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

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．

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

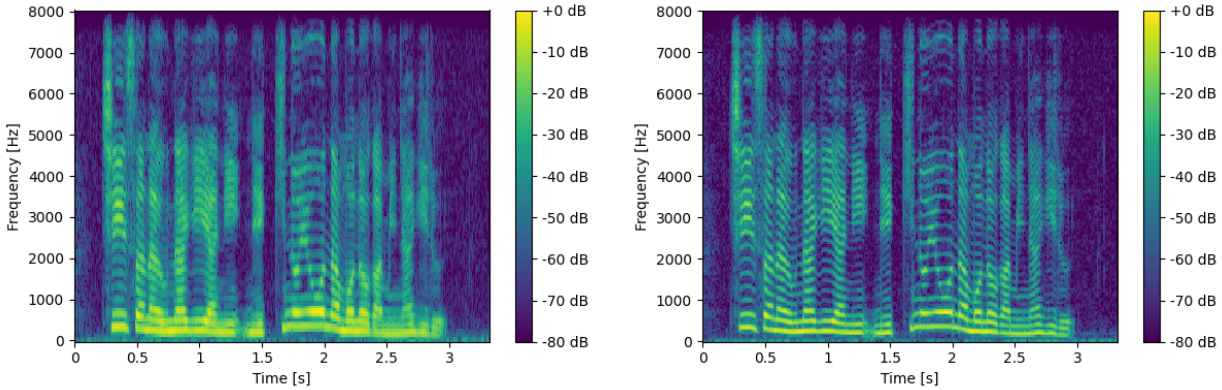
$$= |X[k]|e^{j\angle \mathrm{X[k]}} \tag{2.2}$$

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．

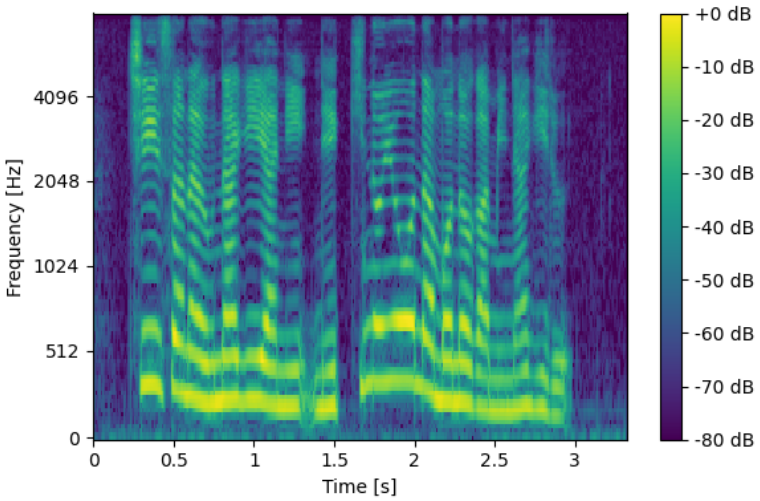


図 2.2: 「小さな鰻屋に，熱気のようなものがみなぎる」と発話した音声に対する対数メルスペクトログラム

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．

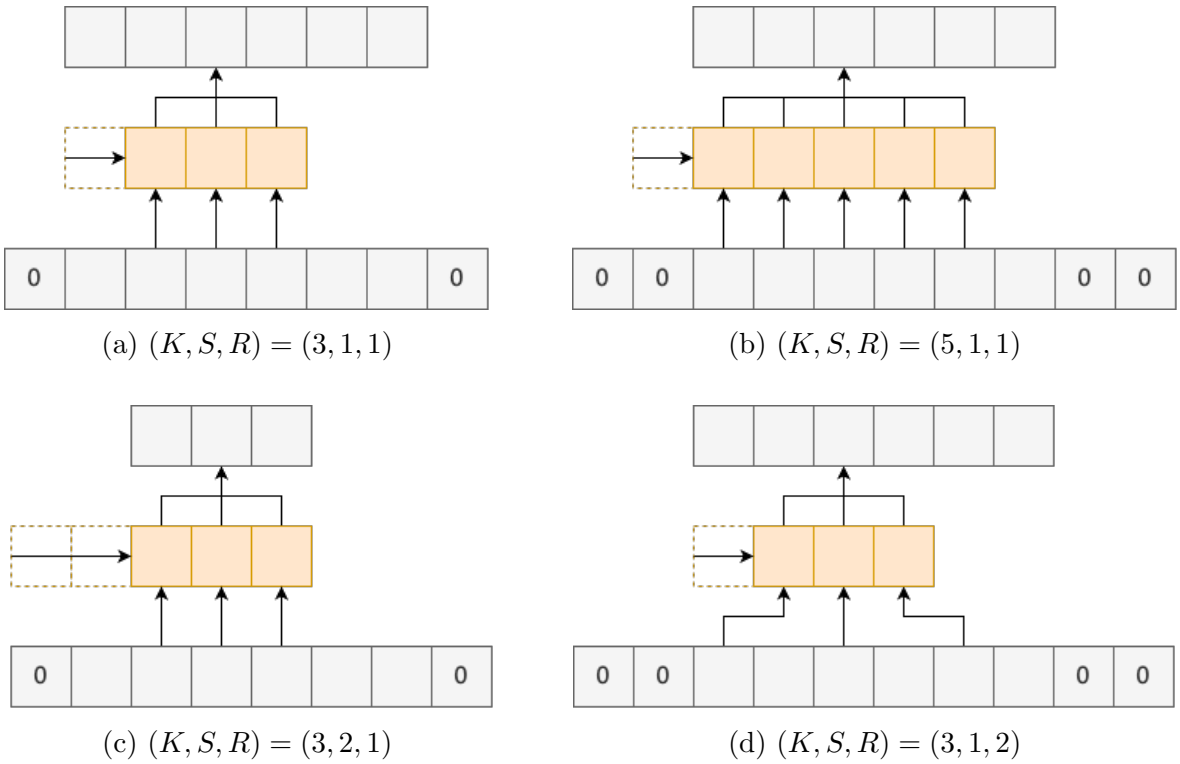
3.1.2　畳み込み層

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) $(K, S, R) = (5, 1, 1)$

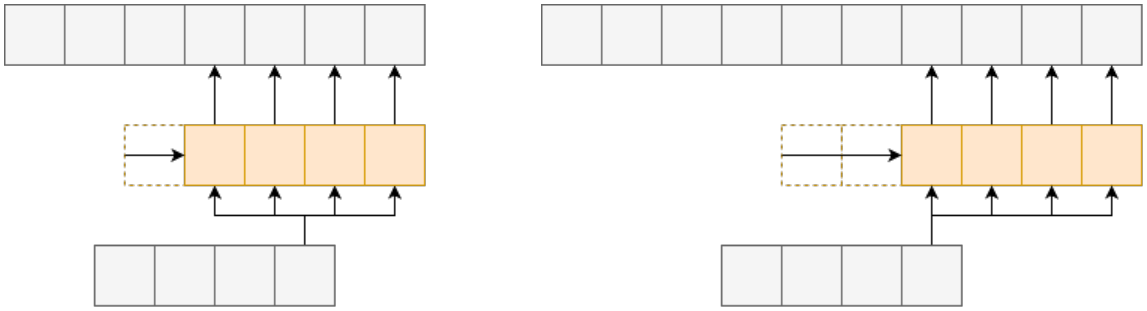
(c) $(K, S, R) = (3, 2, 1)$　(d) $(K, S, R) = (3, 1, 2)$

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．

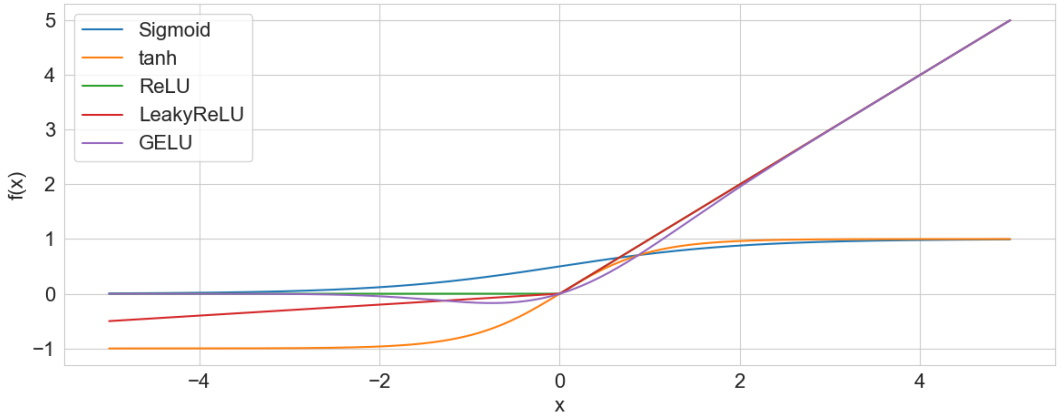
3.1.5 再帰型ニューラルネットワーク

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．

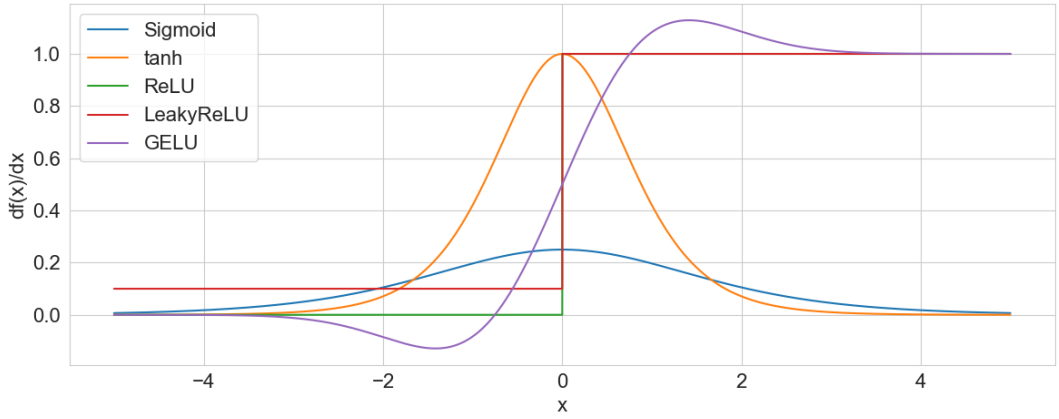
近年よく用いられる RNN として，長・短期記憶（Long Short-Time Memory; LSTM）[9] がある．LSTM は入力ゲート，忘却ゲート，出力ゲートの 3 つを持ち，これらゲートによってネットワーク内部の情報の取捨選択を行うことで，長い系列データからの学習を可能にした．LSTM のネットワーク内部で行われる計算を以下に示す．

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

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．また，$[\cdot,\cdot]$ は次元方向の結合，$\odot$ は要素積を表す．$\boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_h, \boldsymbol{W}_o \in \mathbb{R}^{D_{\text{out}} \times (D_{\text{in}}+D_{\text{out}})}$ 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，

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

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K \oplus \boldsymbol{b}_K \tag{3.40}$$

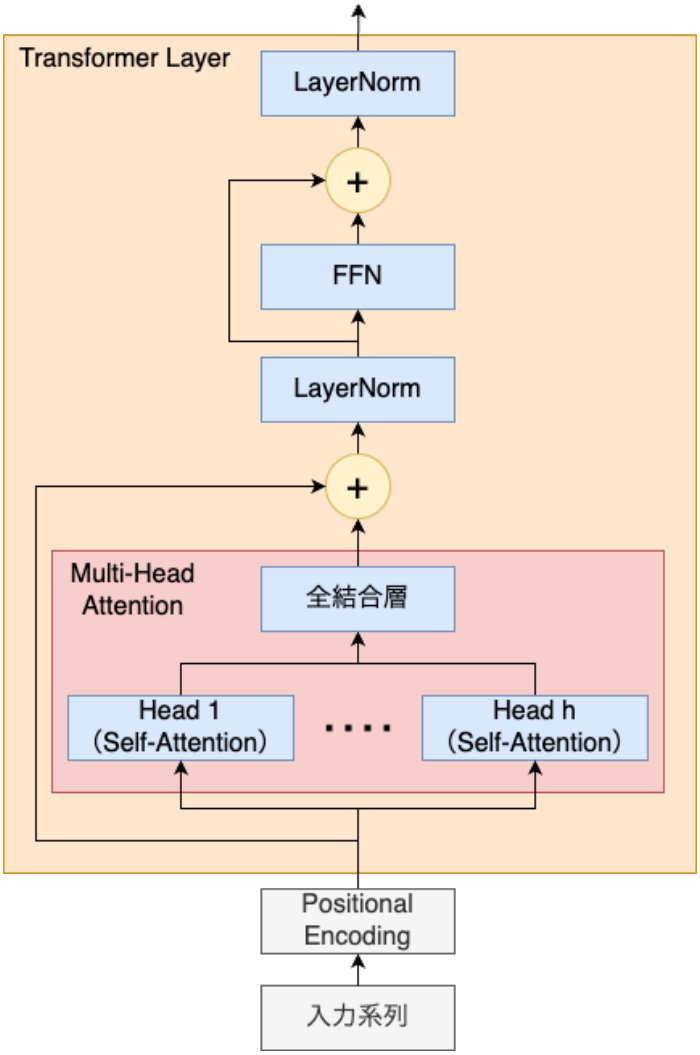


図 3.4: Transformer 層の構造

$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \oplus \boldsymbol{b}_V \tag{3.41}$$

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，演算子 $\oplus$ について，$\boldsymbol{A} \in \mathbb{R}^{D_1 \times D_2}$，$\boldsymbol{b} \in \mathbb{R}^{D_2}$ に対する $\boldsymbol{A} \oplus \boldsymbol{b} \in \mathbb{R}^{D_1 \times D_2}$ の (i, j) 成分 $(\boldsymbol{A} \oplus \boldsymbol{b})_{i,j}$ は，

$$(\boldsymbol{A} \oplus \boldsymbol{b})_{i,j} = a_{i,j} + b_j \tag{3.42}$$

で与えられる．ここで，$a_{i,j}$ は $\boldsymbol{A}$ の (i, j) 成分，b_j は $\boldsymbol{b}$ の j 成分である．

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，

$$\text{Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D_k}}\right)\boldsymbol{V} \tag{3.43}$$

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 $\boldsymbol{k}_t \in \mathbb{R}^{D_k}$，$rt = 1, \ldots, T$ 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，

$$\boldsymbol{Q}^h = \boldsymbol{X}\boldsymbol{W}_Q^h \oplus \boldsymbol{b}_Q^h \tag{3.44}$$

$$\boldsymbol{K}^h = \boldsymbol{X}\boldsymbol{W}_K^h \oplus \boldsymbol{b}_K^h \tag{3.45}$$

$$\boldsymbol{V}^h = \boldsymbol{X}\boldsymbol{W}_V^h \oplus \boldsymbol{b}_V^h \tag{3.46}$$

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，

$$\text{Attention}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right) = \text{softmax}\left(\frac{\boldsymbol{Q}^h r {\boldsymbol{K}^h}^\top}{\sqrt{D_k}/H}\right)\boldsymbol{V}^h \tag{3.47}$$

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，

$$\text{MultiHead}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \left[\text{Attention}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right), \ldots, \text{Attention}\left(\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\right)\right]\boldsymbol{W}_o \oplus \boldsymbol{b}_o \tag{3.48}$$

と表される．ここで，$[\cdot, \ldots, \cdot]$は次元方向の結合を表し，$\boldsymbol{W}_o \in \mathbb{R}^{D_v\times D_{\text{model}}}$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，

$$\boldsymbol{Y} = \text{LayerNorm}\left(\text{MultiHead}r\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} + \boldsymbol{X}\right) \tag{3.49}$$

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，

$$\boldsymbol{Y} = \text{LayerNorm}\left(\left(\text{ReLU}\left(\boldsymbol{Y}\boldsymbol{W}_1 \oplus \boldsymbol{b}_1\right)\boldsymbol{W}_2 \oplus \boldsymbol{b}_2\right) + \boldsymbol{Y}\right) \tag{3.50}$$

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．$D_{\text{fc}}$は$D_{\text{model}}$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，

$$\text{PositionalEncoding}\left(t, d\right) = \begin{cases} \sin\left(\frac{t}{10000^{2d/D_{\text{model}}}}\right) & \text{if } d \bmod 2 = 0 \\ \cos\left(\frac{t}{10000^{2d/D_{\text{model}}}}\right) & \text{if } d \bmod 2 = 1 \end{cases} \tag{3.51}$$

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

$$L_{\text{MAE}}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) = \frac{1}{D}\sum_{d=1}^{D}\left|y_d - \hat{y}_d\right| \tag{3.52}$$

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，

$$L_{\text{MAE}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = \frac{1}{DT}\sum_{d=1}^{D}\sum_{t=1}^{T}\left|y_{d,t} - \hat{y}_{d,t}\right| \tag{3.53}$$

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

$$L_{\text{CE}}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) = -\sum_{c=1}^{C} y_c \log\frac{\exp\left(\hat{y}_c\right)}{\sum_{i=1}^{C}\exp\left(\hat{y}_c\right)} \tag{3.54}$$

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，

$$\sum_{c=1}^{C} y_c = 1 \tag{3.55}$$

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，

$$L_{\text{CE}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C} y_{c,t} \log\frac{\exp\left(\hat{y}_{c,t}\right)}{\sum_{i=1}^{C}\exp\left(\hat{y}_{c,t}\right)} \tag{3.56}$$

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，

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} L(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} L(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})) \tag{3.57}$$

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

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{N_{\text{model}}}} \mathcal{L}(\boldsymbol{\theta}; \{1, \dots, N\}) \tag{3.58}$$

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，

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

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，

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

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，

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

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，

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}) \tag{3.62}$$

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

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{I}) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right)\right) + \frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2 \tag{3.63}$$

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，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

$$\boldsymbol{\theta}_{\tau^{\mathrm{iter}}} = \boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\mathrm{iter}}-1};\mathcal{I}_{\tau^{\mathrm{iter}}}\right) - \lambda'\boldsymbol{\theta} \tag{3.64}$$

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，

$$\boldsymbol{y}_{\mathrm{train}} = \frac{\boldsymbol{x}\odot\boldsymbol{m}}{1-p} \tag{3.65}$$

$$\boldsymbol{y}_{\mathrm{infer}} = \boldsymbol{x} \tag{3.66}$$

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 (3.65) 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，

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

$$= \frac{\boldsymbol{x}}{1-p}\odot\mathbb{E}\left[\boldsymbol{m}\right] \tag{3.68}$$

$$= \frac{\boldsymbol{x}}{1-p}\odot(1-p\ \cdots\ 1-p)^{\top} \tag{3.69}$$

$$= \boldsymbol{x} = \boldsymbol{y}_{\mathrm{infer}} \tag{3.70}$$

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，$\boldsymbol{g}_{\tau^{\text{iter}}} \in \mathbb{R}^{N_{\text{model}}}$ は勾配の一次モーメント，$\boldsymbol{m}_{\tau^{\text{iter}}} \in \mathbb{R}^{N_{\text{model}}}$ が勾配の一次モーメントの指数移動平均，$\boldsymbol{v}_{\tau^{\text{iter}}} \in \mathbb{R}^{N_{\text{model}}}$ が勾配の二次モーメントの指数移動平均である．$\hat{\boldsymbol{m}}_{\tau^{\text{iter}}}$ および $\hat{\boldsymbol{v}}_{\tau^{\text{iter}}}$ はそれぞれ，$\boldsymbol{m}_{\tau^{\text{iter}}}$ および $\boldsymbol{v}_{\tau^{\text{iter}}}$ 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, $\boldsymbol{\theta}_0$, $L(\boldsymbol{\theta})$
Initialize: $\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$
for $\tau^{\text{iter}} = 1$ to $\ldots$ **do**
  $\boldsymbol{g}_{\tau^{\text{iter}}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}\right) + \lambda \boldsymbol{\theta}_{\tau^{\text{iter}}-1}$
  $\boldsymbol{m}_{\tau^{\text{iter}}} = \beta_1 \boldsymbol{m}_{\tau^{\text{iter}}-1} + (1-\beta_1)\, \boldsymbol{g}_{\tau^{\text{iter}}}$
  $\boldsymbol{v}_{\tau^{\text{iter}}} = \beta_2 \boldsymbol{v}_{\tau^{\text{iter}}-1} + (1-\beta_2)\, \boldsymbol{g}_{\tau^{\text{iter}}} \odot \boldsymbol{g}_{\tau^{\text{iter}}}$
  $\tilde{\boldsymbol{m}}_{\tau^{\text{iter}}} = \boldsymbol{m}_{\tau^{\text{iter}}} / \left(1 - \beta_1^{\tau^{\text{iter}}}\right)$
  $\tilde{\boldsymbol{v}}_{\tau^{\text{iter}}} = \boldsymbol{v}_{\tau^{\text{iter}}} / \left(1 - \beta_2^{\tau^{\text{iter}}}\right)$
  $\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \tilde{\boldsymbol{m}}_{\tau^{\text{iter}}} / \left(\sqrt{\tilde{\boldsymbol{v}}_{\tau^{\text{iter}}}} + \epsilon\right)$
end for
Return $\boldsymbol{\theta}_{\tau^{\text{iter}}}$

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, $\boldsymbol{\theta}_0$, $L(\boldsymbol{\theta})$
Initialize: $\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$
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

$$\eta_{\tau^{\text{epoch}}} = \eta_0 \gamma^{\left\lfloor \tau^{\text{epoch}} / \text{step\_size} \right\rfloor} \tag{3.71}$$

で与えるスケジューラである．これは，学習が step_size 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

$$\eta_{\tau^{\text{epoch}}} = \eta_0 \exp\left(-\gamma \tau^{\text{epoch}}\right) \tag{3.72}$$

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

$$\eta_{\tau^{\text{epoch}}} = \begin{cases} \eta_{\min} + \left(\frac{\tau^{\text{epoch}}}{\text{warmup\_steps}}\right)(\eta_{\max} - \eta_{\min}) & \text{if } \tau^{\text{epoch}} < \text{warmup\_steps} \\ \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{\left(\tau^{\text{epoch}} - \text{warmup\_steps}\right)\pi}{\tau_{\max}^{\text{epoch}} - \text{warmup\_steps}}\right)\right) & \text{if } \tau^{\text{epoch}} \geq \text{warmup\_steps} \end{cases} \tag{3.73}$$

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．warmup_steps 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 warmup_steps 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．

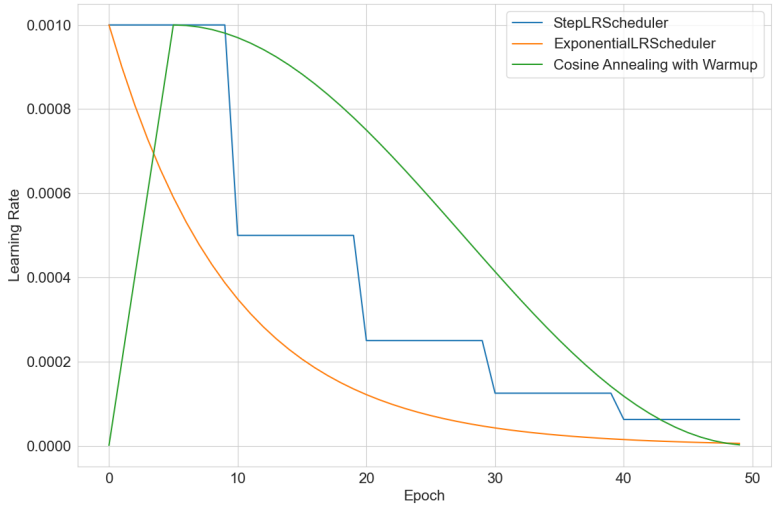


図 3.5: スケジューラによる学習率の変化

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，

$$a_p^n = b_p^n + \sum_{q=1}^{D_{n-1}} w_{q,p}^n o_q^{n-1} \tag{3.74}$$

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，

$$o_q^{n-1} = \phi\left(a_q^{n-1}\right) \tag{3.75}$$

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 (3.74) に対し，$w_{0,p}^n = b_p^n$，$o_0^{n-1} = 1$ とおけば，

$$a_p^n = b_p^n + \sum_{q=1}^{D_{n-1}} w_{q,p}^n o_q^{n-1} = \sum_{q=0}^{D_{n-1}} w_{q,p}^n o_q^{n-1} \tag{3.76}$$

と整理できる．この時，

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

$$= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\partial L\left(\boldsymbol{y}_i, f\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right)\right)}{\partial w_{p,q}^n} \tag{3.78}$$

$$= \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\partial L_i}{\partial w_{p,q}^n} \tag{3.79}$$

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 (3.76) を用いて，

$$\frac{\partial L_i}{\partial w_{p,q}^n} = \frac{\partial L_i}{\partial a_q^n} \frac{\partial a_q^n}{\partial w_{p,q}^n} \tag{3.80}$$

$$= \delta_q^n \frac{\partial}{\partial w_{p,q}^n} \left(\sum_{r=0}^{D_{n-1}} w_{r,q}^n o_r^{n-1} \right) \tag{3.81}$$

$$= \delta_q^n o_p^{n-1} \tag{3.82}$$

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，

$$\frac{\partial L_i}{\partial w_{p,q}^{N_{\text{layer}}}} = \delta_q^{N_{\text{layer}}} o_p^{N_{\text{layer}}-1} \tag{3.83}$$

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L_i}{\partial a_q^{N_{\text{layer}}}} \tag{3.84}$$

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

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

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

$$= o_p^{N_{\text{layer}}-1} \frac{\partial L\left(\boldsymbol{y}_i, \phi\left(\boldsymbol{a}^{N_{\text{layer}}}\right)\right)}{\partial \phi\left(a_q^{N_{\text{layer}}}\right)} \phi'\left(a_q^{N_{\text{layer}}}\right) \tag{3.88}$$

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，

$$\frac{\partial L_i}{\partial w_{p,q}^n} = \delta_q^n o_p^{n-1} \tag{3.89}$$

$$= o_p^{n-1} \frac{\partial L_i}{\partial a_q^n} \tag{3.90}$$

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \frac{\partial L_i}{\partial a_r^{n+1}} \frac{\partial a_r^{n+1}}{\partial a_q^n} \tag{3.91}$$

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

$$= o_p^{n-1} \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} \frac{\partial}{\partial a_q^n} \left(\sum_{s=0}^{D_n} w_{s,r}^{n+1} \phi\left(a_s^n\right) \right) \tag{3.93}$$

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

$$= o_p^{n-1} \phi'\left(a_q^n\right) \sum_{r=0}^{D_{n+1}} \delta_r^{n+1} w_{q,r}^{n+1} \tag{3.95}$$

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，

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right) \leftarrow \frac{c}{\max\{\left\|\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right)\right\|_2, c\}} \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{I}\right) \tag{3.96}$$

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

$$\boldsymbol{g}_{\text{accum}} \leftarrow \boldsymbol{g}_{\text{accum}} + \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}\right) \tag{3.97}$$

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

$$\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \frac{\eta}{N_{\text{accum}}} \boldsymbol{g}_{\text{accum}} \tag{3.98}$$

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，

$$\boldsymbol{x}_t^{\text{masked}} = \begin{cases} \boldsymbol{x}_t & \text{if } t \notin \mathcal{M} \\ \boldsymbol{m} & \text{if } t \in \mathcal{M} \end{cases} \tag{3.99}$$

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，

$$L_{\text{CE-masked}}\left(\boldsymbol{Y}, \hat{\boldsymbol{Y}}\right) = -\frac{1}{|\mathcal{M}|}\sum_{t\in\mathcal{M}}\sum_{c=1}^{C} y_{t,c}\log\left(\hat{y}_{t,c}\right) \tag{3.100}$$

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 $\boldsymbol{y}_n^{\text{sp-wf}}$ から計算された話者埋め込みベクトル，$\boldsymbol{Y}_n^{\text{mel}} \in \mathbb{R}^{D^{\text{mel}} \times T_n^{\text{mel}}}$ は $\boldsymbol{y}_n^{\text{sp-wf}}$ から計算された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 $\boldsymbol{y}_n^{\text{sp-wf}}$ から計算された HuBERT 中間特徴量，$\boldsymbol{Y}_n^{\text{HuB-disc}} \in \{0,1\}^{C \times T_n^{\text{HuB}}}$ は $\boldsymbol{y}_n^{\text{sp-wf}}$ から計算された HuBERT 離散特徴量とする．ここで，話者埋め込めベクトル $\boldsymbol{x}_{s_n}^{\text{spk-emb}}$ は，事前学習済みの話者識別モデル [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．$\mathcal{I}$ 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、

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

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，

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

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

で与えられる．ここで，$\boldsymbol{\theta}_{\text{HuB}}$ は HuBERT の事前学習済み重みを表し，$f_{\text{HuB-conv}}, f_{\text{HuB-trans}}$ にこの重みを読み込んで処理したことを意味する．$f_{\text{k-means}}$ は本実験の主な検討対象である DNN ではないため，ここでは単にクラスタリングを行う関数として表記した．上記の $\boldsymbol{y}_n^{\text{sp-wf}}$ から計算される特徴量は，すべてモデルの学習を開始する以前に計算されているものとする．

提案手法を図??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，

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

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，

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

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，

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

と表される．まとめると，提案手法は口唇動画と話者埋め込み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 では，まず，口唇動画 $\boldsymbol{X}_n^{\text{video}}$ を AVHuBERT に通すことで，特徴量 $\boldsymbol{H}_n^{\mathrm{A}} \in \mathbb{R}^{D^{\mathrm{A}} \times T_n^{\text{video}}}$ に変換する．これは，

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

と表される．次に，$\boldsymbol{H}_n^{\mathrm{A}}$ の各時刻 t におけるベクトルに対し，話者埋め込みベクトル $\boldsymbol{x}_{s_n}^{\text{spk-emb}}$ 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，

$$\boldsymbol{H}_n^{\mathrm{A}} = f_{\text{spk-merge}}\left(\operatorname{concat}\left(\boldsymbol{H}_n^{\mathrm{A}}, \left(\boldsymbol{x}_{s_n}^{\text{spk-emb}}\right)_T\right); \boldsymbol{\theta}_{\mathrm{A}}\right) \tag{4.10}$$

と表される．次に，話者埋め込み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，

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

と表される．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獲る．これは，

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

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

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

と表される．

4.1.3 ネットワーク B

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，

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

と表される．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．

$$\boldsymbol{H}_n^{\text{B}} = f_{\text{spk-merge}}\left(\text{concat}\left(\boldsymbol{H}_n^{\text{B}}, \text{repeat}\left(\boldsymbol{x}_{s_n}^{\text{spk-emb}}\right)\right); \boldsymbol{\theta}_{\text{B}}\right) \tag{4.16}$$

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

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

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

ネットワーク A の処理と見比べていただくと，$f_{\text{spk-merge}}, f_{\text{post}}, f_{\text{fc-mel}}, f_{\text{fc-HuB-disc}}$ は同じ関数で重みのみ異なることがわかる．これは，これらのネットワーク構造が同一であり，学習される重みのみが異なることを表す．

4.1.4 ボコーダ

ボコーダでは，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，

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

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

と表される．前処理後の特徴量 $\boldsymbol{H}_n^{\text{mel-voc}} \in \mathbb{R}^{D^{\text{mel-voc}} \times T_n^{\text{HuB}}}$ 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，系列長を半減しつつ，次元を 2 倍に拡大したものである．一方，前処理後の特徴量 $\boldsymbol{H}_n^{\text{HuB-voc}} \in \mathbb{R}^{D^{\text{HuB-voc}} \times T_n^{\text{HuB}}}$ 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ものである．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，

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

と表される．$f_{\text{voc-main}}$ の具体的な処理について，これは HiFi-GAN の Generator と同様であり，本実験ではボコーダ自体ではなく，ボコーダ入力を予測する手法の検討に焦点を当てたから，ここでは詳細は割愛する．

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 中間特徴量と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．

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，男女二人ずつから収録した合計 4 人分のデータセット [22, 23] を用いた．これは ATR 音素バランス文 [24] から構成され，全話者共通で 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て利用した．各分割ごとの文章数を表 4.1 に示す．

Multi-input Vocoder の学習に利用する音声データセットには，Hi-Fi-Captain（日本語話者二名分）[25] と JVS（parallel100 と nonpara30）[26] を利用した．Hi-Fi-Captain 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て分割した．各分割ごとの文章数を表 4.1 に示す．JVS には話者に対して 1 から 100 まで番号が割り振られており，本実験では 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．また，JVS には読み上げ音声の parallel100 および nonpara30 と，裏声の falset10，囁き声の whisper10 が含まれる．本研究では，parallel100 と nonpara30 のみを利用した．

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．このランドマークを利用すること

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

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．正規化では，グレースケールが 0 から 255 までの値を取るため，最大値の 255 で割り，標準化では，AVHuBERT のプログラムで利用されていた平均値 0.421 と，標準偏差 0.165 をそのまま利用して，平均値を引いた後に標準偏差で割った．全体として，今回は事前学習済みの 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．学習時は，ランダムクロップ，左右反転，Time Masking（一時停止）をデータ拡張として適用した．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，連続する画像の時間平均値を利用することによって，一時停止させるような効果を与えるデータ拡張手法である．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．

音声データは 16 kHz にダウンサンプリングして用いた．それから，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た．さらに，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，メルスペクトログラムを得た上で対数スケールに変換した．また，Multi-input Vocoder の学習に利用した Hi-Fi-Captain と JVS については，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．なぜなら，Multi-input Vocoder の学習時は，全体から 1 秒分をランダムサンプリングするように実装しており，元データに存在する無音区間を除去することによって，学習の安定化および得られる音声の品質改善に繋がったからである．また，話者 Embedding の取得には事前学習済みの話者識別モデル [21] を利用した．動画音声データセット，Hi-Fi-Captain，JVS ともに，各話者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．この値を学習・検証・テストで一貫して用いるため，学習外の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．モデルに入力する際には，ベクトルの大きさで割って正規化した．

HuBERT は，HuggingFace に公開されている ReazonSpeech というデータセットによって学習されたモデル [28, 29] を利用した．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，日本語音声のコンテキストを大量のデータから学習したモデルである．今回用いるデータセットが日本語であることから，本研究の検討対象としては日本語音声に関する事前知識を豊富に有するモデルが適していると考え，このモデルを選択した．本研究で用いる

HuBERT 中間特徴量および HuBERT 離散特徴量について，HuBERT 中間特徴量は，音声波形に対して畳み込み層を適用した出力で，HuBERT の事前学習時にはマスクの対象となる特徴量を利用した．一方，HuBERT 離散特徴量は，HuBERT Transformer 層の 8 層目出力に，k-means 法によるクラスタリングを適用することで得た．あえて 8 層目出力を選択した理由は，HuBERT のレイヤー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 [30] より，8 層目出力がそのどちらとも相関が高く言語的な情報に近いと判断したからである．クラスタ数については決め打ちとなるが，今回は離散化した結果が言語的な情報を持つことを狙い，比較的少ない 100 とした．k-means 法の学習には動画音声データセットにおける学習用データ全てを利用し，これを用いて動画音声データ，Hi-Fi-Captain，JVS に対するクラスタリングを実施した．

4.2.3 学習方法

一段階目について，損失関数はメルスペクトログラムの MAE Loss L_{mel} と HuBERT 離散特徴量の Cross Entropy Loss L_{ssl^d} ，HuBERT 中間特徴量の MAE Loss L_{ssl^i} の重み付け和とした．それぞれの重み係数を $\lambda_{mel}, \lambda_{ssl^d}, \lambda_{ssl^i}$ とすると，

$$L = \lambda_{mel} * L_{mel} + \lambda_{ssl^d} * L_{ssl^d} + \lambda_{ssl^i} * L_{ssl^i} \tag{4.25}$$

となる．最適化手法には AdamW[18] を利用し，$\beta_1 = 0.9$，$\beta_2 = 0.98$，$\lambda = 0.01$ とした．学習率に対するスケジューラには，Cosine Annealing with Warmup を利用した．開始時の学習率は 1.0×10^{-6} として，最大エポック数の 10%に至るまでは学習率を 1.0×10^{-3} まで線形に増加させ，その後のエポックでは cosine 関数に基づいて 10×10^{-6} まで減少させた．バッチサイズはメモリの都合上 4 としたが，学習の安定化のため，Gradient Accumulation によって各イテレーションにおける勾配を累積させ，8 イテレーションに一回重みを更新するようにした．モデルに入力する動画の秒数は 10 秒を上限とし，それを超える場合はランダムにトリミング，それに満たない場合はゼロパディングした．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することで，過度に大きくなることを防止した．最大エポック数は 50 とし，10 エポック連続して検証データに対する損失が小さくならない場合には，学習を中断するようにした（Early Stopping）．また，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，これをテストデータに対する評価に用いた．

第二段階について，損失関数はメルスペクトログラムの MAE Loss と HuBERT 離散特徴量の Cross Entropy Loss の重み付け和とした．これは式 (4.25) において，$\lambda_{ssl^i} = 0.0$ と固定した場合に相当する．最適化手法には AdamW を利用し，$\beta_1 = 0.9$，$\beta_2 = 0.98$，$\lambda = 0.01$ とした．学習率に対するスケジューラには，Cosine Annealing with Warmup を利用した．開始時の学習率は 1.0×10^{-6} として，最大エポック数の 10%に至るまでは学習率を 5.0×10^{-4} まで線形に増加させ，その後のエポックでは cosine 関数に基づいて 10×10^{-6} まで減少させた．学習率の最

大値は第一段階の 1/2 となっているが，これは値を半減させることによって学習を安定させることができたためである．その他のパラメータは，第一段階における値と同じである．

第三段階について，Cosine Annealing with Warmup における最大学習率のみ 1.0×10^{-3} としたが，それ以外は第二段階と同様である．

Multi-input Vocoder の学習では，Hi-Fi-Captain と JVS を用いた．はじめに Hi-Fi-Captain のみを用いて学習させ，その後学習済みモデルを JVS によって再学習した．損失関数は HiFi-GAN と同様である．二つのデータセットを用いた理由について，Hi-Fi-Captain は男女一人ずつの文章数が豊富なデータセットであるため，高品質なモデルを構築可能であった．しかし，学習できる話者数が少ない分，学習外話者に対する合成音声の品質が低かった．そのため，一人当たりの文章数は 100 文章程度と少ないながらも，100 人分の話者からなる JVS を利用して再学習することによって，学習外話者に対する合成音声の品質を向上させた．最適化手法には AdamW を利用し，$\beta_1 = 0.8$，$\beta_2 = 0.99$，$\lambda = 1.0 \times 10^{-5}$ とした．学習率は 2.0×10^{-4} から開始し，1 エポック経過するごとに 0.99 かけて徐々に減衰させた．バッチサイズは 16 とし，ここでは Gradient Accumulation は利用しなかった．モデルへの入力は 1 秒を上限とし，それを超える場合はランダムにトリミング，それに満たない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することで，過度に大きくなることを防止した．最大エポック数は 30 とし，ここでは Early Stopping は適用しなかった．また，学習終了時には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 L1 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，Multi-input Vocoder の提案された先行研究 [5] においては，学習時にあえてメルスペクトログラムにノイズをかけることによって，合成音声に対する汎化性能を向上させる学習方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，Multi-input Vocoder の学習は原音声から計算される特徴量そのもので行い，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．

4.2.4 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，

$$\mathrm{WER}\left(s_1, s_2\right) = \frac{I + D + R}{|s_1|} \tag{4.26}$$

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[31] を利用し，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%から 100%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 Embedding 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，サンプルごとに評価対象音声の話者 Embedding と原音声の話者 Embedding のペアで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 0 から 1 であり，高いほど原音声と類似した合成音声だと判断できる．

4.2.5 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，日本語母語話者であること，聴覚に異常がないこと，イヤホンあるいはヘッドホンを用いて静かな環境で実験を実施可能であることとした．被験者の方に行っていただいた項目は，以下の五つである．

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）

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．実験に使用した音響機器は，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．初めに練習試行を行っていただくことで実験内容を把握してもらい，その後本番施行を行っていただく流れとした．ここで，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二段階で構成した．

1. 音声サンプルのみを一回再生し，発話内容を聞き取ってもらう．
2. 本来の発話内容を確認してもらい，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 段階評価してもらう．

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，本番試行ではテストデータを用いた．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，method_names が手法名のリスト，speaker_names が話者名のリスト，num_total_respondents が被験者総数である．各被験者について，まず sentences と method_names をランダムにシャッフルし，それからランダムサンプリングした speaker_name を合わせて，一つのサンプルを決定するようになっている．この選択方法では，二つのことに注意した．一つ目は，各被験者がユニークな 53 文章を評価することである．評価の際に本来の発話内容を知ることになるため，それを知った上で同じ発話内容のサンプルが出現した場合，音声自体の明瞭性に関わらず発話内容がわかってしまう可能性があると考え，これを避けるようにした．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．今回は手法の比較が実験の目的となるため，各被験者がすべての手法を評価することが望ましいと判断した．また，評価に際し音声サンプルを一回だけ聞いてもらうようにした理由は，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ストレスになると考えられるため，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その手法の聞き取りやすさとして評価するべきだと考えたからである．

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．ここでも初めに練習試行を行っていただくことで実験内容を把握してもらい，その後本番施行を行っていただく流れとした．ここで，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てもらう．
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してもらう．

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

Algorithm 3 Sample Assignment Algorithm

Require: sentences: List of sentences
Require: method_names: List of method names
Require: speaker_names: List of speaker names
Require: num_total_respondents: Total number of respondents

Initialize all_assignments ← []
for respondent_id ← 1 to num_total_respondents **do**
    Initialize assignments ← []
    Randomly shuffle sentences
    Randomly shuffle method_names
    for $i \leftarrow 0$ to len (sentences) $- 1$ **do**
        sentence ← sentences[i]
        method_name ← method_names[i mod len (method_names)]
        speaker_name ← Randomly select from speaker_names
        Append (respondent_id, sentence, method_name, speaker_name) to assignments
    end for
    all_assignments ← all_assignments ∪ assignments
end for
return all_assignments

実験に利用した音声サンプルおよび，被験者ごとの評価サンプルの割り当て方法は明瞭性の評価実験と同様にした．ただし，類似性評価においては，評価対象となる音声に対して同一話者の原音声をランダムに選択し，評価対象となる音声とペアで提示できるようにした．また，明瞭性評価では音声サンプルを一回だけ聞いて評価してもらうようにしたが，類似性評価では何度でも聞けるようにした．聞き取りやすさのようにコミュニケーションを想定した評価というより，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．さらに，明瞭性評価では評価時に発話内容を提示したが，類似性評価は発話内容に依存しないため，提示しなかった．類似性評価は発話内容に依存しないと考えらえるからである．加えて，類似性評価は明瞭性評価を完了した後にしか実施できないようにした．類似性評価と明瞭性評価に用いるサンプルは同一の発話内容のパターンであり，特に類似性評価では原音声を聴取できることから，本来の発話内容を完全に把握できると予想される．その上で明瞭性評価を行った場合，音声自体の明瞭性に関わらず発話内容がわかってしまう可能性があると考えられ，望ましくない．一方，類似性評価は発話内容に依存しない評価であるため，発話内容を知っていることが評価に影響を与えないと考えられる．よって，今回は明瞭性評価の後に類似性評価を行うことにした．

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 [32] 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．具体例として，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「1: 全く同じ話者には聞こえなかった」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．この時，その音声自体の明瞭性や類似性とは無関係に，必ずこの音声によって指定された評価値を選択するよう説明を与えた．本番試行においてダミー音声で指定された評価値を誤って選んだ場合は，すべての回答を無効にする旨を被験者に伝えた．実際，実験終了後にはそのようにデータを処理した．

被験者数は 75 人とし，謝礼は実験一回あたり 40 分程度要すると予想し，650 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，以下の五つである．

1. ベースライン
2. ネットワーク A

3. ネットワークB（Not-Pretrained）
4. ネットワークC（Not-Pretrained）
5. ネットワークB（Pretrained）
6. ネットワークC（Pretrained）

ベースラインは，提案手法（図??）におけるネットワークAで，メルスペクトログラムとHuBERT離散特徴量を予測するマルチタスク学習手法である．ネットワークAは提案手法において，ネットワークBおよびCに入力を与える役割を果たすネットワークである．提案手法自体ではないが，その構成要素とはなっているため，ここでも客観評価指標を確認することとした．ネットワークB（Not-Pretrained）は，HuBERT Transformer層で事前学習済み重みを読み込まなかった場合のネットワークBである．これに関連して，ネットワークC（Not-Pretrained）はネットワークB（Not-Pretrained）を利用したネットワークCである．これらに対し，ネットワークB（Pretrained）およびネットワークC（Pretrained）は，ネットワークBの学習時にHuBERT Transformer層で事前学習済み重みを読み込んだ点のみ異なる．

まず，損失関数 (4.25) の重み係数 λ_{ssl^d} を変化させた時の，客観評価指標の全テストデータに渡る平均値を表 4.2 に示す．各手法ごとに λ_{ssl^d} を 0.0001 から 1.0 まで，10 倍刻みで 5 段階検討するグリッドサーチを行い，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．ここで，提案手法ではネットワークAを学習し，その後Aを固定してBを学習し，最後にAとBを固定してCを学習するように実装している．これに対し，λ^d_{ssl} のグリッドサーチでは，この一連の流れに対して一つの値を検討した．例えば，ネットワークB（Not-Pretrained）で λ_{ssl^d} が 0.0001 の場合，ネットワークAには λ_{ssl^d} が 0.0001 の場合を用いている．ネットワークC（Not-Pretrained）で λ_{ssl^d} が 0.0001 の場合，ネットワークA，Bともに λ_{ssl^d} が 0.0001 の場合を用いている．また，この後手法ごとの比較を行う際には，各手法ごとにグリッドサーチで得られた最適な場合同士を比較するため，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，λ_{ssl^d} 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．次に，図 4.1 にベースラインにおける学習曲線の結果を示す．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 λ_{ssl^d} の違いを表す．また，丸いマーカーは表 4.3 に示した最良エポック時における検証データに対する損失の値を表す．学習曲線より，λ_{ssl^d} の値を変化させることによって，特に L_{ssl^d} の傾向が変化していることがわかる．具体的には，λ_{ssl^d} の値を 0.0001 から 1.0 へと増加させるのに伴って，学習初期における損失の下がり方が急峻になっており，達する最小値自体が小さくなっていることがわかる．また，λ_{ssl^d} の値が 0.1 以上の場合，検証データに対する L_{ssl^d} は早いうちから増加傾向に転じている．これに伴い，今回は検証データに対する L の値を監視し，Early Stopping の適用と最良エポックの決定を行なったため，L_{mel} が下がり切らない状態で学習が中断される結果となった．離散化して冗長性を排除した HuBERT 離散特徴量と比較して，メルスペクトログラムが特に話者性を反映するために必要な特徴量だと考えられるため，

λ_{ssl^d} が 0.1 以上の場合に見られた話者類似度の顕著な低下は，L_{mel} を下げきれなかったことが原因だと考えられる．

ネットワーク A では，提案手法の構成要素であるため，特に最適な手法の選択は行なっていない．ベースラインとの違いは L_{ssl^i} が損失に含まれることであるが，客観評価指標より，ベースラインと性能は概ね同等であることがわかる．学習曲線の傾向については，ベースラインと同様であったため省略する．

ネットワーク B（Not-Pretrained）では，λ_{ssl^d} 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．次に，図 4.2 にネットワーク B（Not-Pretrained）における学習曲線の結果を示す．ベースラインと同様に，λ_{ssl^d} の値を 0.0001 から 1.0 へと増加させるのに伴って，学習初期における L_{ssl^d} の下がり方が急峻になっており，達する最小値自体が小さくなっていることがわかる．また，λ_{ssl^d} が 1.0 の場合に L_{mel} を下げきれなくなる傾向が見られ，ベースラインと同様にこのとき話者類似度の顕著な低下が確認された．加えて，ネットワーク B（Not-Pretrained）では λ_{ssl^d} が 0.1 の場合における L_{ssl^d} の増加が緩やかであり，L_{mel} も十分下げられていることがわかる．さらに，λ_{ssl^d} が 0.1 の場合，L_{mel} の達する最小値自体が，λ_{ssl^d} が 0.01 以下の場合と比較して小さくなっていることがわかる．これはベースラインでは見られなかった新たな傾向であった．最適な λ_{ssl^d} の値は客観評価指標から 0.1 としたが，学習曲線の挙動より，この時他の値の場合と比較して L_{mel} と L_{ssl^d} の両方をバランスよく下げられていたと言える．

ネットワーク C（Not-Pretrained）では，λ_{ssl^d} が 0.1 の場合が最適だと判断した．判断理由はネットワーク B（Not-Pretrained）と同様である．学習曲線の傾向については，ネットワーク B（Not-Pretrained）と同様であったため省略する．

ネットワーク B（Pretrained）では，λ_{ssl^d} 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，事前学習済み重みを読み込まなかったネットワーク B（Not-Pretrained）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．学習曲線の傾向については，ネットワーク B（Not-Pretrained）と同様であったため省略する．学習曲線が同様であるにも関わらず結果の傾向が異なったことについては，重みの初期値が異なっていれば損失の値が同様だとしても異なる局所最適解に収束する可能性があるため，妥当だと判断した．

ネットワーク C（Pretrained）では，λ_{ssl^d} の値が 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．ここでも WER が最低であることを優先して，0.1 が最適だと判断した．学習曲線の傾向はネットワーク B（Not-Pretrained）と同様であったため省略する．

次に，最適なチューニングをした場合における，手法ごとの客観評価指標の全テストデータに渡る平均値を表 4.3 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，Multi-input Vocoder 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値を表す．ベースラインからネットワーク C（Pretrained）については，表 4.2 において太字としたもの，すなわち最適なチューニングだと判断され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 C（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（Not-Pretrained），ネットワークC（Not-Pretrained），ネットワークC（Pretrained）の三つは，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．ここで，HuBERTの事前学習済み重みを初期値とすることの効果について，ネットワークB（Not-Pretrained）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，話者類似度も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，本タスクにおいて有効でないと考えられる．また，ネットワークCの導入効果について，ネットワークB（Not-Pretrained）とネットワークC（Not-Pretrained）を比較すると，ほとんど結果が変わらないことがわかる．一方，ネットワークB（Pretrained）とネットワークC（Pretrained）を比較すると，特に話者類似度についてネットワークC（Pretrained）はネットワークB（Pretrained）よりも0.071高いことがわかる．これより，ネットワークCの効果は，ベースとなるネットワークBの性能に依存して変化すると考えられる．

4.3.2 客観評価 2: 提案手法のさらなる検討

本節では，4.3.1節の客観評価によって得られた知見をもとに，さらなる提案手法の検討を行った結果を述べる．まず，4.3.1節より，これまでに検討した提案手法の中で最も優れているのはネットワークB（Not-Pretrained）だと判断した．選択理由について，まずネットワークC（Not-Pretrained）とネットワークB（Not-Pretrained）はほとんど差がないため，ネットワークCの導入効果はほとんどないと判断しネットワークC（Not-Pretrained）を省いた．次に，ネットワークB（Pretrained）はネットワークB（Not-Pretrained）と比較してWERは1.2%低いがその差は小さく，一方で話者類似度では顕著な低下が見られた．よって，ネットワークB（Pretrained）は本実験下においては効果的でないと判断した．最後に，ネットワークC（Pretrained）について，客観評価指標の値はネットワークB（Not-Pretrained）と同等であり，元となっているネットワークB（Pretrained）を省いたことも考慮して，ネットワークC（Pretrained）も省くこととした．

ネットワークB（Not-Pretrained）に対し，まずは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，入力は事前学習時に揃えてHuBERT中間特徴量としていた．しかし，選択したネットワークB（Not-Pretrained）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，今回はネットワーク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離散特徴量を入力とする場合を比較した．入力の際，メルスペクトログラムは連続したフレームをチャンネル方向に積んで，100Hz・80次元から50Hz・160次元に形状を変換した．HuBERT離散特徴量

は，モデルから出力される予測確率分布をそのまま利用した．HiFi-GAN 入力のように，一度トークンに変換してからベクトル表現を得るという手段も考えられたが，トークンに変換するためには予測確率を元にトークンをサンプリングする必要があり，これは微分不可能な操作であるため学習が不可能となる．そのため，ここでは予測確率分布をそのまま特徴量として利用することにした．パディング部のトークンを含むため，予測確率分布の次元は 101 次元である．これらを結合して $160 + 101 = 261$ 次元の特徴量とした上で，全結合層を通して HuBERT 中間特徴量と同じ 768 次元までチャンネル数を上げ，HuBERT Transformer 層への入力とした．これまでと同様に，五種類の λ_{ssl^d} でグリッドサーチを行った結果を表 4.4 に示す．新たに検討した手法はネットワーク B（Not-Pretrained・Mel-Hub）とした．客観評価指標ごとに最も優れた値を下線で示しており，ネットワーク B（Not-Pretrained・Mel-Hub）については最適だと判断した場合を太字で示している．これより，ネットワーク B（Not-Pretrained・Mel-Hub）では，ネットワーク B（Not-Pretrained）で見られたような WER が下がる重みが確認されないことがわかる．最適だと判断した λ_{ssl^d} が 0.001 の場合においても，話者類似度はネットワーク B（Not-Pretrained）よりも 0.005 とわずかに高いが，WER は 10.1%高い．これより，ネットワーク B（Not-Pretrained・Mel-Hub）は，ネットワーク B（Not-Pretrained）に劣ると判断した．この結果より，HuBERT 中間特徴量はメルスペクトログラムと HuBERT 離散特徴量の複合特徴量と比較して，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方が，ネットワーク B による特徴抽出の対象として優れ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．これまでの提案手法では，先行研究 [4, 5] において報告されたマルチタスク学習の有効性を考慮し，ネットワーク A において HuBERT 中間特徴量だけでなく，メルスペクトログラム，HuBERT 離散特徴量を同時に予測するマルチタスク学習を採用していた．これに対し，ネットワーク A において HuBERT 中間特徴量のみを推定するよう学習する場合を検討することで，マルチタスク学習の有効性を調べた．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（Not-Pretrained・A-SingleTask）とした．客観評価指標ごとに最も優れた値を下線で示しており，ネットワーク B（Not-Pretrained・A-SingleTask）については最適だと判断した場合を太字で示している．これより，ネットワーク B（Not-Pretrained・A-SingleTask）で最適な場合を見ると，ネットワーク B（Not-Pretrained）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，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うよりも，ネットワーク B に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が，これがネットワーク B に入力する HuBERT 中間特徴量を推定する上では，悪影響を与えていると考えられる．

4.3.3 主観評価

主観評価実験は，本実験で検討した提案手法がベースラインに対する改善を達成したか否かを確認するために行った．4.3.1 節および 4.3.2 節の検討結果より，本研究の提案手法の代表としては，ネットワーク B（Not-Pretrained）と，ネットワーク B（Not-Pretrained・SingleTask）を選択した．なぜなら，客観評価指標の結果から，最良だと考えられる提案手法はネットワーク B（Not-Pretrained・SingleTask）であるが，ネットワーク B（Not-Pretrained）もそれに次ぐ性能であり，ネットワーク A におけるマルチタスク学習の有無が主観評価において有意な差であるかも確かめたいと考えたからである．結果として，主観評価実験において比較する音声は以下の五種類である．

1. ベースライン
2. ネットワーク B（Not-Pretrained）
3. ネットワーク B（Not-Pretrained・A-SingleTask）
4. 分析合成
5. 原音声

4.2.5 節で述べたように，主観評価では音声の明瞭性と類似性を五段階評価した．総被験者数は 75 人としたが，アプリの不具合によるエラーを理由に一名のデータを除き，それ以外の被験者についてはダミーサンプルの回答が正しかったためそのまま用いて，74 名分のデータで統計処理を実施した．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 4.3 に示す．被験者の性別は男性 32 名，女性 42 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 18 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 4.6 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 4.7, 4.8 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（N-P, A-S）: ネットワーク B（Not-Pretrained・A-SingleTask）
4. B（N-P）: ネットワーク B（Not-Pretrained）
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．

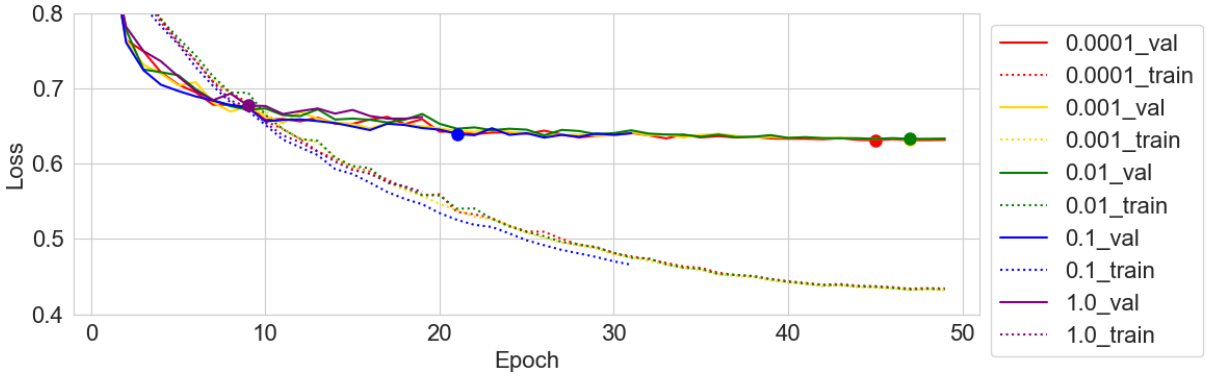
まず，表 4.7 より，提案したネットワーク B（Not-Pretrained）およびネットワーク B（Not-Pretrained・A-SingleTask）は，ベースラインに対して明瞭性の評価値が有意に高いことがわかる．これより，二つの提案手法はどちらもベースラインより話者の想定した発話内容を正確に反映した合成音声を実現できたと考えられる．一方，提案手法二つの間には有意差がないこと

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な差をもたらさなかったと言える．次に，表 4.8 より，提案したネットワーク B（Not-Pretrained）およびネットワーク B（Not-Pretrained・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原音声に似た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明瞭性に有意な差をもたらさなかったが，類似性には有意な差をもたらしたと言える．

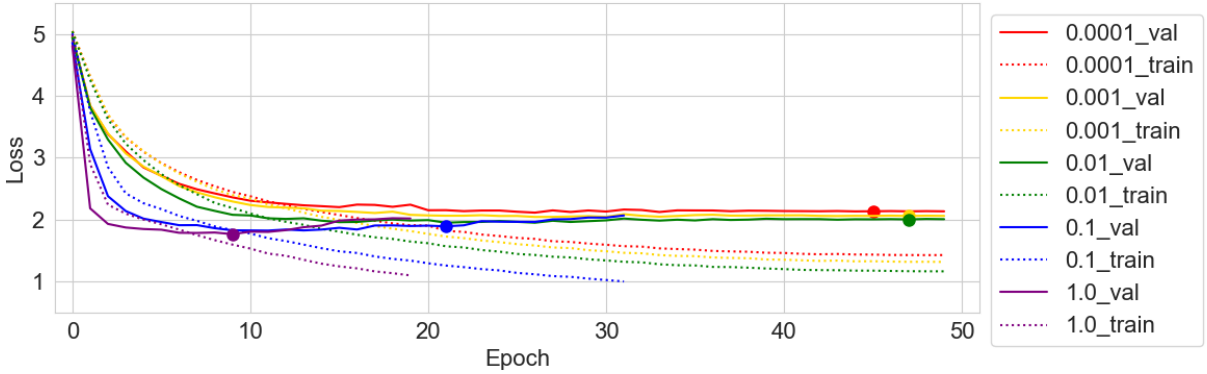
以上のことから，ネットワーク B（Not-Pretrained・A-SingleTask）が明瞭性・類似性の両面において，最も優れた合成音声を実現したと考えられる．一方，ネットワーク B（Not-Pretrained・A-SingleTask）と，合成音声の性能上限を表す分析合成の間には未だ大きな差があり，特に自然音声に迫る合成音の実現は達成されていない．従って，本実験におけるベースラインからの改善は達成したものの，今後もさらなるネットワークの改善が必要だと考えられる．

表 4.2: 損失関数の重み係数 λ_{ssl^d} による客観評価指標の比較

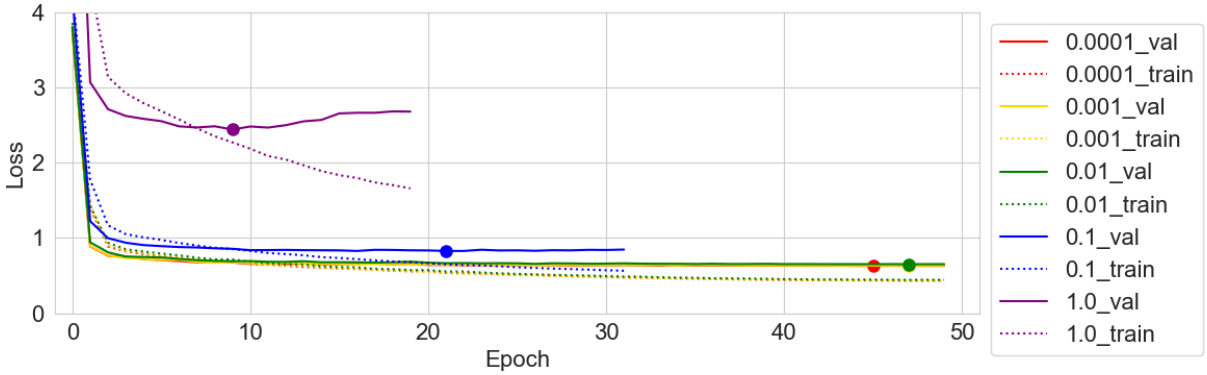
手法	λ_{ssl^d}	WER [%]	話者類似度
ベースライン	0.0001	57.3	0.833
ベースライン	**0.001**	**54.6**	**0.836**
ベースライン	0.01	56.2	0.837
ベースライン	0.1	58.2	0.784
ベースライン	1	59.0	0.685
ネットワーク A	0.0001	55.4	0.841
ネットワーク A	0.001	55.1	0.842
ネットワーク A	0.01	53.4	0.843
ネットワーク A	0.1	54.6	0.809
ネットワーク A	1	58.7	0.698
ネットワーク B（Not-Pretrained）	0.0001	60.6	0.852
ネットワーク B（Not-Pretrained）	0.001	56.7	0.829
ネットワーク B（Not-Pretrained）	0.01	54.4	0.847
ネットワーク B（Not-Pretrained）	**0.1**	**45.3**	**0.840**
ネットワーク B（Not-Pretrained）	1	45.5	0.712
ネットワーク C（Not-Pretrained）	0.0001	58.5	0.860
ネットワーク C（Not-Pretrained）	0.001	55.0	0.850
ネットワーク C（Not-Pretrained）	0.01	56.0	0.848
ネットワーク C（Not-Pretrained）	**0.1**	**45.8**	**0.848**
ネットワーク C（Not-Pretrained）	1	46.9	0.763
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
ネットワーク C（Pretrained）	0.0001	57.8	0.861
ネットワーク C（Pretrained）	0.001	57.2	0.865
ネットワーク C（Pretrained）	0.01	55.7	0.870
ネットワーク C（Pretrained）	**0.1**	**45.5**	**0.849**
ネットワーク C（Pretrained）	1	46.2	0.753



(a) メルスペクトログラムの MAE Loss（式 (4.25) の L_{mel}）

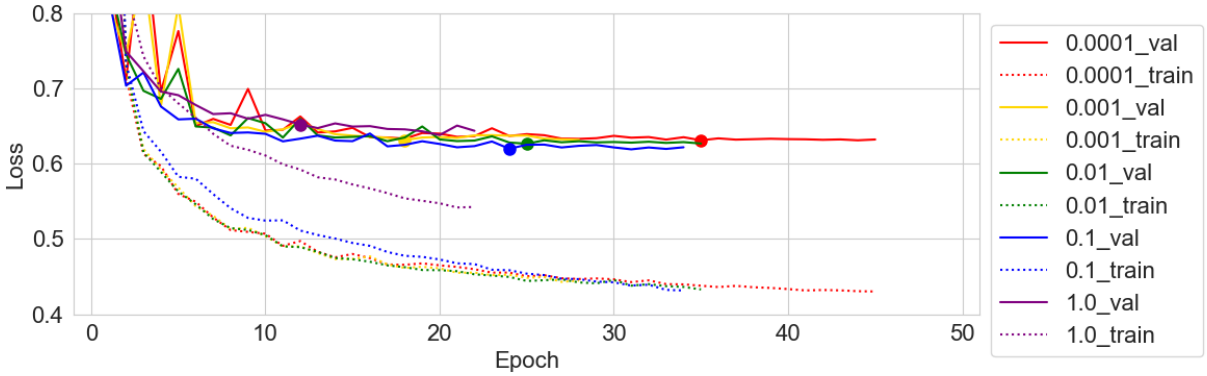


(b) HuBERT 離散特徴量の Cross Entropy Loss（式 (4.25) の L_{ssl^d}）

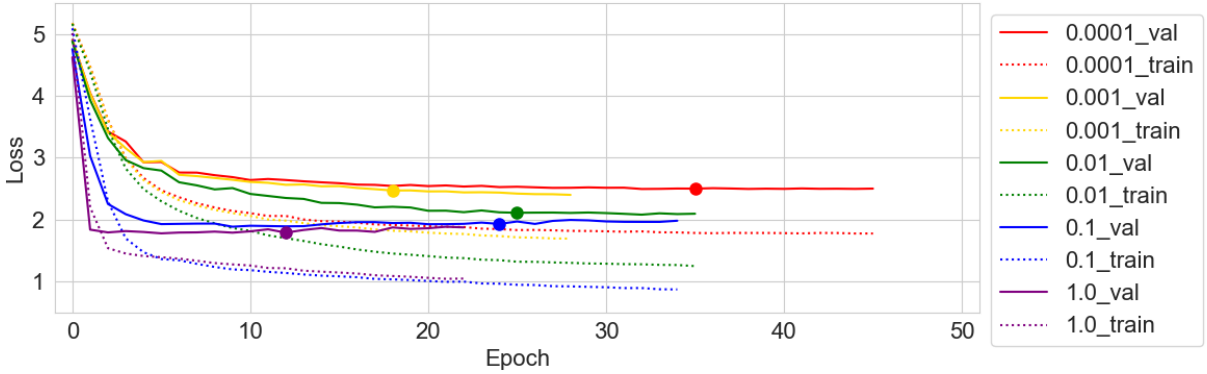


(c) 損失の合計値（式 (4.25) の L）

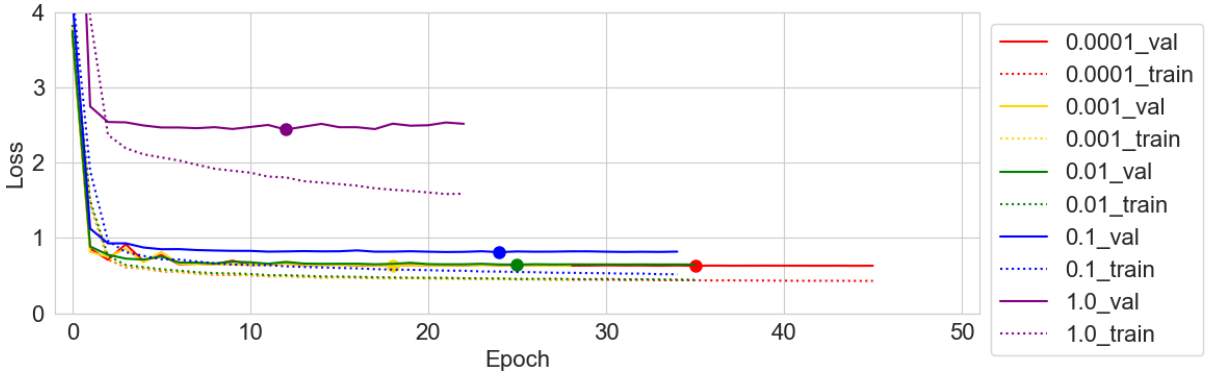
図 4.1: ベースラインにおける学習曲線



(a) メルスペクトログラムの MAE Loss（式 (4.25) の L_{mel}）



(b) HuBERT 離散特徴量の Cross Entropy Loss（式 (4.25) の L_{ssl^d}）



(c) 損失の合計値（式 (4.25) の L）

図 4.2: ネットワーク B（Not-Pretrained）における学習曲線

表 4.3: 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	λ_{ssl^d}	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（Not-Pretrained）	0.1	45.3	0.840
ネットワーク C（Not-Pretrained）	0.1	45.8	0.848
ネットワーク B（Pretrained）	0.1	<u>44.2</u>	0.778
ネットワーク C（Pretrained）	0.1	45.5	<u>0.849</u>
分析合成	-	3.7	0.956
原音声	-	3.7	1.000

表 4.4: 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（Not-Pretrained・Mel-HuB）	0.0001	59.5	0.840
ネットワーク B（Not-Pretrained・Mel-HuB）	**0.001**	**55.4**	**<u>0.845</u>**
ネットワーク B（Not-Pretrained・Mel-HuB）	0.01	56.2	0.803
ネットワーク B（Not-Pretrained・Mel-HuB）	0.1	57.7	0.795
ネットワーク B（Not-Pretrained・Mel-HuB）	1	58.1	0.711
ネットワーク B（Not-Pretrained）	0.1	<u>45.3</u>	0.840

表 4.5: 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（Not-Pretrained・A-SingleTask）	0.0001	54.0	<u>0.867</u>
ネットワーク B（Not-Pretrained・A-SingleTask）	0.001	52.2	0.865
ネットワーク B（Not-Pretrained・A-SingleTask）	0.01	51.8	0.843
ネットワーク B（Not-Pretrained・A-SingleTask）	**0.1**	**<u>42.5</u>**	**0.847**
ネットワーク B（Not-Pretrained・A-SingleTask）	1	43.0	0.768
ネットワーク B（Not-Pretrained）	0.1	45.3	0.840

表 4.6: 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	2.371 ± 0.072	2.933 ± 0.080
ネットワーク B（Not-Pretrained）	0.1	2.753 ± 0.075	3.052 ± 0.085
ネットワーク B（Not-Pretrained・A-SingleTask）	0.1	2.818 ± 0.079	3.182 ± 0.082
分析合成	-	4.749 ± 0.040	4.316 ± 0.071
原音声	-	4.866 ± 0.032	4.705 ± 0.052

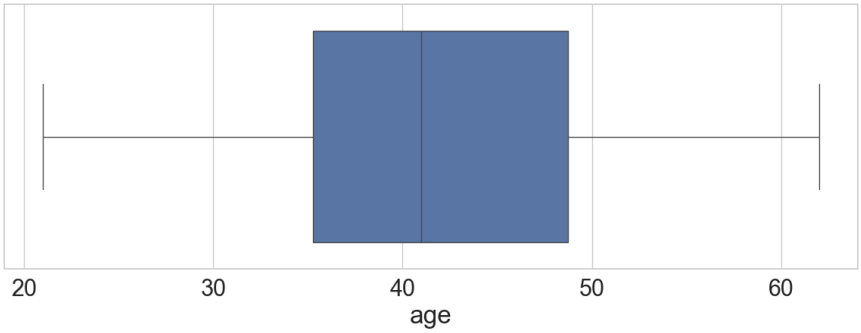


図 4.3: 主観評価実験における被験者の年齢層

表 4.7: 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（N-P, A-S）	B（N-P）	Baseline
GT	5.25×10^{-6}	2.14×10^{-259}	2.82×10^{-285}	0
AbS	-	2.23×10^{-240}	4.56×10^{-266}	0
B（N-P, A-S）	-	-	1.23×10^{-1}	3.61×10^{-16}
B（N-P）	-	-	-	5.77×10^{-13}

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（類似性）

	AbS	B（N-P, A-S）	B（N-P）	Baseline
GT	1.07×10^{-17}	1.93×10^{-156}	1.17×10^{-169}	1.01×10^{-200}
AbS	-	1.31×10^{-82}	5.25×10^{-96}	3.53×10^{-118}
B（N-P, A-S）	-	-	1.70×10^{-2}	1.34×10^{-5}
B（N-P）	-	-	-	2.29×10^{-2}

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.