4.8}$$

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，中間表現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 4.3a に示す。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 $\boldsymbol{H}_n^{\text{A}} \in \mathbb{R}^{D^{\text{A}} \times T_n^{\text{video}}}$ に変換する．これは，

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

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．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 $\boldsymbol{x}_{s_n}^{\text{spk-emb}}$ 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，

$$\boldsymbol{H}_n^{\text{A}} = \text{FC}\left(\left[\boldsymbol{H}_n^{\text{A}}, \left(\boldsymbol{x}_{s_n}^{\text{spk-emb}}\right)_T\right]; \boldsymbol{\theta}_{\text{A-fc-spk}}\right) \tag{4.10}$$

と表される．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，

$$\boldsymbol{H}_n^{\text{A}} = f_{\text{post}}\left(\boldsymbol{H}_n^{\text{A}}; \boldsymbol{\theta}_{\text{A-post}}\right) \tag{4.11}$$

と表される．ここで、後処理層 f_{post} の構造を図 4.4に示す。f_{post} は，一次元畳み込み層を主とした 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 $\boldsymbol{x}_{s_n}^{\text{spk-emb}}$ が次元方向に結合された特徴量 $\boldsymbol{H}_n^{\text{A}}$ に対する話者性を考慮した特徴量の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であるから、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、すなわち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，$\boldsymbol{H}_n^{\text{A}}$ 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，

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

$$\hat{\boldsymbol{Y}}_n^{\text{mel-A}} = \text{FC}\left(\boldsymbol{H}_n^{\text{A}}; \boldsymbol{\theta}_{\text{A-fc-mel}}\right) \tag{4.13}$$

$$\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}} = \text{FC}\left(\boldsymbol{H}_n^{\text{A}}; \boldsymbol{\theta}_{\text{A-fc-HuB-Disc}}\right) \tag{4.14}$$

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 離散特徴量の推定を同時に行った理由は，先行研究 [4, 5] 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 中間特徴量の推定においても有効ではないかと考えたからである。

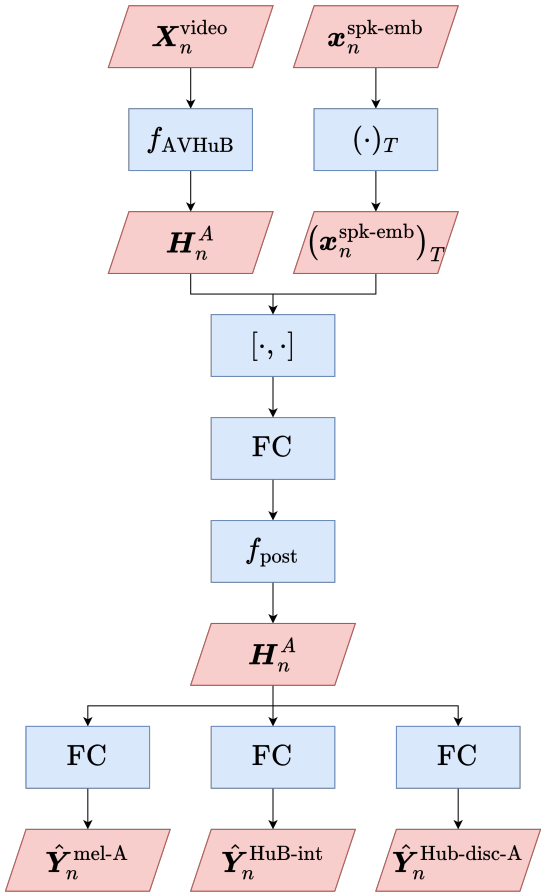
4.1.3 ネットワーク B

ネットワーク B を図 4.3bに示す。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 $\boldsymbol{H}_n^{\text{B}} \in \mathbb{R}^{D^{\text{B}} \times T_n^{\text{HuB}}}$ に変換する．これは，

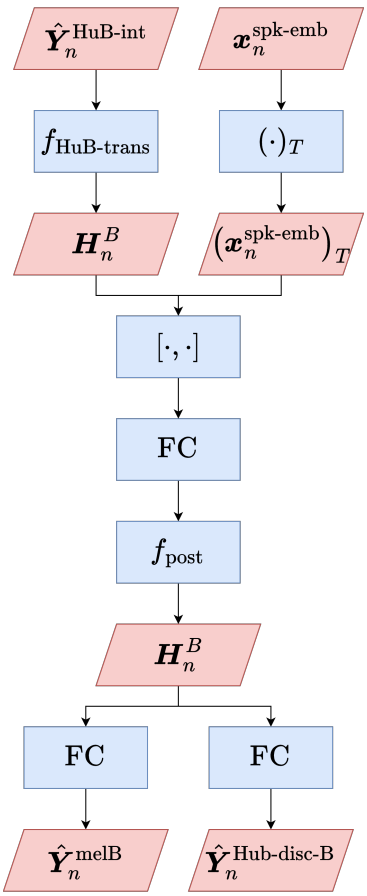
$$\boldsymbol{H}_n^{\text{B}} = f_{\text{HuB-trans}}\left(\hat{\boldsymbol{Y}}_n^{\text{HuB-int}}; \boldsymbol{\theta}_{\text{B-HuB-trans}}\right) \tag{4.15}$$

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二つを検討した．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．

$$\boldsymbol{H}_n^{\text{B}} = \text{FC}\left(\left[\boldsymbol{H}_n^{\text{B}}, \left(\boldsymbol{x}_{s_n}^{\text{spk-emb}}\right)_T\right]; \boldsymbol{\theta}_{\text{B-fc-spk}}\right) \tag{4.16}$$



(a) ネットワーク A



(b) ネットワーク B

図 4.3: ネットワーク A とネットワーク B の構造

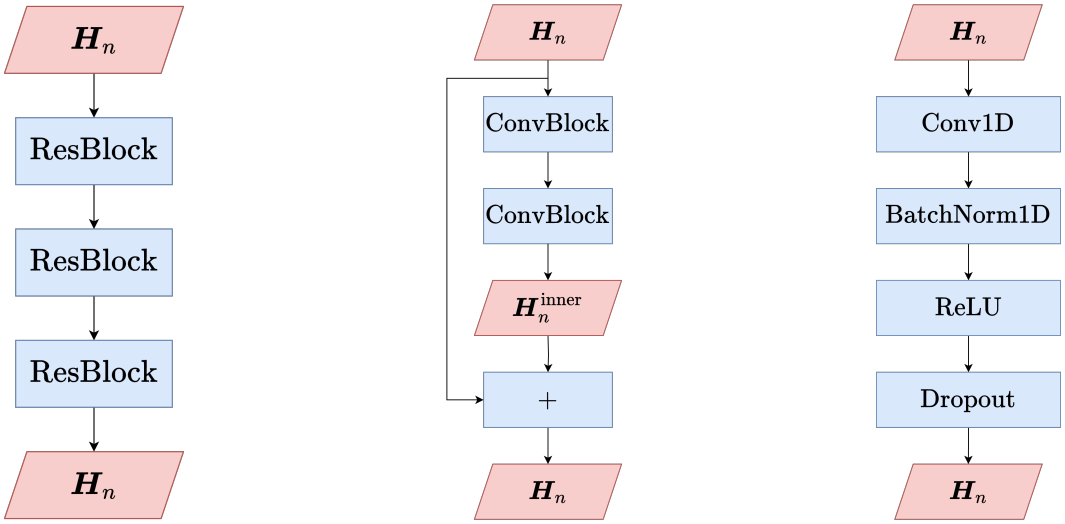
$$\boldsymbol{H}_n^{\mathrm{B}} = f_{\text{post}}\left(\boldsymbol{H}_n^{\mathrm{B}}; \boldsymbol{\theta}_{\text{B-post}}\right) \tag{4.17}$$

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

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

$\boldsymbol{\theta}_{\text{B-fc-spk}}, \boldsymbol{\theta}_{\text{B-post}}, \boldsymbol{\theta}_{\text{B-fc-mel}}, \boldsymbol{\theta}_{\text{B-fc-HuB-disc}}$ 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．HuBERT Transformer 層の転移学習を検討した狙いは、HuBERT の自己教師あり学習の方法に基づく。HuBERT は Masked Prediction という自己教師あり学習を行っており、これは、畳み込みエンコーダ $f_{\text{HuB-conv}}$ からの出力にマスクを適用し，Transformer 層 $f_{\text{HuB-trans}}$ を通すことによって、マスクされた部分を推定する学習方法である．3.2.8節より、Masked Prediction では未知の情報を正しく穴埋めできるように観測できる情報から特徴抽出を行う必要があるから，最適化された HuBERT は音声の文脈的な構造を学習していると考えられる．また、Masked Prediction は音声データのみによる学習が可能であり、近年では大規模なデータによって学習されたモデルも一般に公開されている。よって、本研究では HuBERT Transformer 層を動画音声合成に Fine Tuning することにより，動画を入力とするネットワーク A における推定残差を，大量の



(a) 全体　(b) ResBlock　(c) ConvBlock

図 4.4: 後処理層 f_{post} の構造

音声データから学習された音声自体の文脈を考慮する力によって軽減することで、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推定精度改善を狙った.

4.1.4　ボコーダ

ボコーダを図 4.5に示す。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 [5] 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，

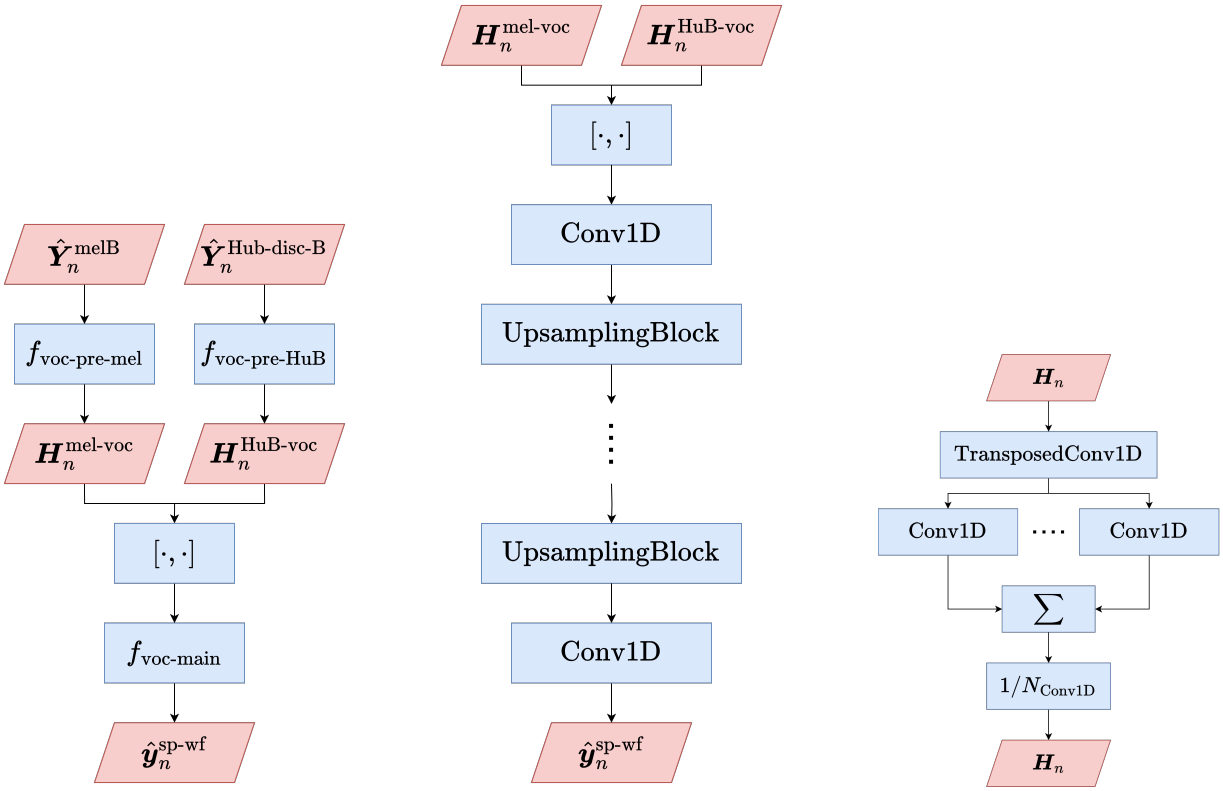
$$\boldsymbol{H}_n^{\text{mel-voc}} = f_{\text{voc-pre-mel}}\left(\hat{\boldsymbol{Y}}_n^{\text{mel-B}}; \boldsymbol{\theta}_{\text{voc-pre-mel}}\right) \tag{4.20}$$

$$\boldsymbol{H}_n^{\text{HuB-voc}} = f_{\text{voc-pre-HuB}}\left(\hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}; \boldsymbol{\theta}_{\text{voc-pre-HuB}}\right) \tag{4.21}$$

と表される．$\boldsymbol{H}_n^{\text{mel-voc}} \in \mathbb{R}^{D^{\text{mel-voc}} \times T_n^{\text{HuB}}}$ 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，$\boldsymbol{H}_n^{\text{HuB-voc}} \in \mathbb{R}^{D^{\text{HuB-voc}} \times T_n^{\text{HuB}}}$ 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，

$$\hat{\boldsymbol{y}}_n^{\text{sp-wf}} = f_{\text{voc-main}}\left(\left[\boldsymbol{H}_n^{\text{mel-voc}}, \boldsymbol{H}_n^{\text{HuB-voc}}\right]; \boldsymbol{\theta}_{\text{voc-main}}\right) \tag{4.22}$$

と表される．$f_{\text{voc-main}}$ は，図 4.5bに示すように，一次元畳み込み層と UpsamplingBlock から構成され，これは HiFi-GAN の Generator と同様である．各 UpsamplingBlock では，初めに一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，その後複数の一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．



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

図 4.5: ボコーダの構造

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，

$$\begin{aligned} &L_A\left(\boldsymbol{Y}_n^{\text{HuB-int}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-int}}, \boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-A}}, \boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}}\right) \\ &=\lambda_{\text{HuB-int}} L_{\text{MAE}}\left(\boldsymbol{Y}_n^{\text{HuB-int}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-int}}\right) + \lambda_{\text{mel}} L_{\text{MAE}}\left(\boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-A}}\right) \\ &\quad + \lambda_{\text{HuB-disc}} L_{\text{CE}}\left(\boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-A}}\right) \end{aligned} \tag{4.23}$$

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，

$$\begin{aligned} &L_B\left(\boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-B}}, \boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}\right) \\ &=\lambda_{\text{mel}} L_{\text{MAE}}\left(\boldsymbol{Y}_n^{\text{mel}}, \hat{\boldsymbol{Y}}_n^{\text{mel-B}}\right) + \lambda_{\text{HuB-disc}} L_{\text{CE}}\left(\boldsymbol{Y}_n^{\text{HuB-disc}}, \hat{\boldsymbol{Y}}_n^{\text{HuB-disc-B}}\right) \end{aligned} \tag{4.24}$$

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．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 と同様であるから，ここでは割愛する．

表 4.1: 利用したデータセットの文章数

	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

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 [22, 23] で収録されたもののうち，ATR 音素バランス文 [24] を読み上げたデータを利用した．このデータセットは、男女二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．各分割ごとの文章数を表 4.1に示す．

ボコーダの学習には，Hi-Fi-Captain[25] と JVS[26] 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．各分割ごとの文章数を表 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．各分割ごとの文章数を表 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 [27] 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，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 秒あたり 0 から 0.5 秒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．

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

音声データは 16 kHz にダウンサンプリングして用いた．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（-40 dBFS 未満かつ 500 ms 継続する区間を 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，動画音声データセット，Hi-Fi-Captain，JVS ともに，各話者に対し学習データの中から 100 文章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 Canonical Correlation Analysis（CCA）によって調べた先行研究 [28] 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一つ追加したため，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，VoxCeleb1[29] と VoxCeleb2，LibriSpeech[30] の other セットで学習されたものを用いた[2]．学習データセットの記述は，このモデルを利用して複数話者 TTS を検討した先行研究 [31] の GitHub リポジトリにある[3]．話者識別モデルは，音声波形をメルスペクトログラムに変換した後，これを 1.6 秒ごとに 0.77 秒のオーバーラップを持つよう分割し，各区間ごとに一つのベクトル表現を獲得した後，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．モデル構造は 3 層の LSTM と全結合層からなり，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，LRS3 と VoxCeleb2 の英語データを利用し，データ拡張として音声データにノイズを付与した場合 [32] 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．ノイズ付与の場合を利用した理由は，音声にノイズが含まれることで動画からの特徴抽出が促進され，より動画タスクに適した重みになっているのではないかと考えたからである．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 によって形状

[2] https://github.com/resemble-ai/Resemblyzer
[3] https://github.com/CorentinJ/Real-Time-Voice-Cloning/wiki/Training
[4] https://github.com/facebookresearch/av_hubert

が $(D \times T \times W \times H)$ である動画からの特徴抽出および空間情報の圧縮が行われ，出力特徴量の形状は $(D \times T)$ となる．これに対して Transformer 層を適用することで，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 次元で，25 Hz である．

HuBERT の事前学習済み重みには，HuggingFace に公開されている ReazonSpeech によって学習されたもの [33, 34] を利用した[5]．ReazonSpeech は約 19000 時間の日本語音声からなるデータセットであり，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 は一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 層を適用することで，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の次元は 768 次元で，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．

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

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積むことで 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，最も確率の高いクラスを選択する．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する．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 の特徴量が得られる．前処理後の二つの特徴量を次元方向に結合することで，256 次元 50 Hz の特徴量が得られる．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 次元まで拡大する．これに対し，転置畳み込み層による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 の特徴量を獲得する．各 UpsamplingBlock におけるパラメータを表 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 は次元，T は系列長である．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．すなわち，転置畳み込み層一層に対し，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われる．最後に，畳み込み層によって 1 次元まで次元を削減することで，16 kHz の音声波形が得られる．

[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

	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	$(4, 2)$	$(\{3, 5, 7, 9, 11\}, \{1, 3, 5\})$	$(16, 16000)$

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[18] 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，$\eta_{\min} = 1.0 \times 10^{-6}, \eta_{\max} = 1.0 \times 10^{-3}, \text{warmupSteps} = 5, \tau_{\max}^{\text{epoch}} = 50$ とした．バッチサイズは 4 とし，8 イテレーションに一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 秒を上限とし，10 秒を超える場合はランダムにトリミング，10 秒未満の場合はゼロパディングした．また，ゼロパディングした部分は損失の計算からは除外した．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 $0.0001, 0.001, 0.01, 0.1, 1.0$ の五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，$\eta_{\min} = 1.0 \times 10^{-6}, \eta_{\max} = 5.0 \times 10^{-4}, \text{warmupSteps} = 5, \tau_{\max}^{\text{epoch}} = 50$ 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 $0.0001, 0.001, 0.01, 0.1, 1.0$ の五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，$\gamma = 0.99$ 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 秒を上限とし，1 秒を超える場合はランダムにトリミング，1 秒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．また，先行研究 [5] においては学習時に，メルスペクトログラムにノイズを付与するデータ拡張手法が提案されている．本研究では，動画から推定されるメルスペクトログラムと HuBERT 離散特徴量の推定精度向上に焦点を当てたため，ボコーダの学習は原音声から計算される特徴量で行い，ボコーダ自体の汎化性能向上による精度改善は追求しなかった．

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，二種類の指標を用いた．一つ目は，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，二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，

$$\mathrm{WER}\left(s_1, s_2\right) = \frac{I + D + R}{|s_1|} \tag{4.25}$$

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[35] 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．

二つ目は，話者ベクトルから計算したコサイン類似度である．モデルへの入力値を計算するのに用いた話者識別モデルを同様に利用して，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二点を評価した．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

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

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．

一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，無回答の三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，ヘッドホンの二つからの選択式とした．

二つ目の練習試行（明瞭性）および三つ目の本番試行（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，本番試行は一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，一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三段階で構成した．

1. 提示された音声サンプルを一回再生し，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 が手法のリスト，speakers が話者のリスト，$N_{\text{respondents}}$ が被験者総数である．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，二つのことに注意した．一つ目

は，各被験者がユニークな発話文章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．二つ目は，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ており，各被験者になるべく均等に評価を受けることが望ましいと判断した．また，評価時に音声サンプルを一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，本実験では一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．

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

四つ目の練習試行（類似性）および五つ目の本番試行（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，本番試行は一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二段階で構成した．

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五段階評価する．

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
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 sentenceEval に対して比較対象となる原音声用の 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．

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, $N_{\text{respondents}}$
allAssignments = {}
for respondentId = 0 to $N_{\text{respondents}} - 1$ **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 [respondentId] = assignments
end for
return allAssignments

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 [36] 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．

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 段階の評価の回答項目からランダムに決定した．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．また，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，謝礼は実験一回あたり 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，以下の四つである．

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マルチタスク学習手法である．よって，損失関数は L_B と同じである．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，ここでも客観評価指標を確認する．ネットワーク B（Randomized）は，HuBERT Transformer 層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 層の転移学習の有効性を調べた．

まず，損失関数の重み係数 $\lambda_{\text{HuB-disc}}$ を変化させた時の，客観評価指標の値を表 4.3に示す．各手法の客観評価指標ごとに，最も優れた値を下線で示している．また，各手法ごとに最適だと判断した場合を太字で示している．

ベースラインでは，$\lambda_{\text{HuB-disc}}$ の値が 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り，話者類似度はほとんど同じであったため，今回は 0.001 が最適であると判断した．図 4.6に学習曲線を示す．横軸がエポック数，縦軸が損失の値を表す．損失の値は各エポックにおける平均値である．実線は検証データに対する損失，点線は学習データに対する損失を表しており，線の色は $\lambda_{\text{HuB-disc}}$ の違いを表す．また，丸いマーカーはテストデータの合成に用いたエポックにおける損失の値を表す．学習曲線より，$\lambda_{\text{HuB-disc}}$ の値を大きくすることによって，HuBERT 離散特徴量についての Cross Entropy Loss の下がり方が急峻になっていることがわかる．また，ベースラインでは $\lambda_{\text{HuB-disc}}$ の値が 0.1 よりも大きくなると，話者類似度が低下する傾向が見られた．これについて，学習曲線を見ると

メルスペクトログラムについての MAE Loss が下がりきっていない傾向が見られる．HuBERT 離散特徴量は言語的な情報を持つ特徴量であり，音声の話者性はメルスペクトログラムに影響されると考えられるため，この損失が高い状態で学習が終了したことが話者類似度低下の原因だと考える．

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等であることがわかる．図 4.6に学習曲線を示す．これについてもベースラインと同様であることが分かる．

ネットワーク B（Randomized）では，$\lambda_{\text{HuB-disc}}$ の値が 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．図 4.8に学習曲線を示す．ベースラインと異なるのは，$\lambda_{\text{HuB-disc}}$ が 0.1 の場合，メルスペクトログラムについての MAE Loss の達する最小値自体が，$\lambda_{\text{HuB-disc}}$ が 0.01 以下の場合と比較して小さくなっていることである．これより，最適と判断した $\lambda_{\text{HuB-disc}} = 0.1$ の場合，メルスペクトログラムについての MAE Loss と HuBERT 離散特徴量についての Cross Entropy Loss をバランスよく下げられていたと考えられる．

ネットワーク B（Pretrained）では，$\lambda_{\text{HuB-disc}}$ を 0.1 としたときに WER が最低となる一方で，0.01 以上の場合と比較したときの話者類似度の低下が顕著であり，これはランダム初期化したネットワーク B（Randomized）とは異なる傾向であった．今回は WER が低いことを優先して，0.1 が最適であると判断した．図 4.9に学習曲線を示す．$\lambda_{\text{HuB-disc}} = 0.1$ の場合に注目すると，ネットワーク B（Randomized）と比較して学習がより早期に停止しており，メルスペクトログラムについての MAE Loss を下げきれていないことが分かり，これが話者類似度の低下につながったと考えられる．

最後に，最適なチューニングをした場合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特徴量を入力としてボコーダで逆変換した合成音声であり，本実験下において合成音声により達成され得る上限を表す．ベースラインからネットワーク B（Pretrained）は，最適なチューニングだと判断したものを選択している．また，ベースラインからネットワーク B（Pretrained）の中で，最も優れた値を下線で示している．これより，提案手法であるネットワーク B（Randomized）は，ベースラインに対して WER と話者類似度の両方を改善したことがわかる．一方，ネットワーク B（Pretrained）は WER の改善を達成したが，話者類似度については悪化したことがわかる．また，ネットワーク B（Randomized）とネットワーク B（Pretrained）を比較すると，ネットワーク B（Pretrained）の方が WER は 1.1%低いが，話者類似度が 0.062 低いことがわかる．実際に音声を聞いてみると，音声に不自然なノイズが含まれており，原音声に対する類似性が下がっていることが確認された．よって，HuBERT の事前学習済み重みを初期値とした HuBERT Transformer 層の転移学習は，本実験条件下においては有効でなかったと考えられる．

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

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当初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，さらなる検討を行った結果を述べる．

まず，HuBERT Transformer 層への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入力と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積むことで 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

表 4.4: 最適なチューニングをした場合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	0.840
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

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

手法	λ_{ssl^d}	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

まま用いた．これらを次元方向に結合することで 261 次元の特徴量を構成し，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 層への入力とした．結果を表 4.5に示す．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，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 は 10.1% 高いことが分かる．図 4.10にネットワークB（Randomized・Mel-HuB）の学習曲線を示す．ネットワークB（Randomized）の学習曲線と比較すると，ネットワークB（Randomized・Mel-HuB）は $\lambda_{\text{HuB-disc}}$ を大きくした時に，メルスペクトログラムの MAE Loss の下がりが悪い傾向にあることが分かる．例えば，$\lambda_{\text{HuB-disc}} = 0.1$ の場合における 20 エポック目の学習データに対する損失を見ると，ネットワークB（Randomized）では 0.5 を下回っているのに対し，ネットワークB（Randomized・Mel-HuB）は 0.5 を上回っている．以上より，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して，ネットワークB の推定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．HuBERT 中間特徴量は 768 次元であったのに対し，メルスペクトログラムと HuBERT 離散特徴量の複合特徴量は 261 次元であるから，より高次元で冗長性が高いことが，ネットワークB による特徴抽出の対象に適していたのではないかと考える．

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

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	λ_{ssl^d}	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

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．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，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．図 4.11にネットワーク B（Randomized・A-SingleTask）の学習曲線を示す．ネットワーク B（Randomized）の学習曲線と比較すると，ネットワーク B（Randomized・Mel-HuB）は $\lambda_{\text{HuB-disc}}$ を大きくした時に，メルスペクトログラムの MAE Loss の下がりが良い傾向にあることが分かる．例えば，$\lambda_{\text{HuB-disc}} = 0.1$ の場合における 10 エポック目の学習データに対する損失を見ると，ネットワーク B（Randomized）では 0.5 を上回っているのに対し，ネットワーク B（Randomized・Mel-HuB）は 0.5 程度に達している．以上より，ネットワーク A ではネットワーク B の入力に必要な HuBERT 中間特徴量のみを推定する方が，ネットワーク B（Randomized）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．マルチタスク学習を行う場合，メルスペクトログラムや HuBERT 離散特徴量が損失に加わるため，それらの損失を小さくするための勾配も考慮した重みの更新が行われる．この影響が，特にネットワーク B（Randomized）への入力となる HuBERT 中間特徴量を推定する上では，悪影響を与えていたと考えられる．

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，以下の五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五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．

まず，被験者に対するアンケートで得られた統計について，被験者の年齢層は 21 歳から 62 歳に渡った．年齢層の箱ひげ図を図 4.12に示す．被験者の性別は男性 32 名，女性 43 名であった．また，実験に利用した音響機器はヘッドホンが 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，手法ごとに平均値と 95%信頼区間を計算した結果を表 4.7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（片側検定）を行った結果を表 4.8, 4.9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 $\mu_i > \mu_j$ とした片側検定で計算された p 値に対し，Benjamini/Hochberg 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，表の文字列はそれぞれの以下の略称とする．

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

また，本実験における有意水準は 5%とする．

まず，表 4.8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，ベースラインに対して明瞭性の評価値が有意に高いことがわかる．これより，二つの提案手法はどちらもベースラインより明瞭性の高い合成音声を実現できたと考えられる．一方，提案手法二つの間には有意差がないことも分かる．これより，ネットワーク A の学習方法の違いは明瞭性に有意差をもたらさなかったと言える．次に，表 4.9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，ベースラインに対して類似性の評価値が有意に高いことがわかる．これより，二つの提案手法はどちらもベースラインより類似性の高い合成音声を実現できたと考えられる．また，ここでは提案手法二つの間にも有意差があることが分かる．これより，ネットワーク A の学習方法の違いは類似性に有意な差をもたらしたと言える．

以上のことから，ネットワーク B（Randomized・A-SingleTask）が明瞭性・類似性の両面において，優れた合成音声を実現したと考えられる．一方，ネットワーク B（Randomized・A-SingleTask）と，合成音声の性能上限を表す分析合成の間には未だ大きな差があり，自然音声に迫る合成音の実現は達成されていない．従って，本実験におけるベースラインからの改善は達成したものの，今後もさらなるネットワークの改善が必要だと考えられる．

4.4 考察

表 4.7: 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

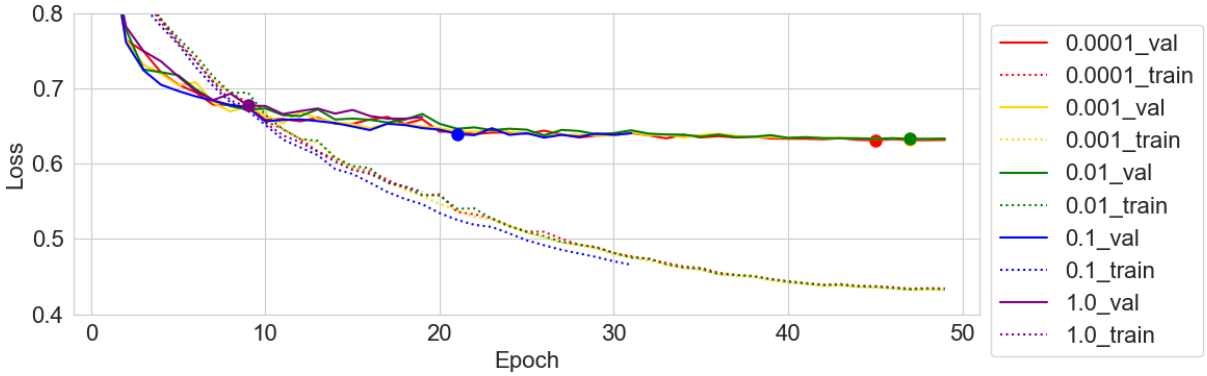
手法	λ_{ssl^d}	明瞭性	類似性
ベースライン	0.001	2.37 ± 0.07	2.92 ± 0.08
ネットワーク B（Randomized）	0.1	2.76 ± 0.07	3.04 ± 0.08
ネットワーク B（Randomized・A-SingleTask）	0.1	2.82 ± 0.08	3.18 ± 0.08
分析合成	-	4.75 ± 0.04	4.31 ± 0.07
原音声	-	4.87 ± 0.03	4.71 ± 0.05

表 4.8: 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）

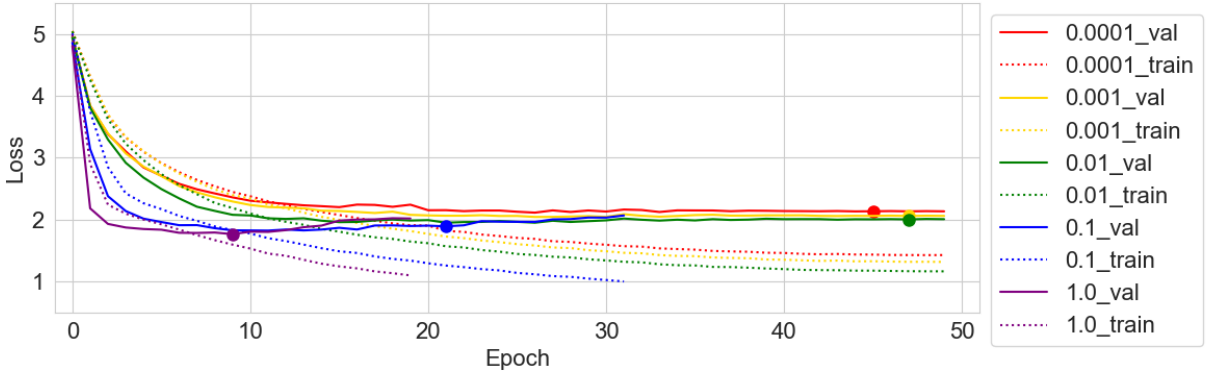
	AbS	B（Rand, A-S）	B（Rand）	Baseline
GT	6.67×10^{-6}	6.17×10^{-262}	2.84×10^{-289}	0
AbS	-	3.22×10^{-243}	2.40×10^{-270}	0
B（Rand, A-S）	-	-	1.23×10^{-1}	7.07×10^{-17}
B（Rand）	-	-	-	1.12×10^{-13}

表 4.9: 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）

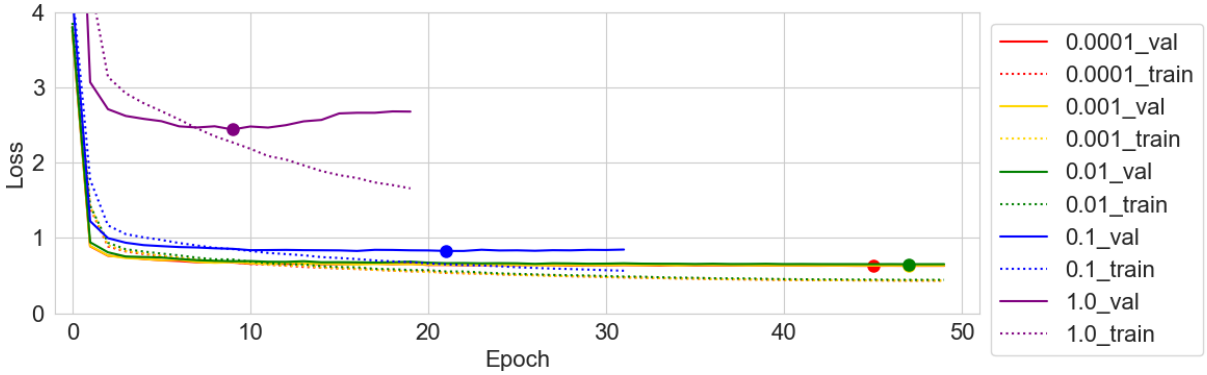
	AbS	B（Rand, A-S）	B（Rand）	Baseline
GT	1.24×10^{-18}	5.86×10^{-160}	9.90×10^{-174}	1.28×10^{-205}
AbS	-	3.11×10^{-83}	1.73×10^{-97}	3.58×10^{-120}
B（Rand, A-S）	-	-	1.13×10^{-2}	5.46×10^{-6}
B（Rand）	-	-	-	2.13×10^{-2}



(a) メルスペクトログラムについての MAE Loss

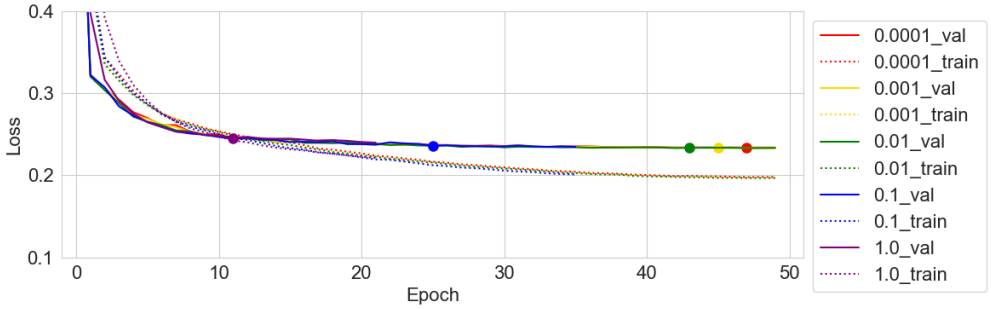


(b) HuBERT 離散特徴量についての Cross Entropy Loss

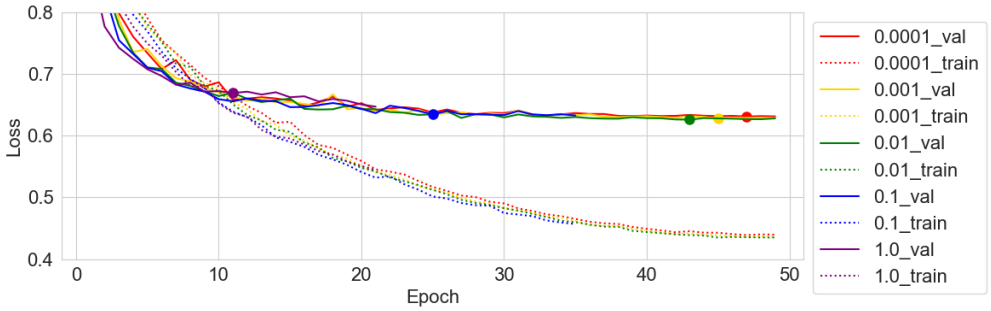


(c) 損失の合計値

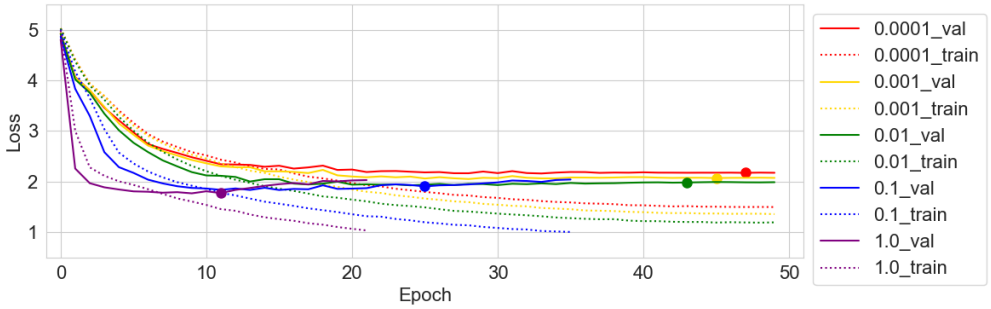
図 4.6: ベースラインにおける学習曲線



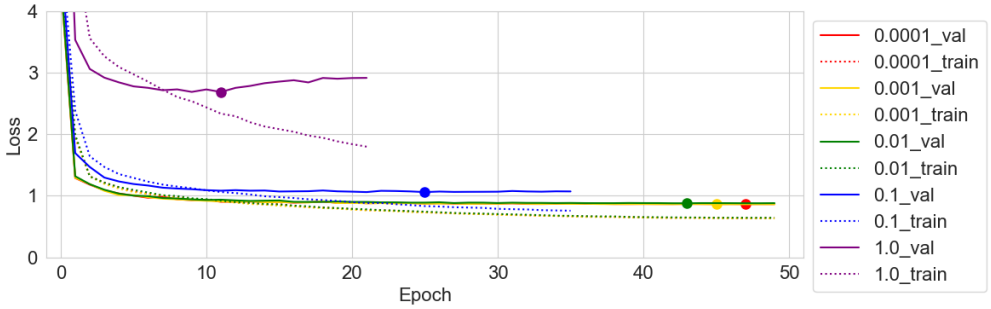
(a) HuBERT 中間特徴量についての MAE Loss



(b) メルスペクトログラムについての MAE Loss

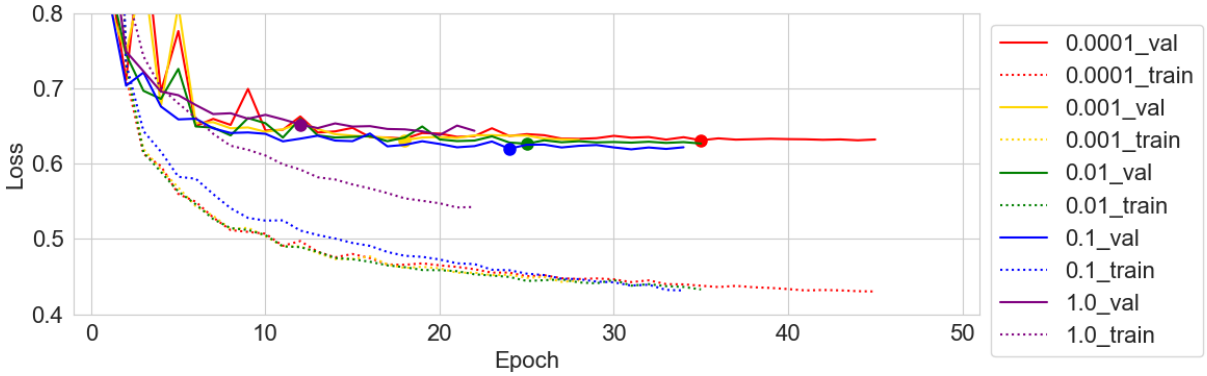


(c) HuBERT 離散特徴量についての Cross Entropy Loss

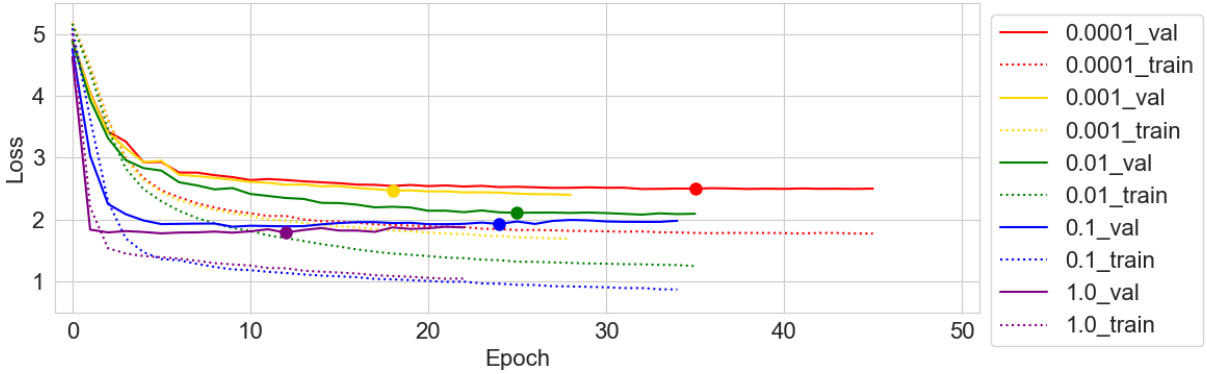


(d) 損失の合計値

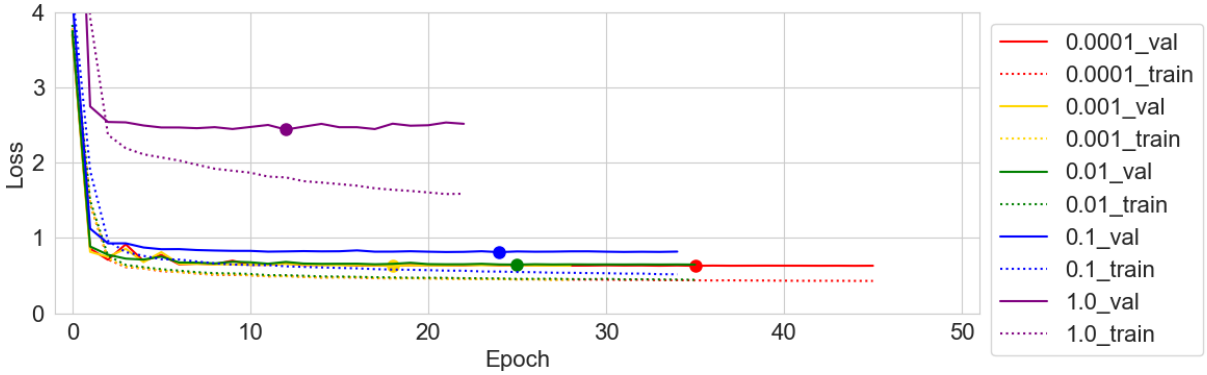
図 4.7: ネットワーク A における学習曲線



(a) メルスペクトログラムについての MAE Loss

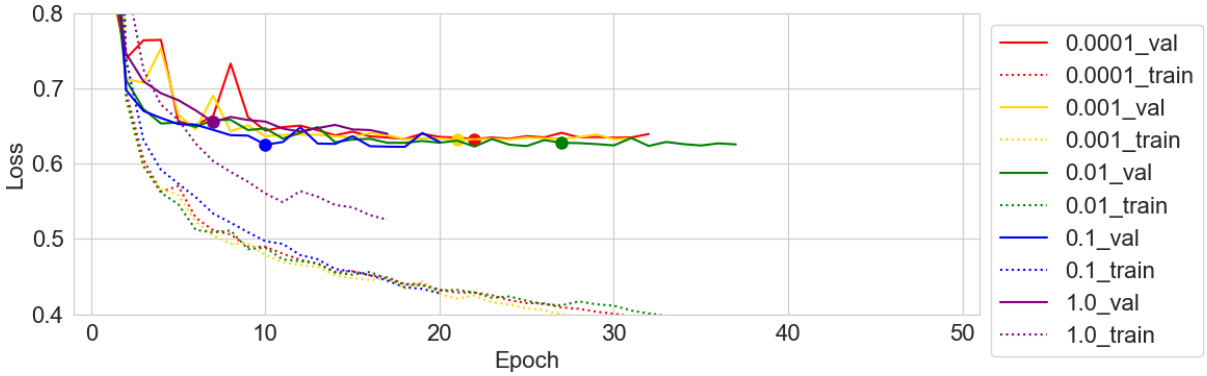


(b) HuBERT 離散特徴量についての Cross Entropy Loss

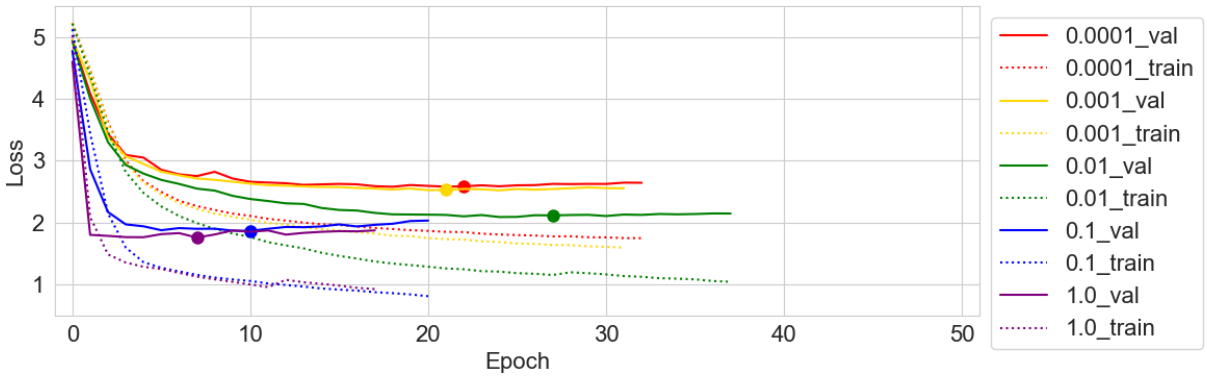


(c) 損失の合計値

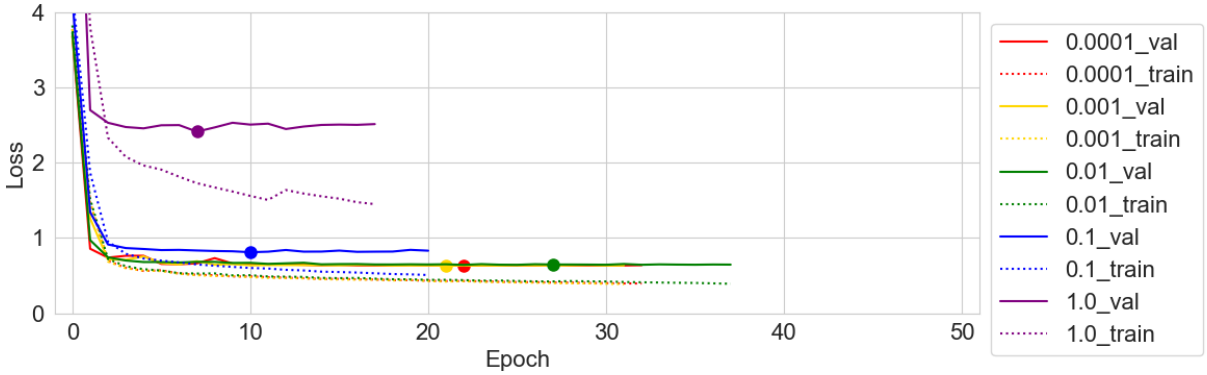
図 4.8: ネットワーク B（Randomized）における学習曲線



(a) メルスペクトログラムについての MAE Loss

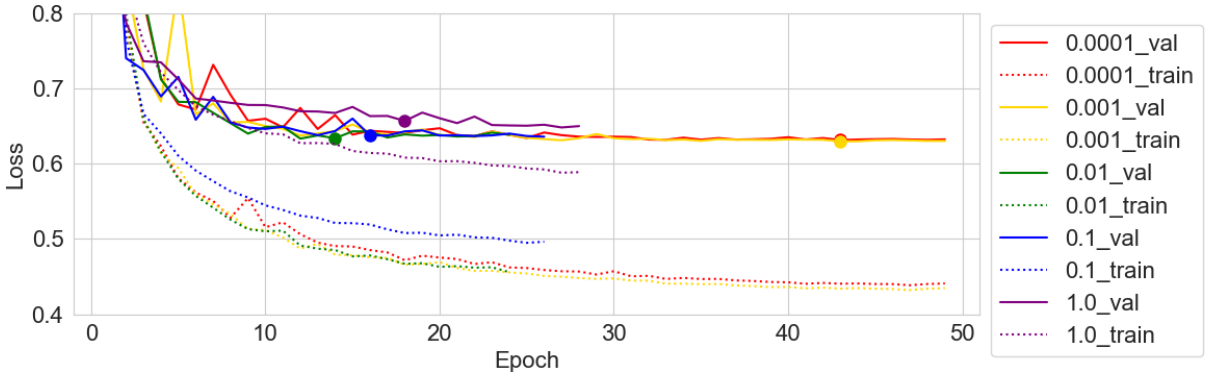


(b) HuBERT 離散特徴量についての Cross Entropy Loss

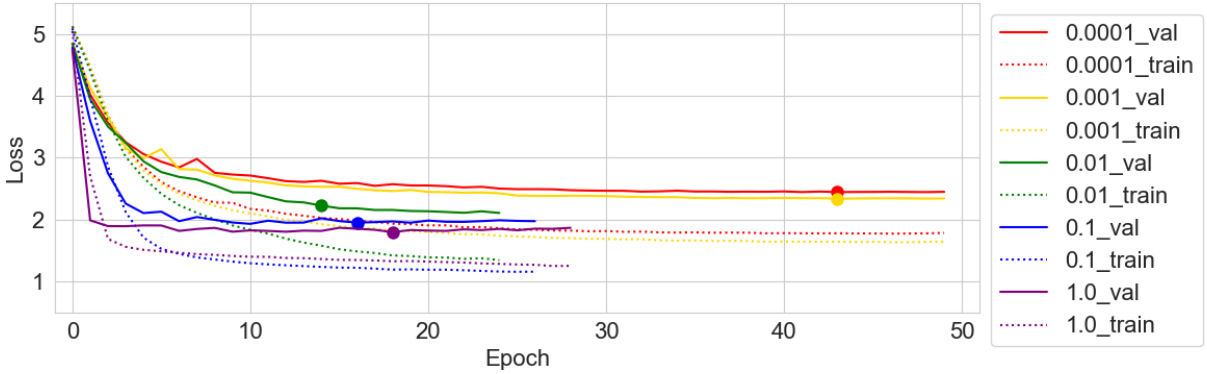


(c) 損失の合計値

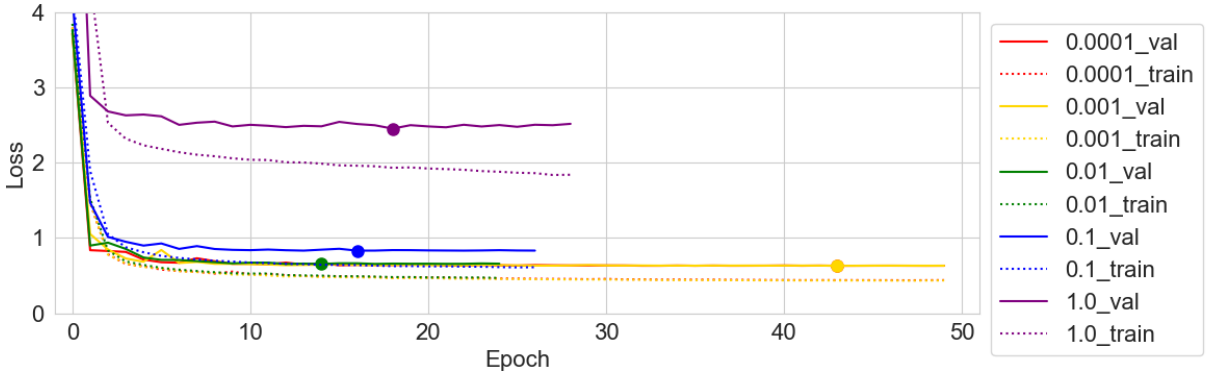
図 4.9: ネットワーク B（Pretrained）における学習曲線



(a) メルスペクトログラムについての MAE Loss

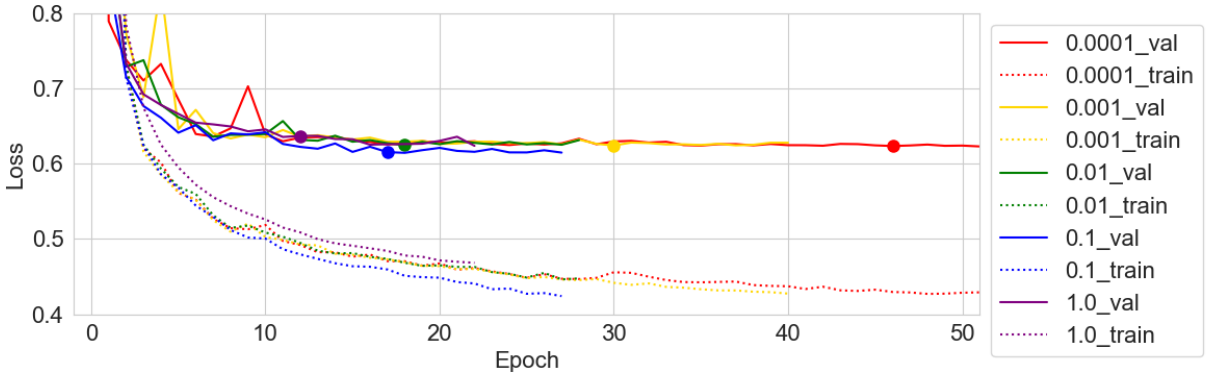


(b) HuBERT 離散特徴量についての Cross Entropy Loss

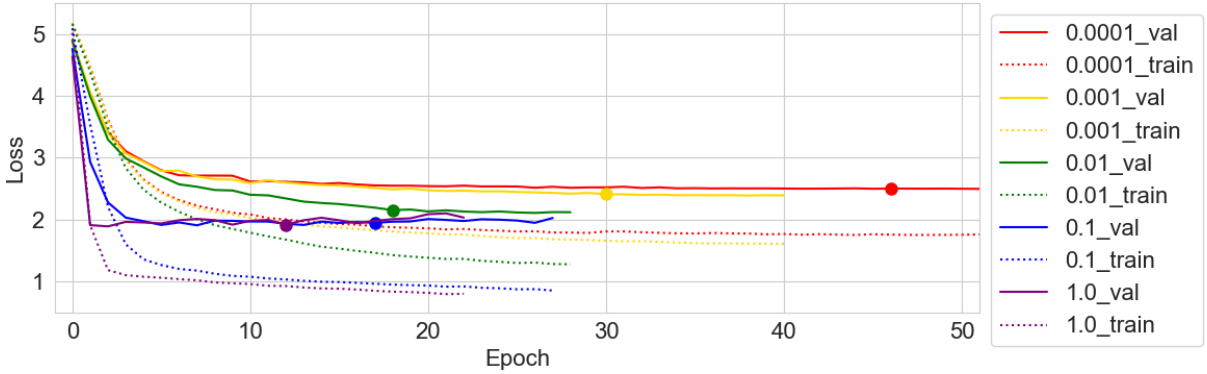


(c) 損失の合計値

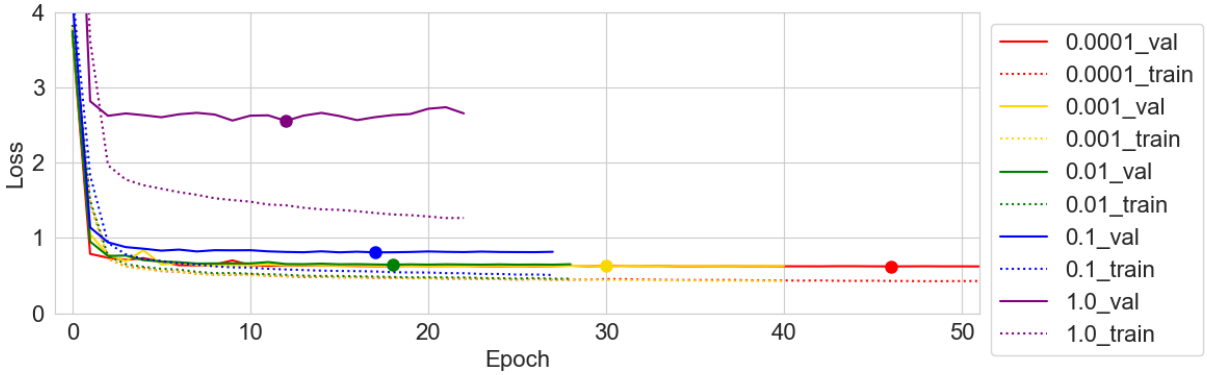
図 4.10: ネットワーク B（Randomized・Mel-HuB）における学習曲線



(a) メルスペクトログラムについての MAE Loss



(b) HuBERT 離散特徴量についての Cross Entropy Loss



(c) 損失の合計値

図 4.11: ネットワーク B（Randomized・A-SingleTask）における学習曲線

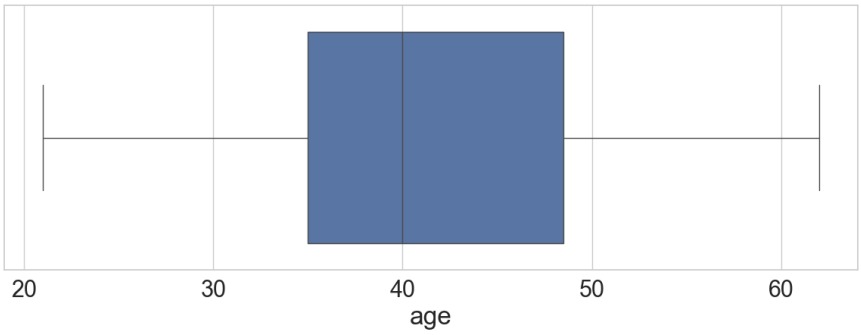


図 4.12: 主観評価実験における被験者の年齢層

4.5　まとめ

5　結論

謝辞

参考文献

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

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

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

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

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

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

[13] 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.

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

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

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] Yukiya Hono, Kentaro Mitsui, and Kei Sawada. rinna/japanese-hubert-base.

[34] 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.

[35] 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.

[36] 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.