

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 鎚木 時彦 教授

概要

目次

1 序論	1
1.1 背景	1
1.2 目的	1
1.3 本論文の構成	1
2 音声信号処理	2
2.1 音声のフーリエ変換	2
2.2 メルスペクトログラム	4
3 深層学習	5
3.1 DNN の構成要素	5
3.1.1 全結合層	5
3.1.2 畳み込み層	5
3.1.3 転置畳み込み層	6
3.1.4 活性化関数	7
3.1.5 再帰型ニューラルネットワーク	9
3.1.6 正規化層	11
3.1.7 Transformer	12
3.2 学習方法	15
3.2.1 損失関数	15
3.2.2 勾配降下法	15
3.2.3 正則化	17
3.2.4 最適化手法	18
3.2.5 学習率のスケジューリング	18
3.2.6 誤差逆伝播法	20
3.2.7 学習の安定化	22
3.2.8 自己教師あり学習	23
4 動画音声合成モデルの検討	24
4.1 音声合成法	24
4.1.1 全体像	24
4.1.2 ネットワーク A	25
4.1.3 ネットワーク B	26
4.1.4 ボコーダ	28
4.1.5 損失関数	29
4.2 実験方法	30

4.2.1	利用したデータセット	30
4.2.2	データの前処理	30
4.2.3	本実験で利用した事前学習済みモデルについて	31
4.2.4	本実験で独自に構築したモデルについて	32
4.2.5	学習方法	33
4.2.6	客観評価	34
4.2.7	主観評価	34
4.3	結果	38
4.3.1	客観評価 1: ベースラインと提案手法の比較	38
4.3.2	客観評価 2: 提案手法のさらなる検討	45
4.3.3	主観評価	52
4.4	考察	53
4.4.1	ネットワーク B (Randomized) について	53
4.4.2	ネットワーク B (Pretrained) について	55
5	結論	57
	謝辞	58
	参考文献	59

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のべき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。

$$X[k] = \text{Re}(X[k]) + j \text{Im}(X[k]) \quad (2.1)$$

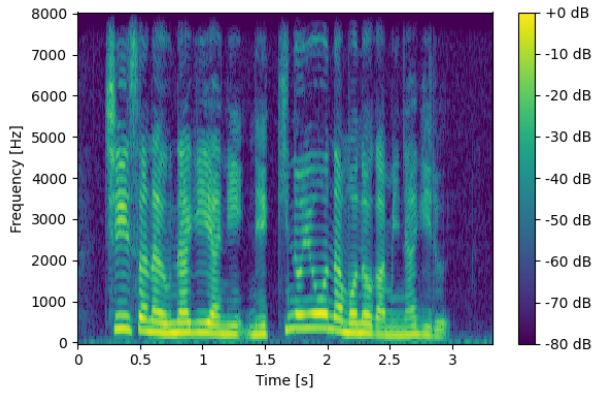
$$= |X[k]| \exp^{j\angle X[k]} \quad (2.2)$$

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、

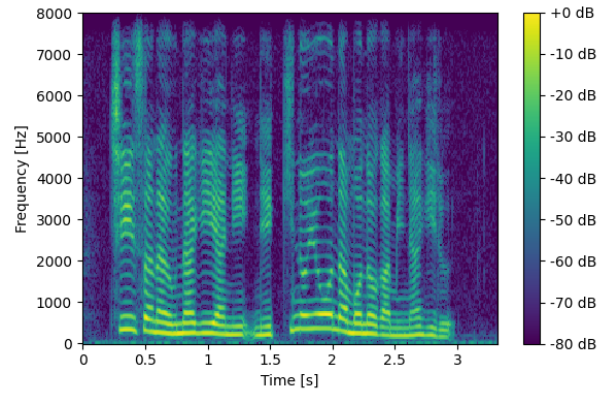
$$|X[k]| = \sqrt{\text{Re}(X[k])^2 + \text{Im}(X[k])^2} \quad (2.3)$$

$$\angle X[k] = \tan^{-1} \frac{\text{Im}(X[k])}{\text{Re}(X[k])} \quad (2.4)$$

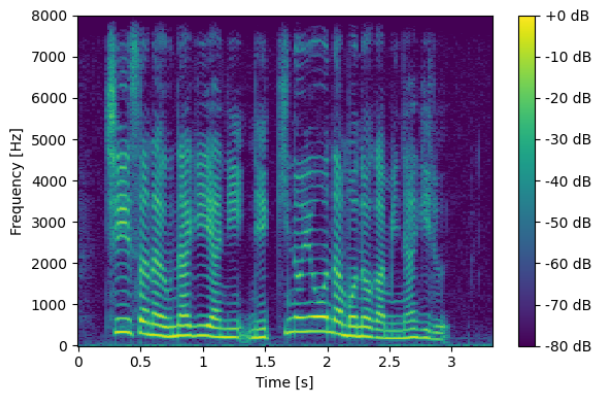
と表される。また、 $|X[k]|^2$ はパワースペクトルと呼ばれる。これにより、信号中にどのような周波数成分がどれくらい含まれているかを調べることができる。しかし、音声はフォルマントやピッチが時々刻々と変化するため、信号全体に対して直接フーリエ変換を適用したとしても有用な結果が得られない。このような音声の持つ非定常性の問題に対して、十分短い時間幅においては信号の定常性が成り立つという仮定のもと、短時間フーリエ変換（short-time Fourier transform; STFT）が用いられる。STFTでは、音声信号に対して窓関数による窓処理を適用し、短時間に区切られた信号それぞれに対して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には時間分解能と周波



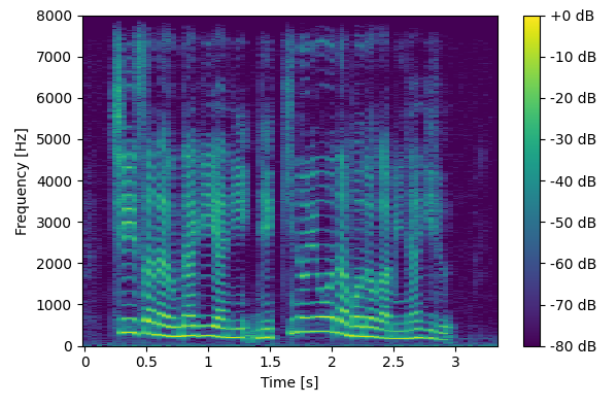
(a) 窓長 12.5ms, シフト幅 5ms



(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

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 t 、周波数インデックスを k として $X[t, k]$ と表すと、 $X[t, k]$ 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、 $|X[t, k]|$ を振幅スペクトログラム、 $\angle X[t, k]$ を位相スペクトログラム、 $|X[t, k]|^2$ 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．

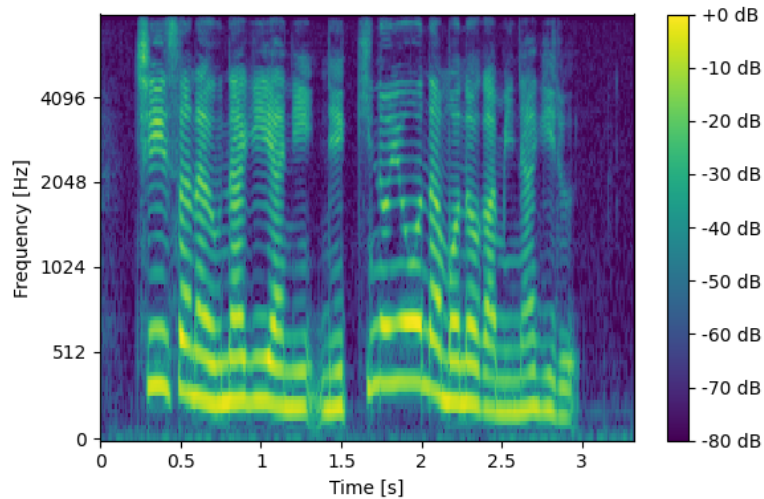


図 2.2: 「小さな鰻屋に、熱気のようなものがみなぎる」と発話した音声に対する対数メルスペクトログラム

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。

$$\text{Mel}(f) = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (2.5)$$

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 $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$ とすると、出力 $\mathbf{y} \in \mathbb{R}^{D_{\text{out}}}$ は、

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (3.1)$$

で与えられる。ここで、 $\mathbf{W} \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$ は重み、 $\mathbf{b} \in \mathbb{R}^{D_{\text{out}}}$ 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 $\mathbf{X} \in \mathbb{R}^{D_{\text{in}} \times T_{\text{in}}}$ 、出力を $\mathbf{Y} \in \mathbb{R}^{D_{\text{out}} \times T_{\text{out}}}$ 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、

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

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。

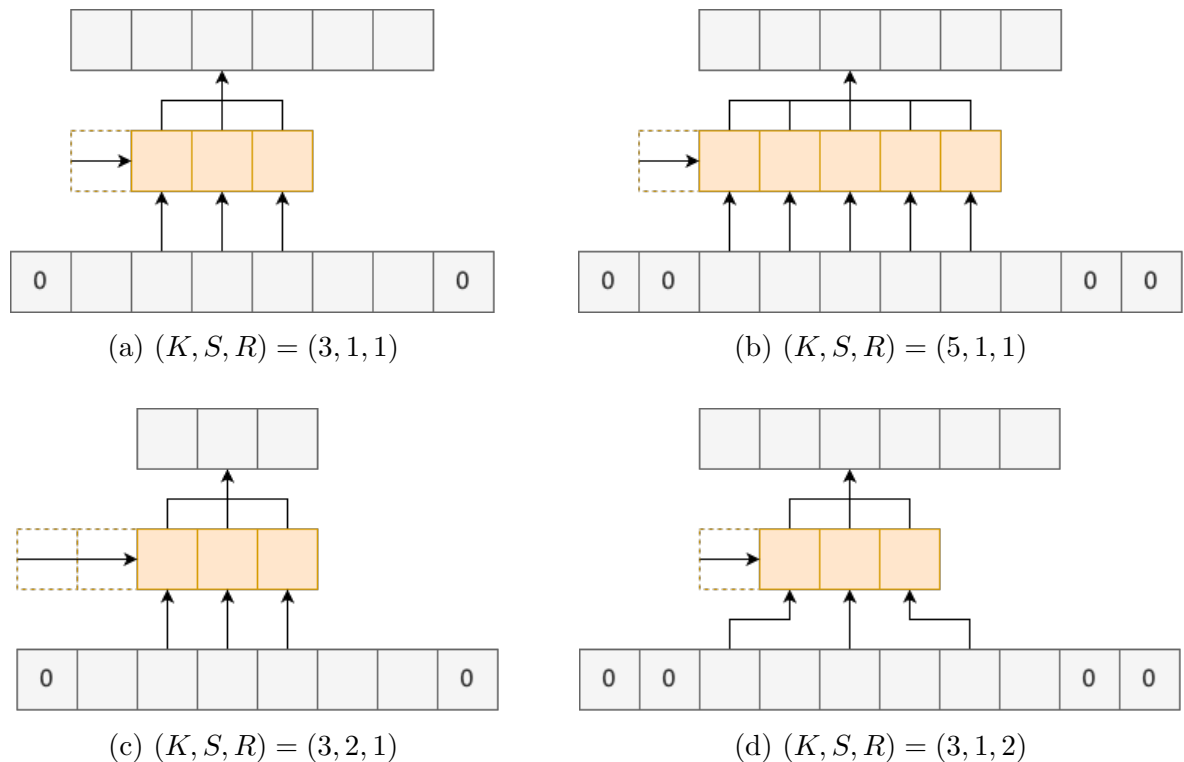


図 3.1: ある次元における一次元畳み込み層の処理. K はカーネルサイズ, S はストライド, R はダイレーションを表し, 図中の 0 はパディング部を表す.

畳み込み層における主要なパラメータは3つある. 1つ目はカーネルサイズであり, これによって考慮できる入力特徴量の範囲が定まる. 2つ目はストライドであり, これによってカーネルのシフト幅を設定できる. 3つ目はダイレーションであり, これは畳み込み演算において計算対象となる入力特徴量の間隔を表す. ダイレーションを大きくすることで, カーネルサイズが同じでも考慮できる入力特徴量の範囲を広げることが可能である. また, 出力系列長 T_{out} を入力系列長 T_{in} の整数倍に保つには, 上記のパラメータに対して適切なパディング長を指定する必要がある. 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

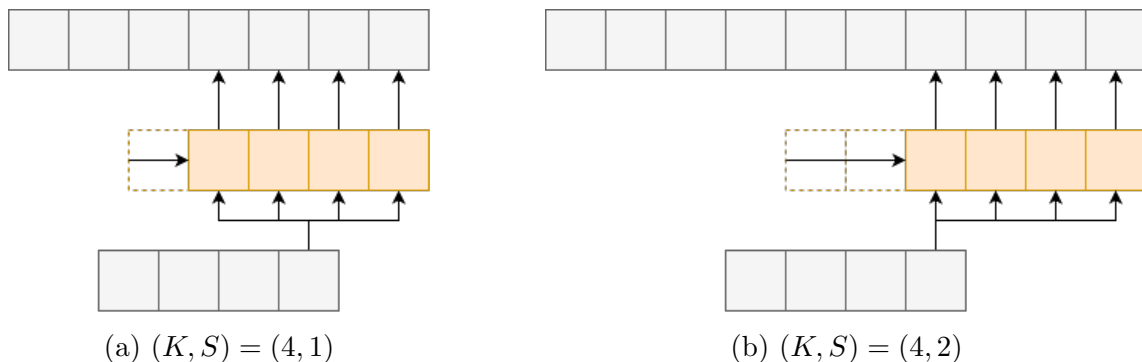


図 3.2: ある次元における一次元転置畳み込み層の処理. K はカーネルサイズ, S はストライドを表す.

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 6 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.

1 つ目は, シグモイド関数である. シグモイド関数は

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

で与えられ, その一階導関数は

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

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.

2 つ目は, tanh 関数である. tanh は

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

で与えられ、その一階導関数は

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

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

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階導関数の値が小さく, 勾配消失のリスクを抱えていることがわかる.

3 つ目は, ReLU である. ReLU は

$$\text{ReLU}(x) = \max(0, x) \quad (3.8)$$

で与えられ、その一階導関数は

$$\frac{d\text{ReLU}(x)}{dx} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (3.9)$$

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 $d\text{ReLU}(0)/dx = 0$ とした. ReLU は入力 x 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 x が 0 未満の値を取るとき, ReLU 入力についての出力の勾配は 0 になるから, ReLU 以前の層の重みが更新されず, 学習が遅くなる可能性がある.

3 つ目は, LeakyReLU[6] である. LeakyReLU は

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x \leq 0 \end{cases} \quad (3.10)$$

で与えられ、その一階導関数は

$$\frac{d\text{LeakyReLU}(x)}{dx} = \begin{cases} 1 & \text{if } x > 0 \\ a & \text{if } x \leq 0 \end{cases} \quad (3.11)$$

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 $d\text{LeakyReLU}(0)/dx = a$ 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.

5 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.

6つ目は、GELU[8]である。GELUは

$$\text{GELU}(x) = x\Phi(x) \quad (3.12)$$

で与えられる。ここで、

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

である。GELUの一階導関数は、

$$\frac{d\text{GELU}(x)}{dx} = \Phi(x) + \frac{x}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \quad (3.14)$$

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、 $m \sim \text{Bernoulli}(\Phi(x))$ とすると、

$$\text{GELU}(x) = x\Phi(x) \quad (3.15)$$

$$= 1x \cdot \Phi(x) + 0x \cdot (1 - \Phi(x)) \quad (3.16)$$

$$= \mathbb{E}[mx] \quad (3.17)$$

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。

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f) \quad (3.18)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_i) \quad (3.19)$$

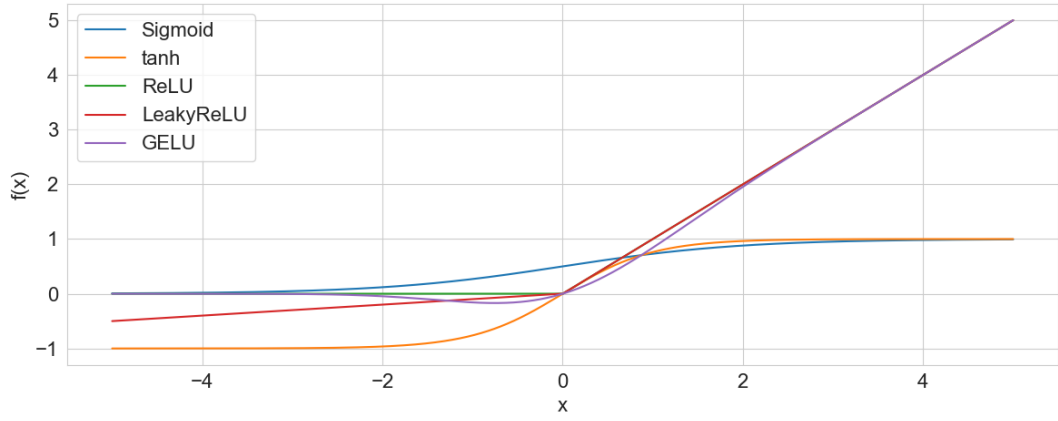
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_h[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_h) \quad (3.20)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (3.21)$$

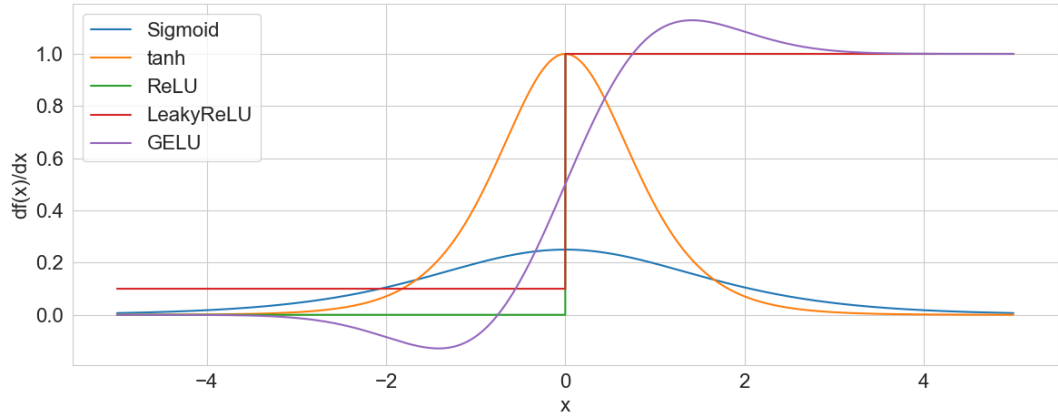
$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_o) \quad (3.22)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (3.23)$$

ここで、 $\mathbf{x}_t \in \mathbb{R}^{D_{\text{in}}}$ は時刻 t の入力、 $\mathbf{f}_t \in [0, 1]^{D_{\text{out}}}$ は忘却ゲートの出力、 $\mathbf{i}_t \in [0, 1]^{D_{\text{out}}}$ は入力ゲートの出力、 $\mathbf{c}_t \in [-1, 1]^{D_{\text{out}}}$ が時刻 t におけるセルの状態、 $\mathbf{o}_t \in [0, 1]^{D_{\text{out}}}$ 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, $\mathbf{h}_t \in [-1, 1]^{D_{\text{out}}}$ が時刻 t における隠れ状態を表す. $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_h, \mathbf{W}_o \in \mathbb{R}^{D_{\text{out}} \times (D_{\text{in}} + D_{\text{out}})}$ は重み, $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_h, \mathbf{b}_o \in \mathbb{R}^{D_{\text{out}}}$ はバイアスである. 忘却ゲート出力 \mathbf{f}_t が前時刻のセル状態 \mathbf{c}_t に含まれる情報の選択, 入力ゲート出力 \mathbf{i}_t が新たな入力 $\tilde{\mathbf{c}}_t$ に含まれる情報の選択に用いられ, \mathbf{c}_t が決まる. その後, 出力ゲート出力 \mathbf{o}_t が \mathbf{c}_t に含まれる情報の選択に用いられ, \mathbf{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.

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

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

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

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

ここで、 $\mathbf{x}_t \in \mathbb{R}^{D_{\text{in}}}$ が時刻 t における入力、 $\mathbf{z}_t \in [0, 1]^{D_{\text{out}}}$ が更新ゲートの出力、 $\mathbf{r}_t \in [0, 1]^{D_{\text{out}}}$ がリセットゲートの出力、 $\mathbf{h}_t \in [-1, 1]^{D_{\text{out}}}$ が時刻 t における隠れ状態を表す。また、 $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{D_{\text{out}} \times (D_{\text{in}} + D_{\text{out}})}$ は重み、 $\mathbf{b}_z, \mathbf{b}_r, \mathbf{b}_h \in \mathbb{R}^{D_{\text{out}}}$ はバイアスである。更新ゲート出力 \mathbf{z}_t が \mathbf{h}_{t-1} と $\tilde{\mathbf{h}}_t$ に含まれる情報の選択、リセットゲート出力 \mathbf{r}_t が \mathbf{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わるため、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 $\mathbf{x}_n \in \mathbb{R}^D$ ($n = 1, \dots, N$)、出力特徴量を $\mathbf{y}_n \in \mathbb{R}^D$ ($n = 1, \dots, N$) とする。このとき、各 n に対し入力特徴量 \mathbf{x}_n の d 次元目の成分を $x_{n,d}$ 、出力特徴量 \mathbf{y}_n の d 次元目の成分を $y_{n,d}$ とすると、 $y_{n,d}$ は

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

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

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

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

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

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

$$\beta_d = \mu_d^B \quad (3.33)$$

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ツ

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

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

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

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

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

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 \mathbf{w} を、

$$\mathbf{w} = \frac{\mathbf{v}}{\|\mathbf{v}\|} g \quad (3.38)$$

のように単位ベクトル $\mathbf{v}/\|\mathbf{v}\|$ (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 \mathbf{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 $\mathbf{X} \in \mathbb{R}^{T \times D_{\text{model}}}$ 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 $\mathbf{Q} \in \mathbb{R}^{T \times D_k}$ 、キー $\mathbf{K} \in \mathbb{R}^{T \times D_k}$ 、バリュー $\mathbf{V} \in \mathbb{R}^{T \times D_v}$ の計算を行う。これは、

$$\mathbf{Q} = \mathbf{XW}_Q + ((\mathbf{b}_Q)_T)^\top \quad (3.39)$$

$$\mathbf{K} = \mathbf{XW}_K + ((\mathbf{b}_K)_T)^\top \quad (3.40)$$

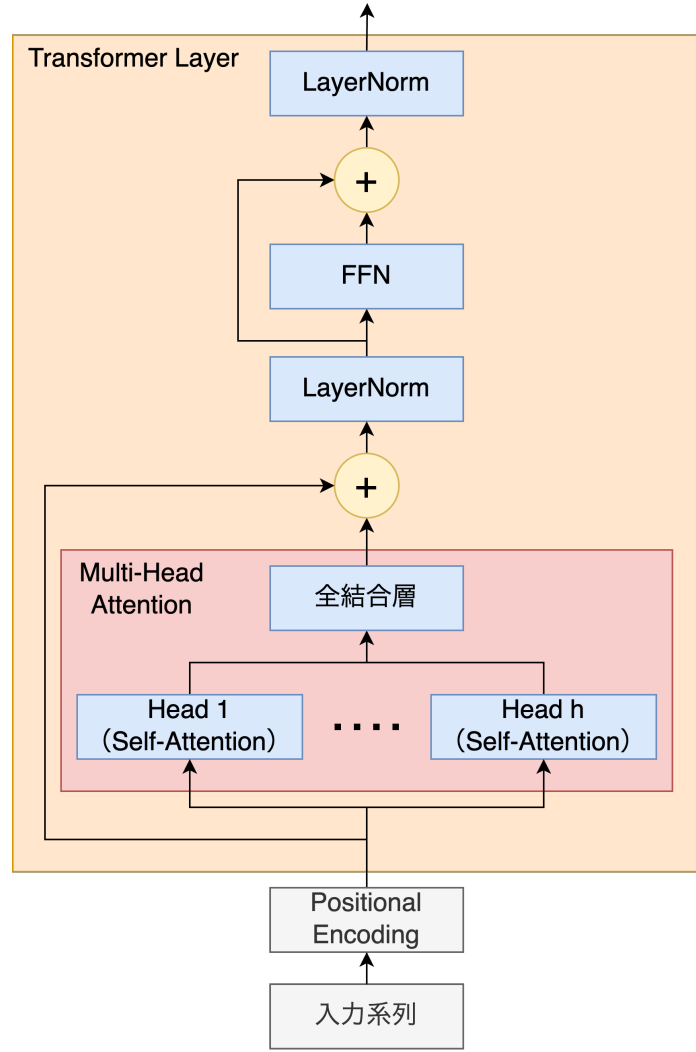


図 3.4: Transformer 層の構造

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

で与えられる．ここで， $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{D_{\text{model}} \times D_k}$, $\mathbf{W}_V \in \mathbb{R}^{D_{\text{model}} \times D_v}$ は重み， $\mathbf{b}_Q, \mathbf{b}_K \in \mathbb{R}^{D_k}$, $\mathbf{b}_V \in \mathbb{R}^{D_v}$ はバイアスである．また， $(\mathbf{b}_Q)_T \in \mathbb{R}^{D_k \times T}$ は，列ベクトル \mathbf{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，

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

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

ここまですelf-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 $\mathbf{Q}^h \in \mathbb{R}^{T \times \frac{D_k}{H}}$ ，キー $\mathbf{K}^h \in \mathbb{R}^{T \times \frac{D_k}{H}}$ ，

バリュー $\mathbf{V}^h \in \mathbb{R}^{T \times \frac{D_v}{H}}$ の計算は,

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

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

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

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

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

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 Attention の出力が得られる. すなわち,

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{Attention}(\mathbf{Q}^h, \mathbf{K}^h, \mathbf{V}^h), \dots, \text{Attention}(\mathbf{Q}^h, \mathbf{K}^h, \mathbf{V}^h)] \mathbf{W}_o + ((\mathbf{b}_o)_T)^\top \quad (3.47)$$

と表される. ここで, $\mathbf{W}_o \in \mathbb{R}^{D_v \times D_{\text{model}}}$ は重み, $\mathbf{b}_o \in \mathbb{R}^{D_{\text{model}}}$ 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 $\mathbf{Y} \in \mathbb{R}^{T \times D_{\text{model}}}$ は,

$$\mathbf{Y} = \text{LayerNorm}(\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) + \mathbf{X}) \quad (3.48)$$

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,

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

となる. ここで, $\mathbf{W}_1 \in \mathbb{R}^{D_{\text{model}} \times D_{\text{fc}}}$, $\mathbf{W}_2 \in \mathbb{R}^{D_{\text{fc}} \times D_{\text{model}}}$ は重み, $\mathbf{b}_1 \in \mathbb{R}^{D_{\text{fc}}}$, $\mathbf{b}_2 \in \mathbb{R}^{D_{\text{model}}}$ 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,

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

で与えられる.

3.2 学習方法

本節において、 $\theta \in \mathbb{R}^{N_{\text{model}}}$ 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 1 つに、MAE (Mean Absolute Error) Loss がある。推定対象を $\mathbf{y} \in \mathbb{R}^D$ 、DNN による推定結果を $\hat{\mathbf{y}} \in \mathbb{R}^D$ とすると、MAE Loss は

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

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

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

で与えられる。

一方、分類問題において用いられる関数の 1 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 $\mathbf{y} \in [0, 1]^C$ 、DNN による推定結果を $\hat{\mathbf{y}} \in \mathbb{R}^C$ とすると、Cross Entropy Loss は

$$L_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{c=1}^C y_c \log \frac{\exp(\hat{y}_c)}{\sum_{i=1}^C \exp(\hat{y}_i)} \quad (3.53)$$

で与えられる。ここで、 \mathbf{y} はクラスに対する確率分布であり、

$$\sum_{c=1}^C y_c = 1 \quad (3.54)$$

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 $\mathbf{Y} \in [0, 1]^C$ の場合、DNN による推定結果を $\hat{\mathbf{Y}} \in \mathbb{R}^{C \times T}$ とすると、

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

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

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

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 $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in \mathcal{I}} \subset \mathcal{D}_{\text{train}}$ 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

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

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,

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

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, 3 種類の方法がある [15]. 1 つ目は, バッチ勾配降下法である. これは,

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

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

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

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. 3 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,

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

で与えられる. これより, 各イテレーションで2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3つ述べる。

1つ目は、L2正則化である。これは、 $\mathcal{L}(\boldsymbol{\theta}; \mathcal{I})$ を

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

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。

2つ目は、Weight Decayである。これは、重みの更新式を

$$\boldsymbol{\theta}_{\tau \text{ iter}} = \boldsymbol{\theta}_{\tau \text{ iter} - 1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau \text{ iter} - 1}; \mathcal{I}_{\tau \text{ iter}}) - \lambda' \boldsymbol{\theta} \quad (3.63)$$

とすることで与えられる。ここで、 $\lambda' \in [0, \infty)$ は正則化の度を調整するパラメータである。これより、Weight Decayは重み $\boldsymbol{\theta}$ の絶対値が過大になることを防ぐ手法だと言える。

3つ目は、Dropout[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 $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$ 、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 $\mathbf{y}_{\text{train}} \in \mathbb{R}^{D_{\text{out}}}$ および推論時のDropout出力 $\mathbf{y}_{\text{infer}} \in \mathbb{R}^{D_{\text{out}}}$ は、

$$\mathbf{y}_{\text{train}} = \frac{\mathbf{x} \odot \mathbf{m}}{1 - p} \quad (3.64)$$

$$\mathbf{y}_{\text{infer}} = \mathbf{x} \quad (3.65)$$

で与えられる。ここで、 $\mathbf{m} \in \{0, 1\}^{D_{\text{in}}}$ 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 \mathbf{m} の従う確率分布上で $\mathbf{y}_{\text{train}}$ の期待値をとれば、

$$\mathbb{E}[\mathbf{y}_{\text{train}}] = \mathbb{E}\left[\frac{\mathbf{x} \odot \mathbf{m}}{1 - p}\right] \quad (3.66)$$

$$= \frac{\mathbf{x}}{1 - p} \odot \mathbb{E}[\mathbf{m}] \quad (3.67)$$

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

$$= \mathbf{x} = \mathbf{y}_{\text{infer}} \quad (3.69)$$

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。

3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、 $\mathbf{g}_{\tau^{\text{iter}}} \in \mathbb{R}^{N_{\text{model}}}$ は勾配の一次モーメント、 $\mathbf{m}_{\tau^{\text{iter}}} \in \mathbb{R}^{N_{\text{model}}}$ が勾配の一次モーメントの指数移動平均、 $\mathbf{v}_{\tau^{\text{iter}}} \in \mathbb{R}^{N_{\text{model}}}$ が勾配の二次モーメントの指数移動平均である。 $\hat{\mathbf{m}}_{\tau^{\text{iter}}}$ および $\hat{\mathbf{v}}_{\tau^{\text{iter}}}$ はそれぞれ、 $\mathbf{m}_{\tau^{\text{iter}}}$ および $\mathbf{v}_{\tau^{\text{iter}}}$ 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

```

1: Input:  $\eta, \beta_1, \beta_2, \lambda, \boldsymbol{\theta}_0, L(\boldsymbol{\theta})$ 
2:  $\mathbf{m}_0 = 0, \mathbf{v}_0 = 0$ 
3: for  $\tau^{\text{iter}} = 1$  to  $\dots$  do
4:    $\mathbf{g}_{\tau^{\text{iter}}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}) + \lambda \boldsymbol{\theta}_{\tau^{\text{iter}}-1}$ 
5:    $\mathbf{m}_{\tau^{\text{iter}}} = \beta_1 \mathbf{m}_{\tau^{\text{iter}}-1} + (1 - \beta_1) \mathbf{g}_{\tau^{\text{iter}}}$ 
6:    $\mathbf{v}_{\tau^{\text{iter}}} = \beta_2 \mathbf{v}_{\tau^{\text{iter}}-1} + (1 - \beta_2) \mathbf{g}_{\tau^{\text{iter}}} \odot \mathbf{g}_{\tau^{\text{iter}}}$ 
7:    $\tilde{\mathbf{m}}_{\tau^{\text{iter}}} = \mathbf{m}_{\tau^{\text{iter}}} / (1 - \beta_1^{\tau^{\text{iter}}})$ 
8:    $\tilde{\mathbf{v}}_{\tau^{\text{iter}}} = \mathbf{v}_{\tau^{\text{iter}}} / (1 - \beta_2^{\tau^{\text{iter}}})$ 
9:    $\boldsymbol{\theta}_{\tau^{\text{iter}}} = \boldsymbol{\theta}_{\tau^{\text{iter}}-1} - \eta \tilde{\mathbf{m}}_{\tau^{\text{iter}}} / (\sqrt{\tilde{\mathbf{v}}_{\tau^{\text{iter}}} + \epsilon})$ 
10: end for
11: 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、3つのスケジューラを例として述べる。また、各スケジューラを用いた場合における学習率の遷移を図3.5に示す。

Algorithm 2 AdamW

```
1: Input:  $\eta, \beta_1, \beta_2, \lambda, \theta_0, L(\theta)$ 
2:  $\mathbf{m}_0 = 0, \mathbf{v}_0 = 0$ 
3: for  $\tau^{\text{iter}} = 1$  to  $\dots$  do
4:    $\mathbf{g}_{\tau^{\text{iter}}} = \nabla_{\theta} \mathcal{L}(\theta_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}})$ 
5:    $\mathbf{m}_{\tau^{\text{iter}}} = \beta_1 \mathbf{m}_{\tau^{\text{iter}}-1} + (1 - \beta_1) \mathbf{g}_{\tau^{\text{iter}}}$ 
6:    $\mathbf{v}_{\tau^{\text{iter}}} = \beta_2 \mathbf{v}_{\tau^{\text{iter}}-1} + (1 - \beta_2) \mathbf{g}_{\tau^{\text{iter}}} \odot \mathbf{g}_{\tau^{\text{iter}}}$ 
7:    $\tilde{\mathbf{m}}_{\tau^{\text{iter}}} = \mathbf{m}_{\tau^{\text{iter}}} / (1 - \beta_1^{\tau^{\text{iter}}})$ 
8:    $\tilde{\mathbf{v}}_{\tau^{\text{iter}}} = \mathbf{v}_{\tau^{\text{iter}}} / (1 - \beta_2^{\tau^{\text{iter}}})$ 
9:    $\theta_{\tau^{\text{iter}}} = \theta_{\tau^{\text{iter}}-1} - \eta \tilde{\mathbf{m}}_{\tau^{\text{iter}}} / (\sqrt{\tilde{\mathbf{v}}_{\tau^{\text{iter}}} + \epsilon}) - \lambda \theta$ 
10: end for
11: Return  $\theta_{\tau^{\text{iter}}}$ 
```

1つ目は、StepLRSchedulerである。これは、初期学習率を η_0 として、エポック τ^{epoch} における学習率 $\eta_{\tau^{\text{epoch}}}$ を

$$\eta_{\tau^{\text{epoch}}} = \eta_0 \gamma^{\lfloor \tau^{\text{epoch}} / \text{stepSize} \rfloor} \quad (3.70)$$

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。

2つ目は、ExponentialLRSchedulerである。これは、 $\eta_{\tau^{\text{epoch}}}$ を

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

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。

3つ目は、Cosine Annealing with Warmupである。これは、 $\eta_{\tau^{\text{epoch}}}$ を

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

で与えるスケジューラである。ここで、 η_{\min} は最小学習率、 η_{\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 η_{\min} から η_{\max} まで線形に増加させるのにかか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。

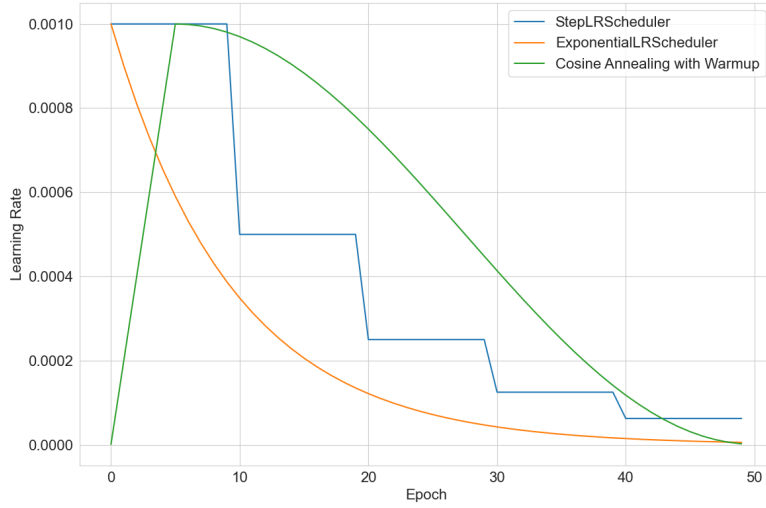


図 3.5: スケジューラによる学習率の変化

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、

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

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、

$$o_q^{n-1} = \phi(a_q^{n-1}) \quad (3.74)$$

である．例外として、各 $i \in \mathcal{I}$ に対し、 $\mathbf{o}^0 = \mathbf{x}_i$ 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、

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

と整理できる．この時、

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

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

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

となる. ここで, $L(\mathbf{y}_i, f(\mathbf{x}_i; \boldsymbol{\theta})) = L_i$ とおいた. この時, 各 $n \in \{1, \dots, N_{\text{layer}}\}$ に対し, $\partial L_i / \partial w_{p,q}^n$ は式 (3.75) を用いて,

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

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

$$= \delta_q^n o_p^{n-1} \quad (3.81)$$

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,

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

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

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

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

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

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

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,

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

$$= o_p^{n-1} \frac{\partial L_i}{\partial a_q^n} \quad (3.89)$$

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

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

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

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

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

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、

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

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 $\mathbf{g}_{\text{accum}}$ とすると、この更新は

$$\mathbf{g}_{\text{accum}} \leftarrow \mathbf{g}_{\text{accum}} + \nabla_{\theta} \mathcal{L}(\theta_{\tau^{\text{iter}}-1}; \mathcal{I}_{\tau^{\text{iter}}}) \quad (3.96)$$

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 θ の更新は

$$\theta_{\tau^{\text{iter}}} = \theta_{\tau^{\text{iter}}-1} - \frac{\eta}{N_{\text{accum}}} \mathbf{g}_{\text{accum}} \quad (3.97)$$

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 $\mathbf{X} \in \mathbb{R}^{T \times D}$ 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 $\mathbf{X}^{\text{masked}}$ の各時刻 t における値 $\mathbf{x}_t^{\text{masked}}$ は、

$$\mathbf{x}_t^{\text{masked}} = \begin{cases} \mathbf{x}_t & \text{if } t \notin \mathcal{M} \\ \mathbf{m} & \text{if } t \in \mathcal{M} \end{cases} \quad (3.98)$$

である。 $\mathbf{m} \in \mathbb{R}^D$ 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 $\mathbf{Y} \in \{0, 1\}^{T \times C}$ とする。各時刻 t における教師ラベル \mathbf{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 $\hat{\mathbf{Y}} \in [0, 1]^{T \times C}$ に対する損失は、

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

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

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

とする。各 n に対し、 $\mathbf{X}_n^{\text{video}} \in \mathbb{R}^{D^{\text{video}} \times T_n^{\text{video}} \times W^{\text{video}} \times H^{\text{video}}}$ は口唇動画、 $\mathbf{y}_n^{\text{sp-wf}} \in \mathbb{R}^{T_n^{\text{sp-wf}}}$ は原音声の音声波形、 $\mathbf{x}_{s_n}^{\text{spk-emb}} \in \mathbb{R}^{D^{\text{spk-emb}}}$ は話者ベクトル、 $\mathbf{Y}_n^{\text{mel}} \in \mathbb{R}^{D^{\text{mel}} \times T_n^{\text{mel}}}$ はメルスペクトログラム、 $\mathbf{Y}_n^{\text{HuB-int}} \in \mathbb{R}^{D^{\text{HuB-int}} \times T_n^{\text{HuB}}}$ は HuBERT 中間特徴量、 $\mathbf{Y}_n^{\text{HuB-disc}} \in \{0, 1\}^{C \times T_n^{\text{HuB}}}$ は HuBERT 離散特徴量とする。ここで、話者ベクトル $\mathbf{x}_{s_n}^{\text{spk-emb}}$ 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、

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

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、

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

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、

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

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

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 3 つからなる。まず、ネットワーク A を f_A とすると、 f_A の行う処理は、

$$\hat{\mathbf{Y}}_n^{\text{HuB-int}}, \hat{\mathbf{Y}}_n^{\text{mel-A}}, \hat{\mathbf{Y}}_n^{\text{HuB-disc-A}} = f_A(\mathbf{X}_n^{\text{video}}, \mathbf{x}_{s_n}^{\text{spk-emb}}; \boldsymbol{\theta}_A) \quad (4.6)$$

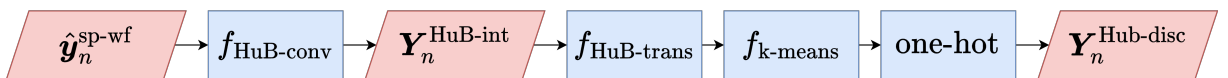


図 4.1: HuBERT を利用した特徴量計算の流れ

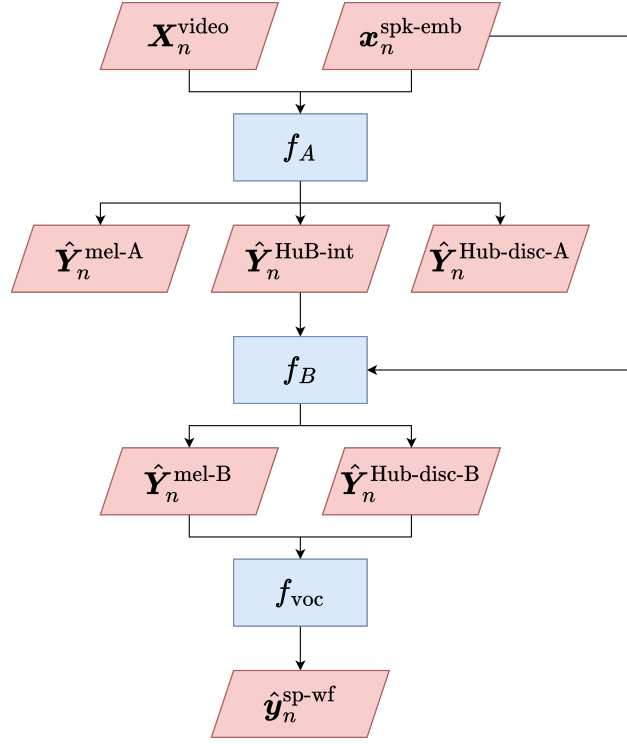


図 4.2: 提案手法の全体像

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

$$\hat{\mathbf{Y}}_n^{\text{mel-B}}, \hat{\mathbf{Y}}_n^{\text{HuB-disc-B}} = f_B \left(\hat{\mathbf{Y}}_n^{\text{HuB-int}}, \mathbf{x}_n^{\text{spk-emb}}; \boldsymbol{\theta}_B \right) \quad (4.7)$$

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

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

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 $\mathbf{X}_n^{\text{video}}$ を AVHuBERT に通すことで, 特徴量 $\mathbf{H}_n^A \in \mathbb{R}^{D^A \times T_n^{\text{video}}}$ に変換する. これは,

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

と表される． $\theta_{\text{A-AVHuB}}$ は，AVHuBERT の事前学習済み重みで初期化した．次に， \mathbf{H}_n^{A} の各時刻 t におけるベクトルに対し，話者ベクトル $\mathbf{x}_{s_n}^{\text{spk-emb}}$ 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，

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

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，

$$\mathbf{H}_n^{\text{A}} = f_{\text{post}} (\mathbf{H}_n^{\text{A}}; \theta_{\text{A-post}}) \quad (4.11)$$

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 $\mathbf{x}_{s_n}^{\text{spk-emb}}$ が次元方向に結合された特徴量 \mathbf{H}_n^{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． f_{AVHuB} は Masked Prediction によって事前学習されたモデルだから，動画の文脈的な構造を考慮するのに適していると考えられる一方で，音声合成に必要な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， \mathbf{H}_n^{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，

$$\hat{\mathbf{Y}}_n^{\text{HuB-int}} = \text{FC} (\mathbf{H}_n^{\text{A}}; \theta_{\text{A-fc-HuB-int}}) \quad (4.12)$$

$$\hat{\mathbf{Y}}_n^{\text{mel-A}} = \text{FC} (\mathbf{H}_n^{\text{A}}; \theta_{\text{A-fc-mel}}) \quad (4.13)$$

$$\hat{\mathbf{Y}}_n^{\text{HuB-disc-A}} = \text{FC} (\mathbf{H}_n^{\text{A}}; \theta_{\text{A-fc-HuB-Disc}}) \quad (4.14)$$

と表される． $\theta_{\text{A-fc-spk}}$, $\theta_{\text{A-post}}$, $\theta_{\text{A-fc-HuB-int}}$, $\theta_{\text{A-fc-mel}}$, $\theta_{\text{A-fc-HuB-Disc}}$ 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 $\hat{\mathbf{Y}}_n^{\text{HuB-int}}$ を $f_{\text{HuB-trans}}$ に通すことで，特徴量 $\mathbf{H}_n^{\text{B}} \in \mathbb{R}^{D^{\text{B}} \times T_n^{\text{HuB}}}$ に変換する．これは，

$$\mathbf{H}_n^{\text{B}} = f_{\text{HuB-trans}} \left(\hat{\mathbf{Y}}_n^{\text{HuB-int}}; \theta_{\text{B-HuB-trans}} \right) \quad (4.15)$$

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

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

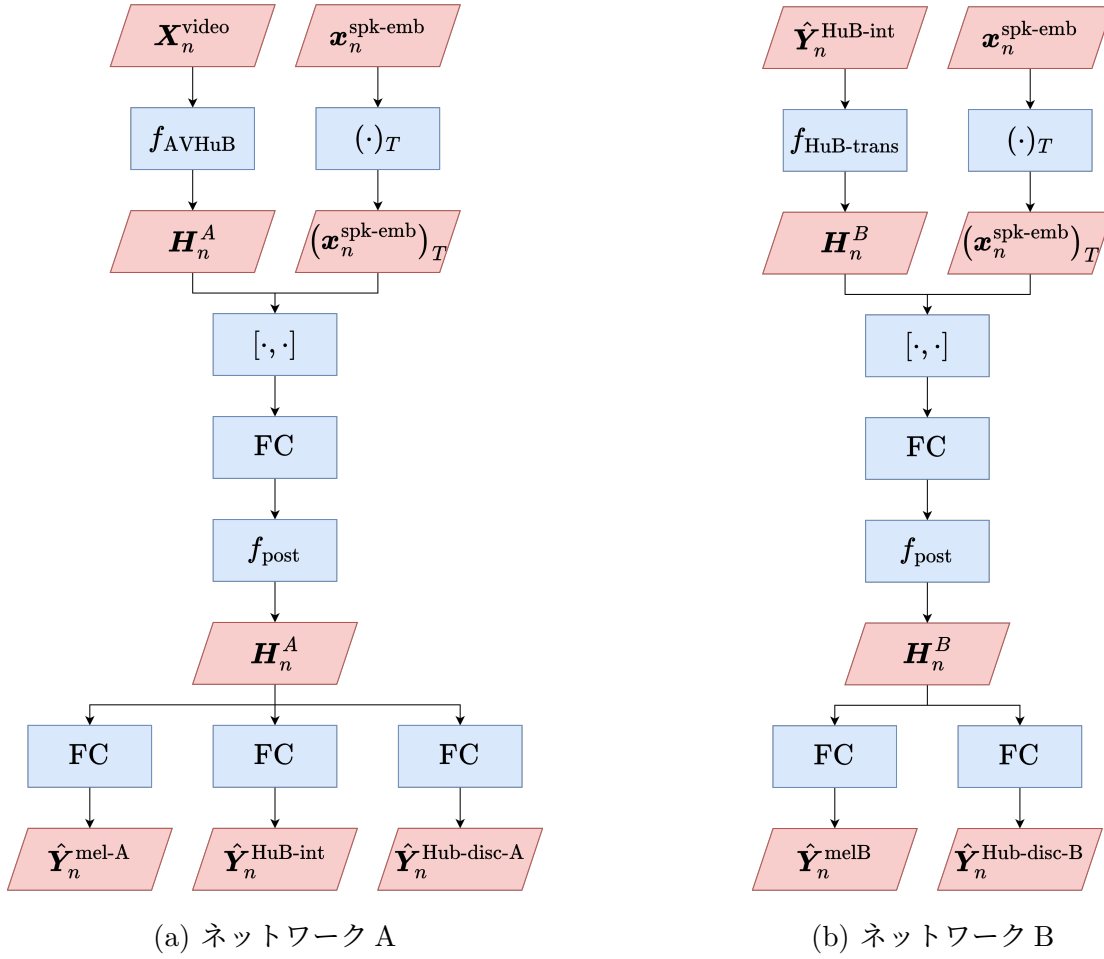


図 4.3: ネットワーク A とネットワーク B の構造

$$H_n^B = f_{\text{post}}(H_n^B; \theta_{\text{B-post}}) \quad (4.17)$$

$$\hat{Y}_n^{\text{mel-B}} = \text{FC}(H_n^B; \theta_{\text{B-fc-mel}}) \quad (4.18)$$

$$\hat{Y}_n^{\text{HuB-disc-B}} = \text{FC}(H_n^B; \theta_{\text{B-fc-HuB-disc}}) \quad (4.19)$$

$\theta_{\text{B-fc-spk}}, \theta_{\text{B-post}}, \theta_{\text{B-fc-mel}}, \theta_{\text{B-fc-HuB-disc}}$ 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、マスクされたフレームにおける教師ラベルを推定する学習方法である。3.2.8 節より、Masked Prediction では欠損された入力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 の推定残差を、大量の音声データから学習された

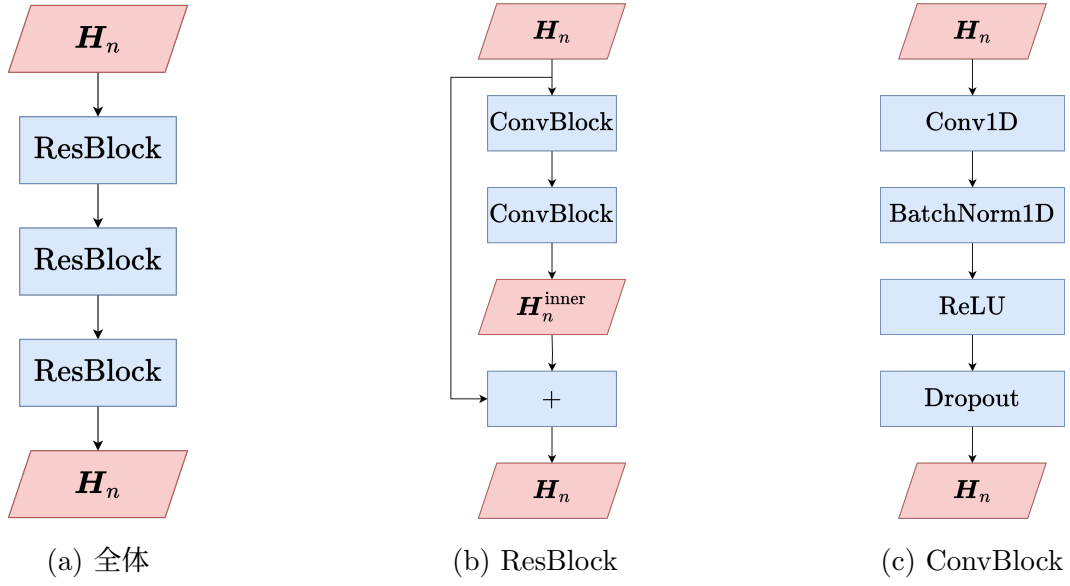


図 4.4: 後処理層 f_{post} の構造

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 $\hat{\mathbf{Y}}_n^{\text{mel-B}}$ と HuBERT 離散特徴量に対するロジット $\hat{\mathbf{Y}}_n^{\text{HuB-disc-B}}$ を前処理層に通し、扱いやすい形状に変換する。これは、

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

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

と表される。 $\mathbf{H}_n^{\text{mel-voc}} \in \mathbb{R}^{D^{\text{mel-voc}} \times T_n^{\text{HuB}}}$ は、 $\hat{\mathbf{Y}}_n^{\text{mel-B}}$ 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 $\hat{\mathbf{Y}}_n^{\text{HuB-disc-B}}$ 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、 $\mathbf{H}_n^{\text{HuB-voc}} \in \mathbb{R}^{D^{\text{HuB-voc}} \times T_n^{\text{HuB}}}$ は、 $\hat{\mathbf{Y}}_n^{\text{HuB-disc-B}}$ 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、 $\mathbf{H}_n^{\text{mel-voc}}, \mathbf{H}_n^{\text{HuB-voc}}$ を入力として、音声波形を生成する。これは、

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

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。

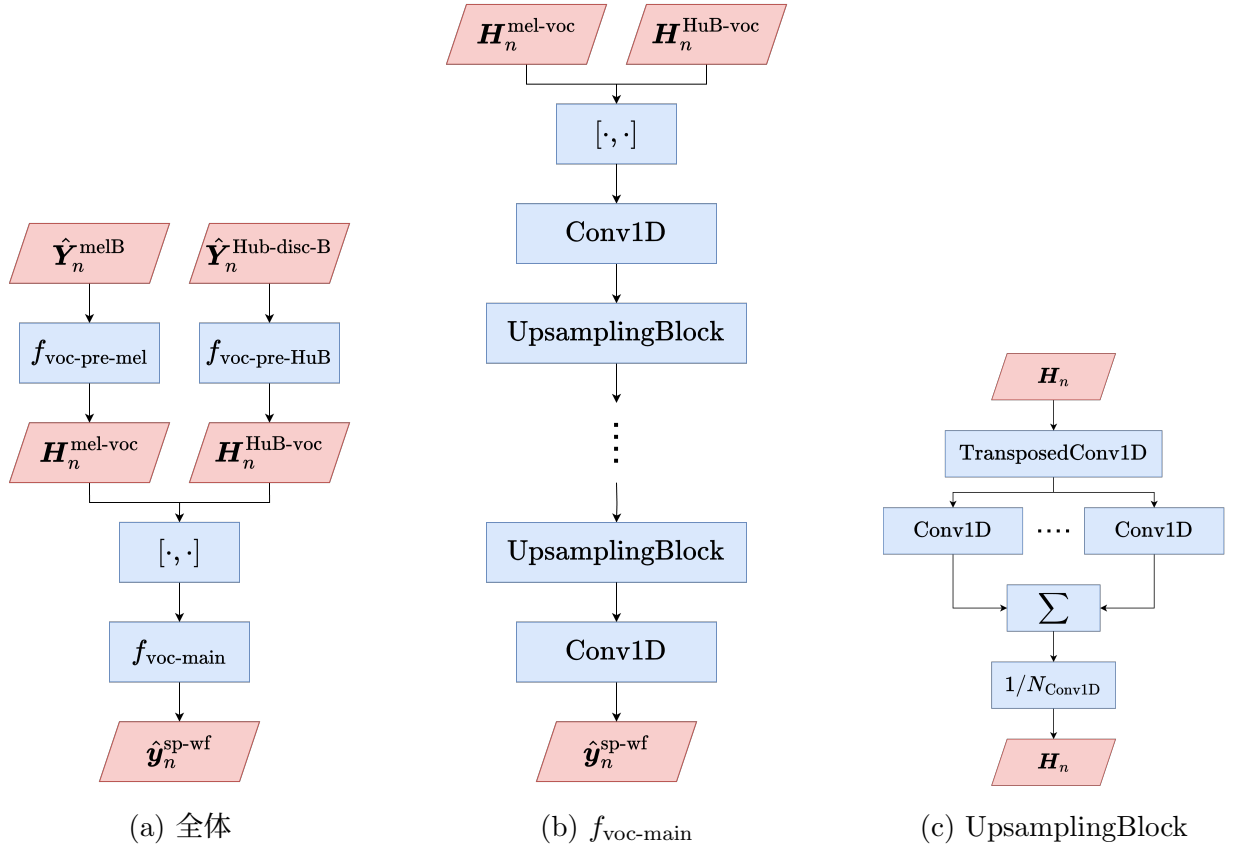


図 4.5: ボコーダの構造

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、

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

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、

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

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 2 人ずつから収録された合計 4 人分のデータから構成される。分割は、A から H セットを学習データ、I セットを検証データ、J セットをテストデータとした。各分割ごとの文章数を表 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 1 つ追加したため、合計 101 クラスとして扱った。

4.2.3 本実験で利用した事前学習済みモデルについて

話者ベクトルの計算に用いた話者識別モデルの事前学習済み重みには、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 1 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>

⁴https://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 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は 768 次元 50 Hz である．

4.2.4 本実験で独自に構築したモデルについて

本実験で独自に構築したモデルは、ネットワーク A と B における f_{post} と、ボコーダ f_{voc} である．

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

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 2 つの特徴量を次元方向に結合することで、256 次元 50 Hz の特徴量が得られる．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 1 層に対し、その後の特徴量抽出は 15 種類の異なるカーネルサイズ、ダイレーションを設定した畳み込み層によって行われる．最後に、16 次元 16 kHz の特徴量を畳み込み層に通して 1 次元まで次元を削減することで、16 kHz の音声波形が得られる．

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 1 回重みを更新するよう Gradient Accumulation を適用した。モデルに入力する動画の秒数は 10 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 5 段階でグリッドサーチした。

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

ボコーダについて、ここでははじめに 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。

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

4.2.6 客観評価

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

$$\text{WER}(s_1, s_2) = \frac{I + D + R}{|s_1|} \quad (4.25)$$

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。

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

4.2.7 主観評価

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

1. 日本語話者である方
2. 聴覚に異常のない方
3. イヤホンあるいはヘッドホンでの実施が可能な方
4. 静かな環境での実施が可能な方

⁶<https://github.com/openai/whisper>

5. 以前に行った主観評価実験に参加されなかった方

とした、実験項目は、

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

である。

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

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

1. 提示された音声サンプルを1回再生し、発話内容を聞き取る。
2. 「発話内容を表示」ボタンを押し、本来の発話内容を表示する。
3. 聴取者が想定していた発話内容と本来の発話内容を照らし合わせ、音声の聞き取りやすさを5段階評価する。

5段階評価の回答項目は以下のようにした。

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

実験に利用した音声サンプルについて、練習試行では検証データである ATR 音素バランス文の I セット、本番試行ではテストデータである ATR 音素バランス文の J セットを用いた。被験者ごとの評価サンプルの割り当て方法をアルゴリズム 3 に示す。sentences は文章のリスト、methods が手法のリスト、speakers が話者のリスト、 N_p 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、2つのことに注意した。1つ目は、各被験者がユニークな発話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。2つ目は、各手法がなるべく均等な回数出現することである。今回の実験は手法の比較を目的としたため、各被験者がすべての手法をなるべく均等な回数評価できるようにした。また、評価時に音声サンプルを1回だけ聞けるようにしたことについて、代用音声をコミュニケーションツールとして利用する場面を想定したとき、会話において何度も聞き返されることは利用者のストレスになると考えられる。よって、本実験では1回の発話で意図した発話内容をどの程度聞き取ってもらえるかを聞き取りやすさとして評価したいと考え、この制限を設けた。

Algorithm 3 被験者に対する評価サンプル割り当て方法（明瞭性）

```

1: Input: sentences, methods, speakers,  $N_p$ 
2: allAssignments = {}
3: for participantId = 0 to  $N_p - 1$  do
4:   assignments = []
5:   Randomly shuffle sentences
6:   Randomly shuffle methods
7:   for  $i = 0$  to  $\text{len}(\text{sentences}) - 1$  do
8:     sentence = sentences[ $i$ ]
9:     method = methods[ $i \bmod \text{len}(\text{methods})$ ]
10:    speaker = Randomly select from speakers
11:    Append [sentence, method, speaker] to assignments
12:  end for
13:  allAssignments[participantId] = assignments
14: end for
15: return allAssignments

```

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

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

5段階評価の回答項目は以下のようにした。

1. 全く似ていなかった
2. あまり似ていなかった
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）

```

1: Input: sentences, methods, speakers,  $N_p$ 
2: allAssignments = {}
3: for participantId = 0 to  $N_p - 1$  do
4:   assignments = []
5:   Randomly shuffle sentences
6:   Randomly shuffle methods
7:   for  $i = 0$  to  $\text{len}(\text{sentences}) - 1$  do
8:     sentenceEval = sentences[ $i$ ]
9:     method = methods[ $i \bmod \text{len}(\text{methods})$ ]
10:    speaker = Randomly select from speakers
11:    sentenceGT = Randomly select from sentences
12:    Append [sentenceEval, method, speaker, sentenceGT] to assignments
13:  end for
14:  allAssignments[participantId] = assignments
15: end for
16: 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段階評価の回答項目からランダムに選択した。被験者には、ダミー音声提示された場合はその音声自体の明瞭性や類似性とは無関係に、必ず音声で指定された評価値を選択するよう伝えた。練習試行では1つ、本番試行では2つ混入させ、本番試行でダミー音声に対する評価値を誤った被験者については、適当に回答した恐れがあると判断し、最終的な統計処理には用いなかった。

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

4.3 結果

4.3.1 客観評価 1: ベースラインと提案手法の比較

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

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

ベースラインは、提案手法におけるネットワーク A で、メルスペクトログラムと HuBERT 離散特徴量のみを予測する場合であり、損失関数は L_B を用いた。ネットワーク A は提案手法自体ではないが、その構成要素となっているため客観評価指標を確認した。ネットワーク B (Randomized) は、HuBERT Transformer 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良値を下線で示し、各手法ごとに最適だと判断した $\lambda_{\text{HuB-disc}}$ の行を太字で示している。

ベースラインでは、 $\lambda_{\text{HuB-disc}}$ が 0.001 のときに WER が最も低く、0.01 のときに話者類似度が最も高くなった。現状 WER の高さが特に課題であるため、今回は 0.001 が最適だと判断した。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。 $\lambda_{\text{HuB-disc}}$ を大きくすることで HuBERT 離散特徴量についての Cross Entropy Loss の下がり方がより早く、より低い値に達するように変化しており、係数の大きさが学習に影響を与えていることがわかる。また、最適とした 0.001 の場合、学習データと検証データの両方において、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 がバランスよく下がっていることがわかる。客観評価

指標が同程度であった 0.0001 と 0.01 では、学習曲線にも顕著な差は見られなかった。ここで、0.1 と 1.0 では、0.01 以下の場合と比較して話者類似度が顕著に低下したが、この時学習データに対するメルスペクトログラムについての MAE Loss が十分に下がり切らないまま、Early Stopping によって学習が中断されていたことが分かる。これより、話者性を適切に反映するためには、学習データに対するメルスペクトログラムについての MAE Loss を低下させることが重要だと考えられる。

ネットワーク A では、提案手法の構成要素であるため最適な手法の選択は行わなかった。ベースラインとの違いは HuBERT 中間特徴量についての MAE Loss が損失に含まれることであるが、客観評価指標の値はベースラインと概ね同様であった。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り、客観評価指標の結果と合わせて妥当だと考える。

ネットワーク B (Randomized) では、 $\lambda_{\text{HuB-disc}}$ が 0.1 の時に WER が最も低く、0.0001 の時に話者類似度が最も高くなった。ここでは WER の低さを優先し、0.1 が最適だと判断した。図 4.8 に学習曲線を示す。最適とした 0.1 の場合、学習データと検証データの両方において、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 がバランスよく下がっていることがわかる。1.0 では、0.1 以下の場合と比較して話者類似度の低下が顕著であったが、この時学習データに対するメルスペクトログラムについての MAE Loss が十分に下がり切らないまま、Early Stopping によって学習が中断されていたことが分かる。これはベースラインと同様の傾向であった。

ネットワーク B (Pretrained) では、 $\lambda_{\text{HuB-disc}}$ が 0.1 の時に WER が最も低くなった一方で、0.01 以下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、ネットワーク B (Randomized) とは異なる傾向を示した。今回は WER が低いことを優先して、0.1 が最適だと判断した。図 4.9 に学習曲線を示す。ネットワーク B (Randomized) と概ね同様であるが、 $\lambda_{\text{HuB-disc}} = 0.1$ の場合に学習データに対するメルスペクトログラムについての MAE Loss が十分に下がりきっていない傾向が見られる。これまでの結果より、これは話者類似度の低下につながると考えられるため、ネットワーク B (Pretrained) が $\lambda_{\text{HuB-disc}} = 0.1$ の場合に話者類似度を高い値で保てなかったのは、学習の結果として妥当だと考える。

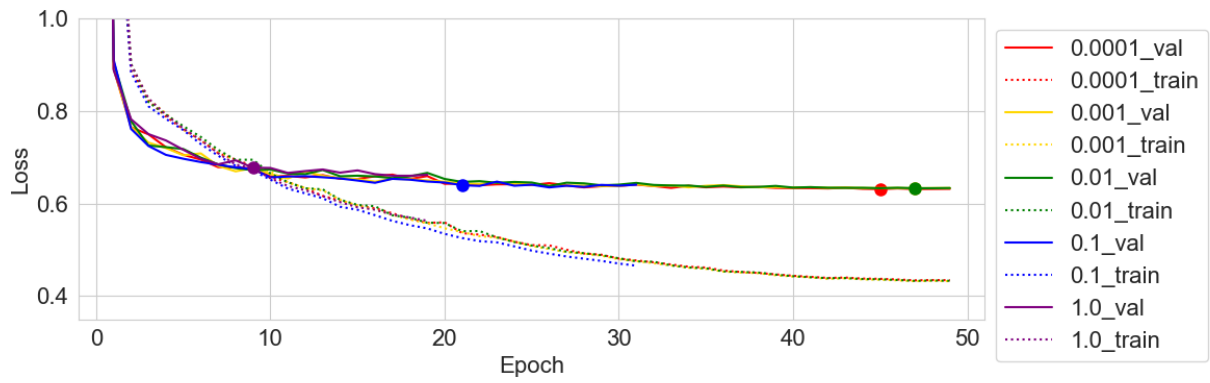
最後に、最適な $\lambda_{\text{HuB-disc}}$ における手法ごとの比較を行った結果を表 4.4 に示す。分析合成は、原音声から計算した特徴量を入力としてボコーダで逆変換した合成音声であり、本実験において合成音声により達成され得る上限を表す。ベースライン、ネットワーク B (Randomized)、ネットワーク B (Pretrained) は、それぞれ最適な $\lambda_{\text{HuB-disc}}$ だと判断したものを載せている。また、ベースライン、ネットワーク B (Randomized)、ネットワーク B (Pretrained) の中で、最良値を下線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

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

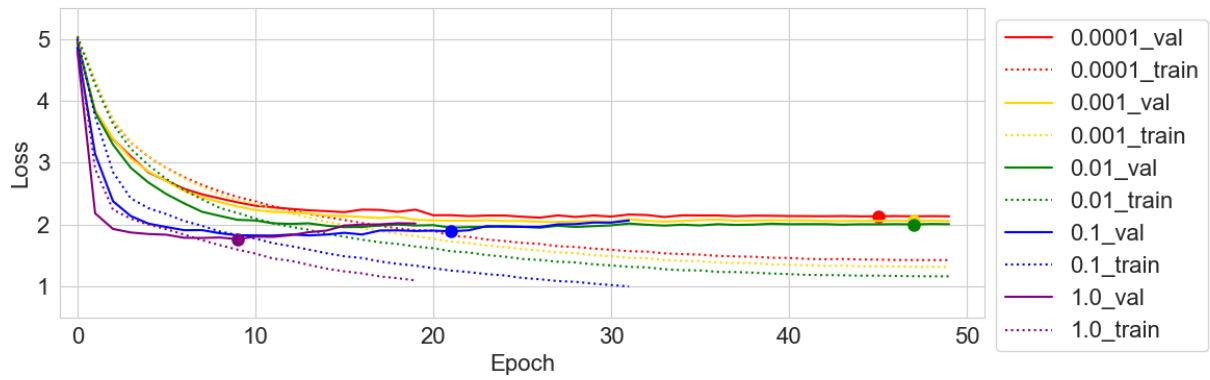
手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ベースライン	0.0001	57.3	0.833
ベースライン	0.001	<u>54.6</u>	0.836
ベースライン	0.01	56.2	<u>0.837</u>
ベースライン	0.1	58.2	0.784
ベースライン	1	59.0	0.685
ネットワーク A	0.0001	55.4	0.841
ネットワーク A	0.001	55.1	0.842
ネットワーク A	0.01	<u>53.4</u>	<u>0.843</u>
ネットワーク A	0.1	54.6	0.809
ネットワーク A	1	58.7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

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

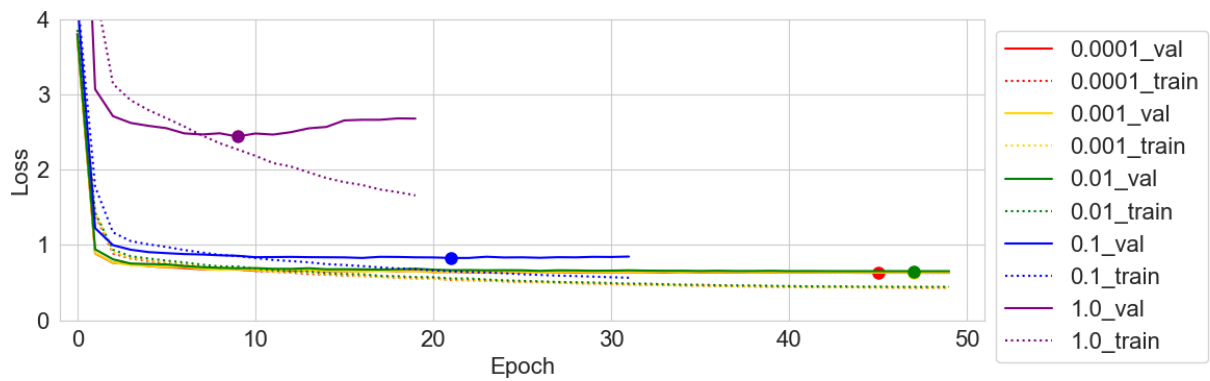
手法	$\lambda_{\text{HuB-disc}}$	WER [%]	話者類似度
ベースライン	0.001	54.6	0.836
ネットワーク B (Randomized)	0.1	45.3	<u>0.840</u>
ネットワーク B (Pretrained)	0.1	<u>44.2</u>	0.778
分析合成	-	3.7	0.956
原音声	-	3.7	1.000



(a) メルスペクトログラムについての MAE Loss

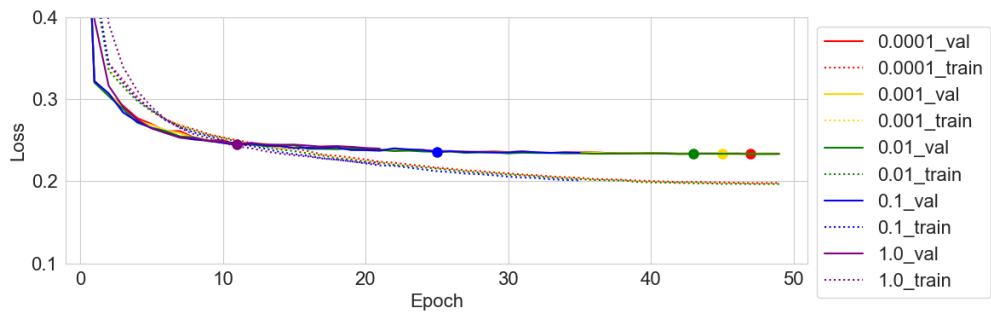


(b) HuBERT 離散特徴量についての Cross Entropy Loss

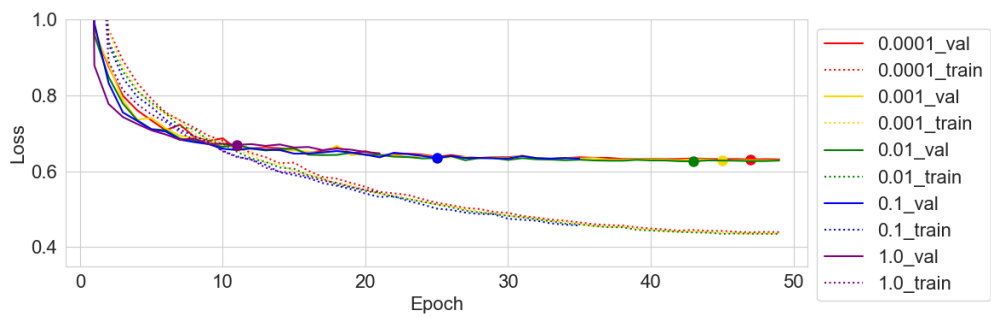


(c) 損失の合計値

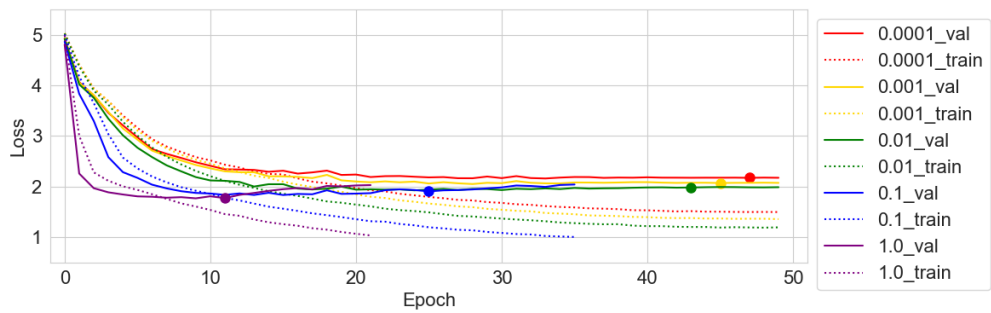
図 4.6: ベースラインにおける学習曲線



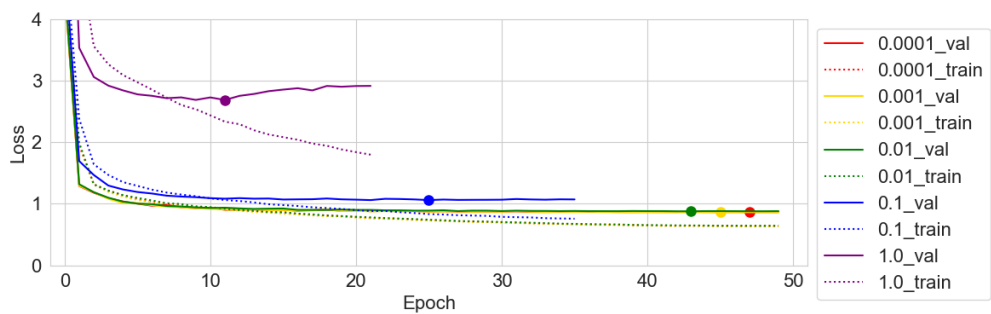
(a) HuBERT 中間特徴量についての MAE Loss



(b) メルスペクトログラムについての MAE Loss

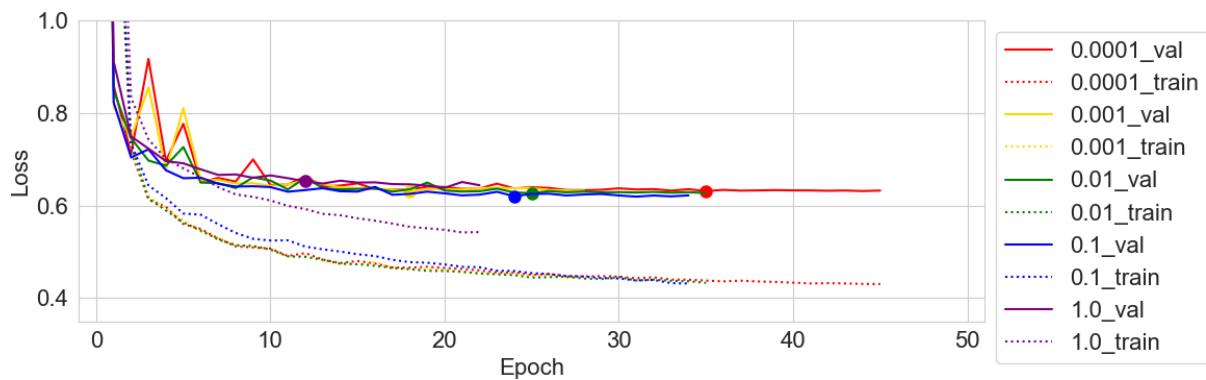


(c) HuBERT 離散特徴量についての Cross Entropy Loss

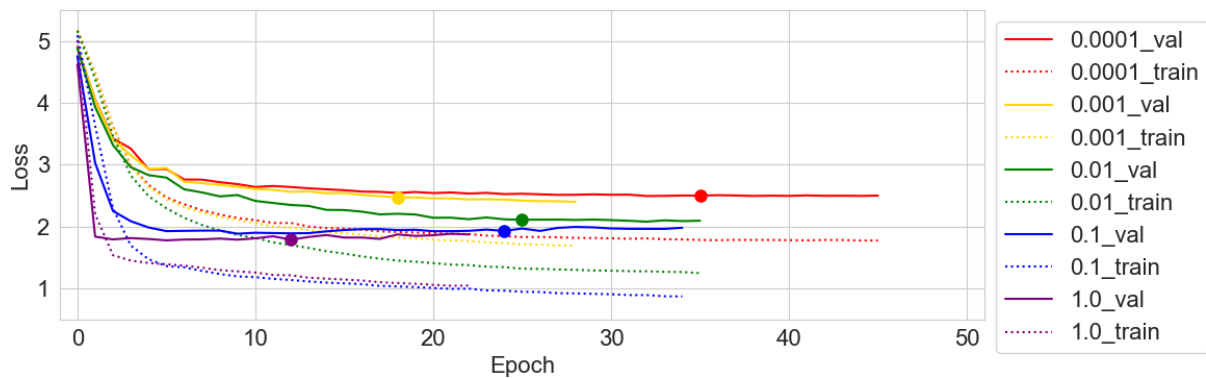


(d) 損失の合計値

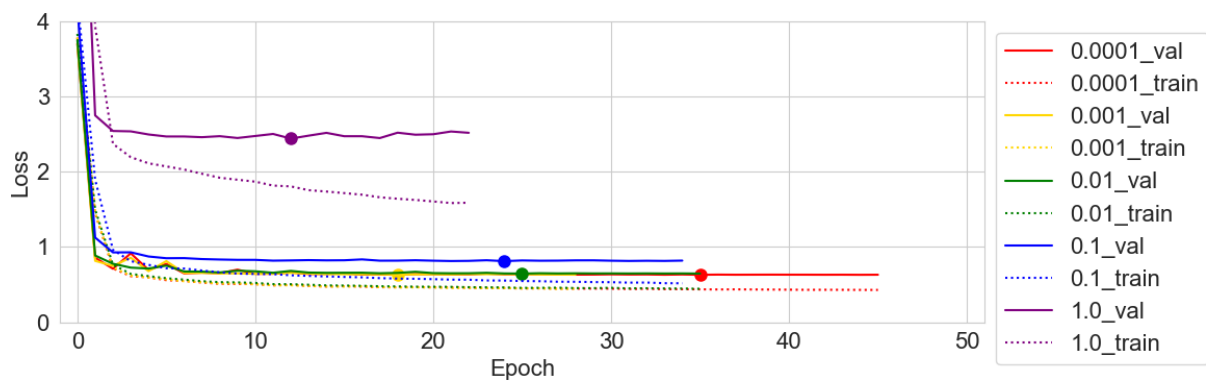
図 4.7: ネットワーク A における学習曲線



(a) メルスペクトログラムについての MAE Loss

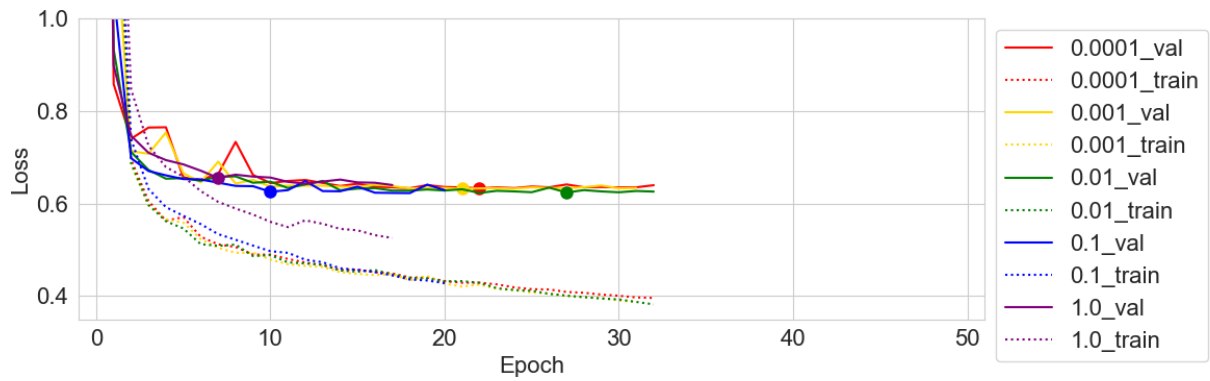


(b) HuBERT 離散特徴量についての Cross Entropy Loss

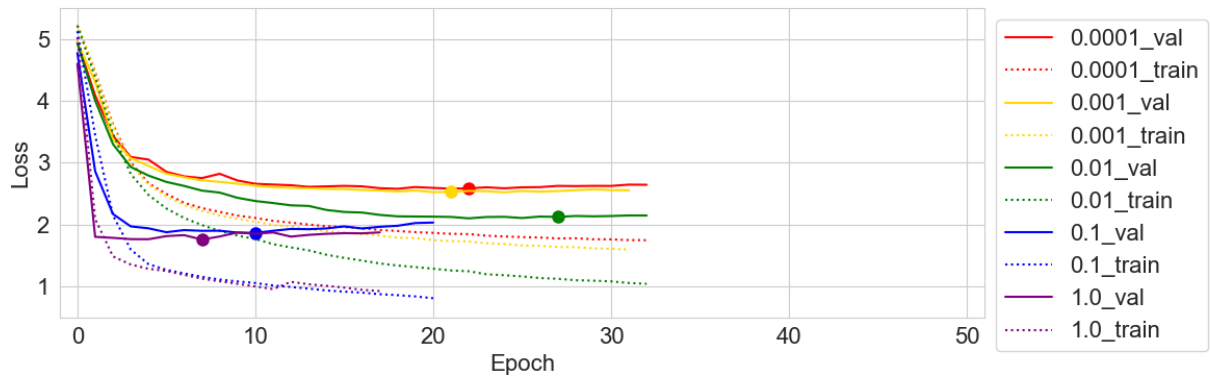


(c) 損失の合計値

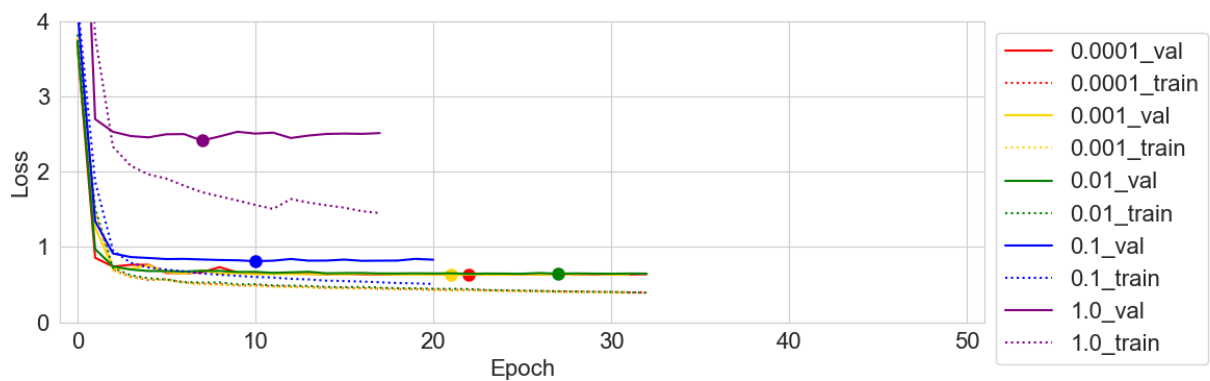
図 4.8: ネットワーク B (Randomized) における学習曲線



(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

4.3.2 客観評価 2: 提案手法のさらなる検討

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

まず、入力特徴量について検討した。4.3.1節では事前学習済み重みを用いることの有効性を調べる目的があったため、入力特徴量は HuBERT の事前学習時に揃えて、HuBERT 中間特徴量とした。しかし、ネットワーク B (Randomized) は事前学習済み重みを利用しないため、HuBERT 中間特徴量を入力とすることが意味をなしているか不明である。これに対し、ネットワーク A からマルチタスク学習によって同時に予測される、メルスペクトログラムと HuBERT 離散特徴量に対するロジットを組み合わせて入力する場合を比較した。メルスペクトログラムは、時間方向に隣接した 2 フレームを次元方向に積むことで 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まま用いた。これらを次元方向に結合することで 261 次元 50 Hz の特徴量を構成し、全結合層を通して HuBERT 中間特徴量と同じ 768 次元に変換することで、HuBERT Transformer 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。下線は掲載した手法全てにおける最良値を示す。これより、最適なネットワーク B (Randomized・Mel-HuB) はネットワーク B (Randomized) よりも話者類似度は 0.005 高いが、WER も 10.1% 高いことが分かる。図 4.10 にネットワーク B (Randomized・Mel-HuB) の学習曲線を示す。傾向自体はネットワーク B (Randomized) と概ね変わらないことがわかる。また、ネットワーク B (Randomized) では $\lambda_{\text{HuB-disc}}$ の値を大きくすることによって、0.0001 から 0.1 まで WER が低下傾向にあった。これに対し、ネットワーク B (Randomized・Mel-HuB) では WER の低下傾向は見られず、結果としてネットワーク B (Randomized) に劣る結果となった。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。

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

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

表 4.6: ネットワーク A におけるマルチタスク学習の有無による比較

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

した. ここでは, これまでネットワーク A で採用していた HuBERT 中間特徴量, メルスペクトログラム, HuBERT 離散特徴量を予測するマルチタスク学習に対し, HuBERT 中間特徴量のみを予測するシングルタスク学習の場合を比較した. 結果を表 4.6 に示す. 新たに検討した手法はネットワーク B (Randomized · A-SingleTask) とし, $\lambda_{\text{HuB-disc}} = 0.1$ の場合が最適だと判断した. これより, 最適なネットワーク B (Randomized · A-SingleTask) はネットワーク B (Randomized) よりも WER が 2.8% 低く, 話者類似度は 0.007 高いことがわかる. 図 4.11 にネットワーク B (Randomized · A-SingleTask) の学習曲線を示す. これはネットワーク B (Randomized) と概ね同様の傾向であった. 以上より, ネットワーク A では HuBERT 中間特徴量, メルスペクトログラム, HuBERT 離散特徴量を予測するマルチタスク学習を行うよりも, HuBERT 中間特徴量のみを推定する方が, ネットワーク B に対してより良い入力特徴量を与えられるのだと考えられる.

最後に, これまでネットワーク A とネットワーク B を 2 段階で学習させたのに対し, ネットワーク A からネットワーク B までを一度に学習させる方法を検討した. これにより, HuBERT 中間特徴量に表現力を制限させないことで, さらなる精度改善を狙った. ここでは, ネットワーク B の出力に対する損失関数 L_B のみを用いて学習させ, ネットワーク A の出力は単にネットワーク B への入力とした. ネットワーク A およびネットワーク B (Randomized) を未学習の状態で初期化した場合の結果を表 4.7 に示す. 新たに検討した手法はネットワーク E2E (Randomized) とし, $\lambda_{\text{HuB-disc}} = 0.1$ が最適だと判断した. これより, 最適なネットワーク E2E (Randomized) はネットワーク B (Randomized · A-SingleTask) よりも WER が 20.0% 高く, 話者類似度は 0.080 低いことがわかる. 図 4.12 に学習曲線を示す. 2 段階で学習させたネットワーク B (Randomized · A-SingleTask) と比較して損失の下がりが悪く, 客観評価指標が顕著に悪化したことは妥当だと考える. 以上より, 2 段階学習は学習の安定化につながっていたと言える. 次に, ここまでの検討で最も良好な結果を示した, ネットワーク B (Randomized · A-SingleTask) かつ $\lambda_{\text{HuB-disc}} = 0.1$ の場合における学習済み重みでネットワーク A とネットワーク B を初期化した場合の結果を表 4.8 に示す. 新たに検討した手法はネットワーク E2E (Pretrained) とし, ここでは $\lambda_{\text{HuB-disc}}$ の値は 0.1 のままで固定して, スケジューラである Cosine Annealing with Warmup の最大学習率 η_{max} を変更することでチューニングした. 最適値は $\eta_{\text{max}} = 5.0 \times 10^{-5}$

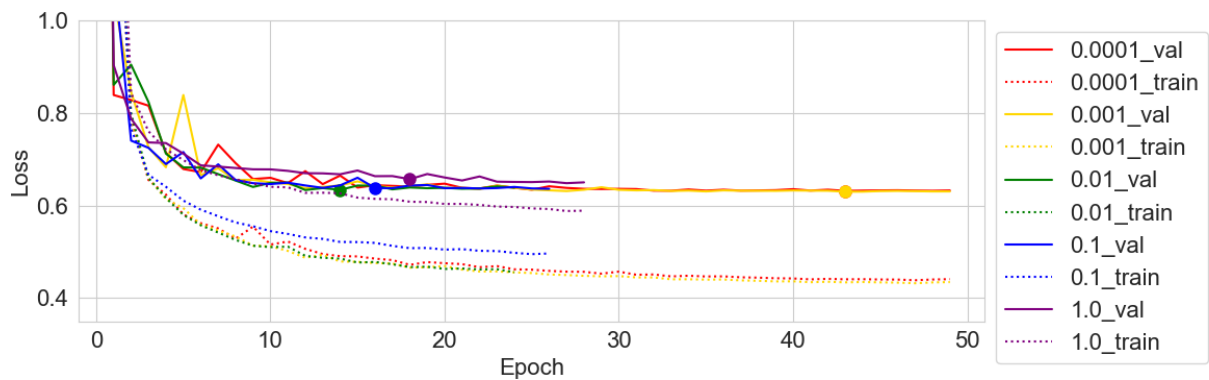
表 4.7: ネットワーク A からネットワーク B までを未学習の状態に初期化し、一度に学習させた場合の比較

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

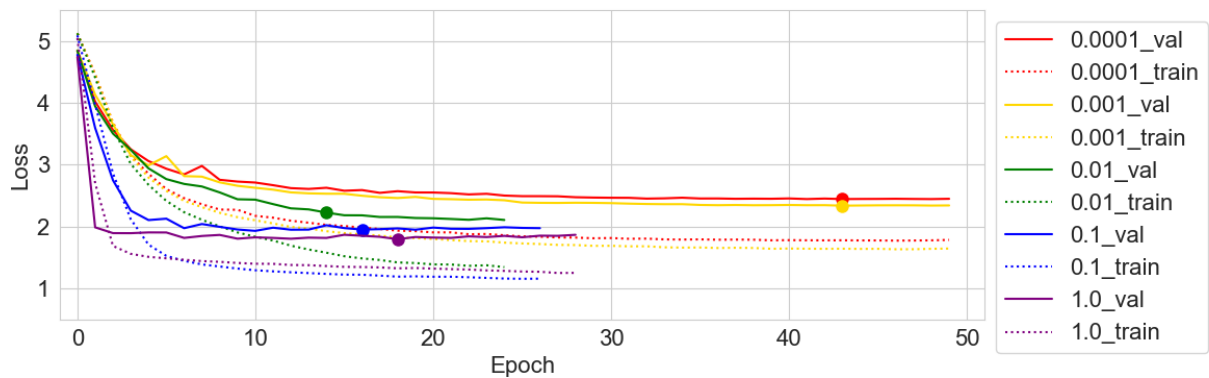
表 4.8: ネットワーク B (Randomized・A-SingleTask) かつ $\lambda_{\text{HuB-disc}} = 0.1$ の場合における学習済み重みでネットワーク A とネットワーク B を初期化し、一度に学習させた場合の比較

手法	η_{max}	WER [%]	話者類似度
ネットワーク E2E (Pretrained)	5.0×10^{-4}	44.6	0.851
ネットワーク E2E (Pretrained)	5.0×10^{-5}	43.5	<u>0.853</u>
ネットワーク E2E (Pretrained)	5.0×10^{-6}	43.6	<u>0.853</u>
ネットワーク B (Randomized・A-SingleTask)	5.0×10^{-4}	<u>42.5</u>	0.847

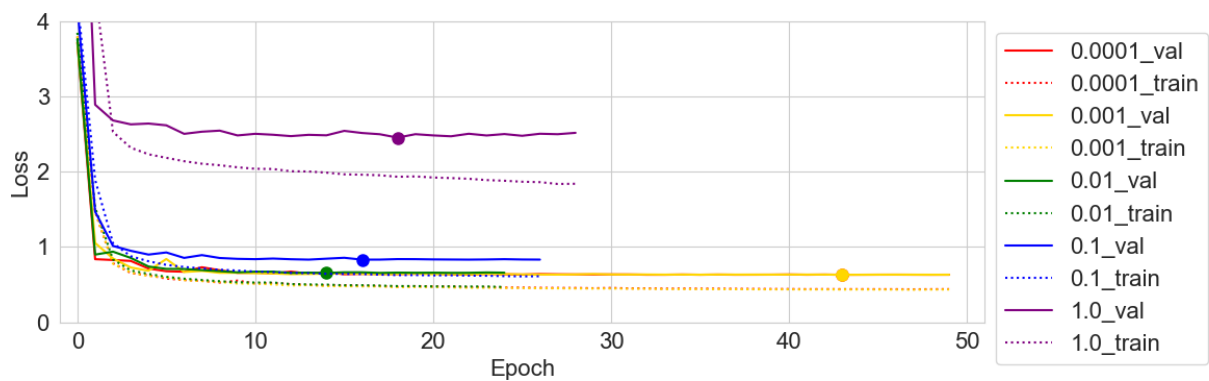
とした。これより、最適なネットワーク E2E (Pretrained) はネットワーク B (Randomized・A-SingleTask) よりも WER が 1.3% 高く、話者類似度も 0.006 高いことがわかる。図 4.13 に学習曲線を示す。線の色は η_{max} の違いを表す。損失が学習開始時からほとんど低下していないため、ネットワーク全体を再学習することは効果的でなかったと考えられる。



(a) メルスペクトログラムについての MAE Loss

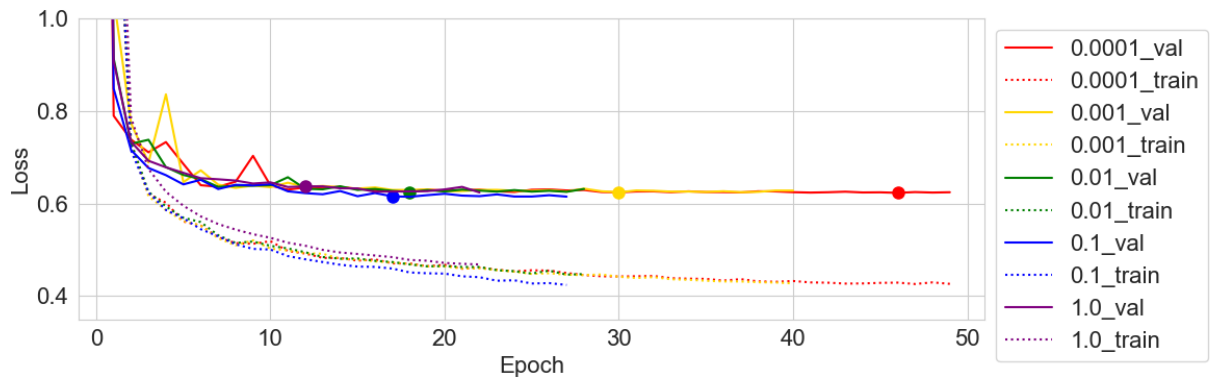


(b) HuBERT 離散特徴量についての Cross Entropy Loss

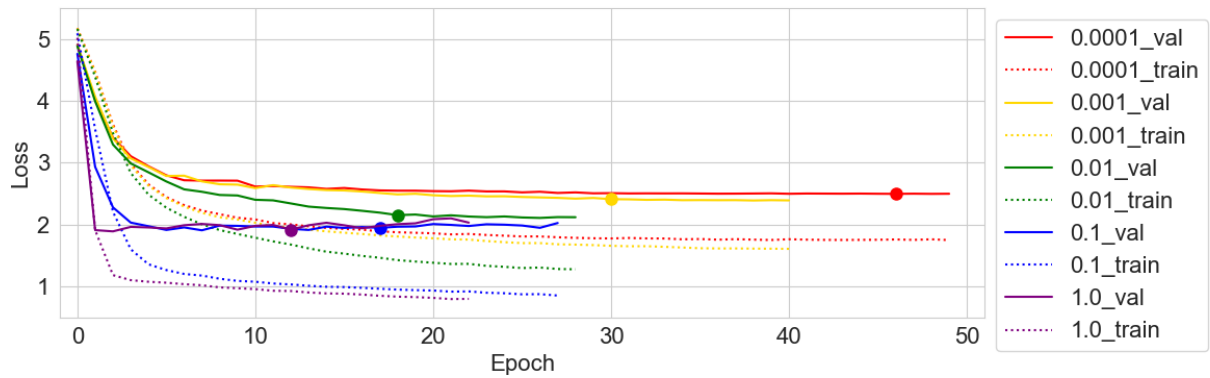


(c) 損失の合計値

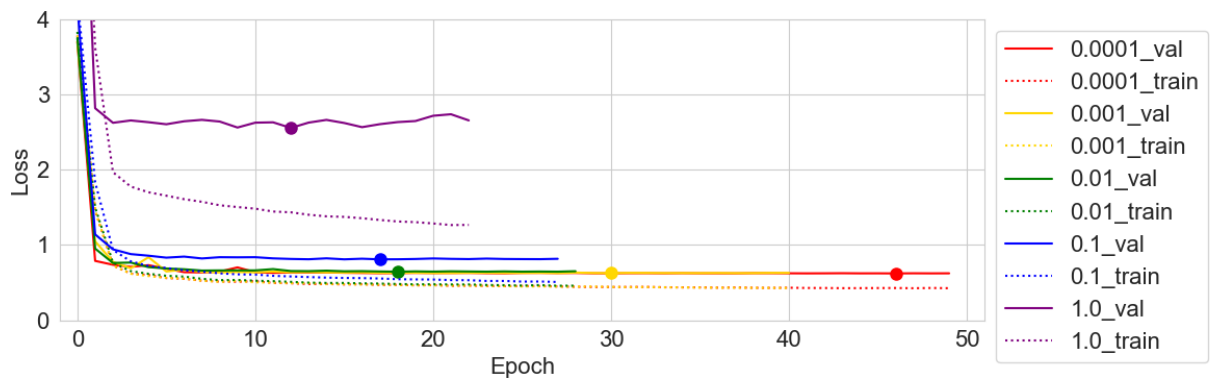
図 4.10: ネットワーク B (Randomized・Mel-HuB) における学習曲線



(a) メルスペクトログラムについての MAE Loss

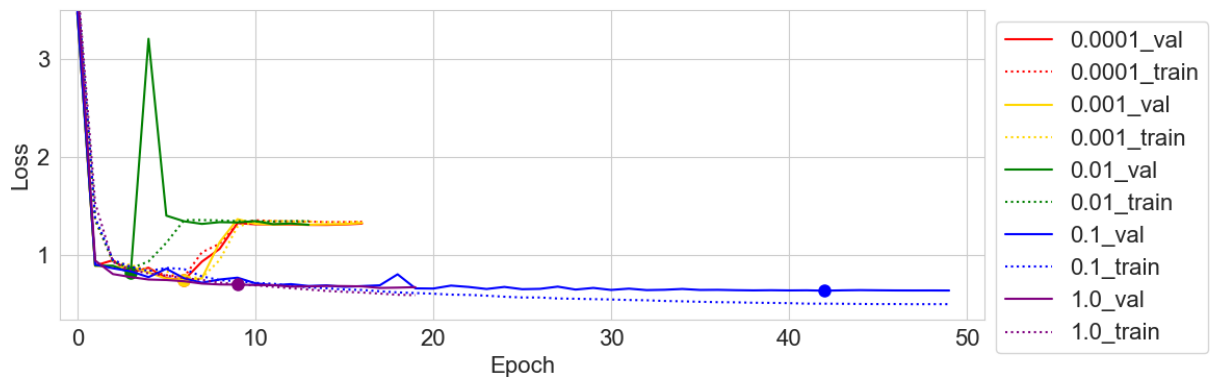


(b) HuBERT 離散特徴量についての Cross Entropy Loss

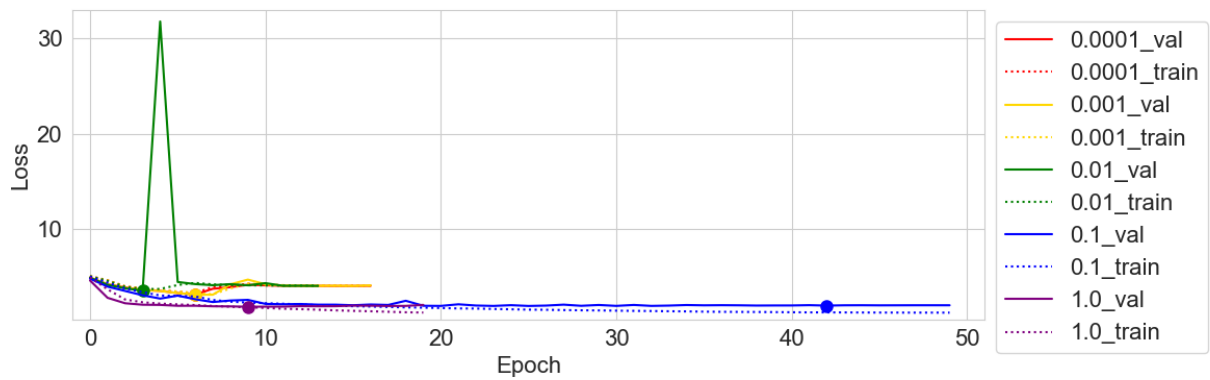


(c) 損失の合計値

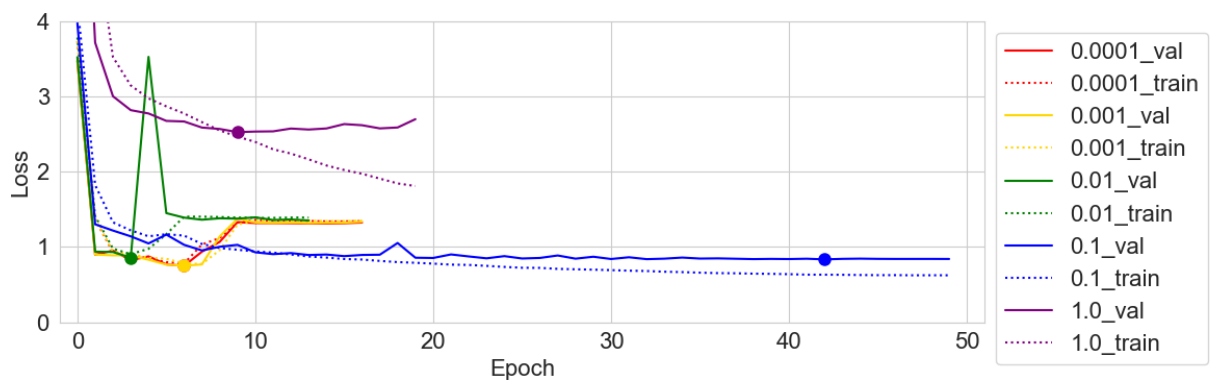
図 4.11: ネットワーク B (Randomized・A-SingleTask) における学習曲線



(a) メルスペクトログラムについての MAE Loss

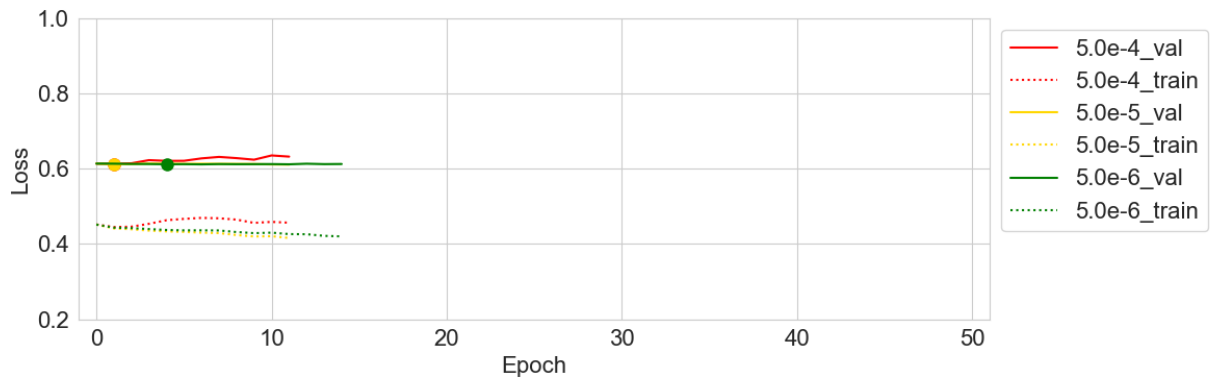


(b) HuBERT 離散特徴量についての Cross Entropy Loss

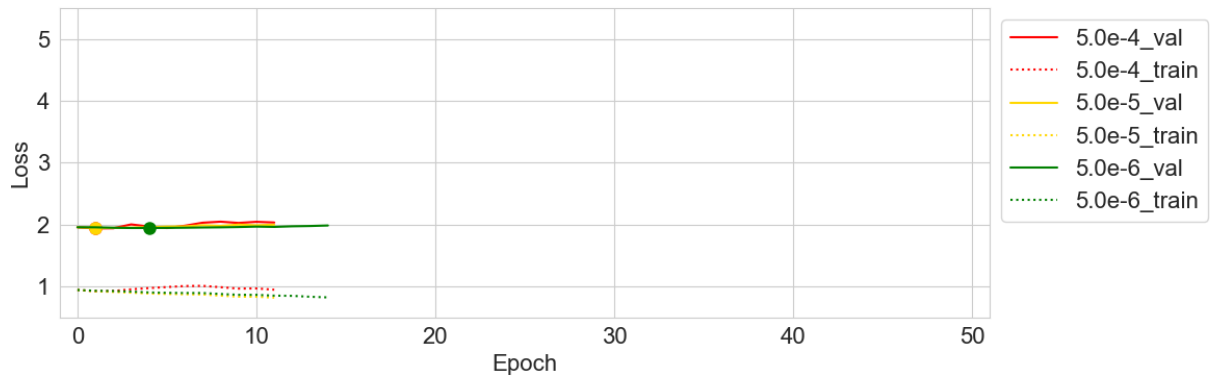


(c) 損失の合計値

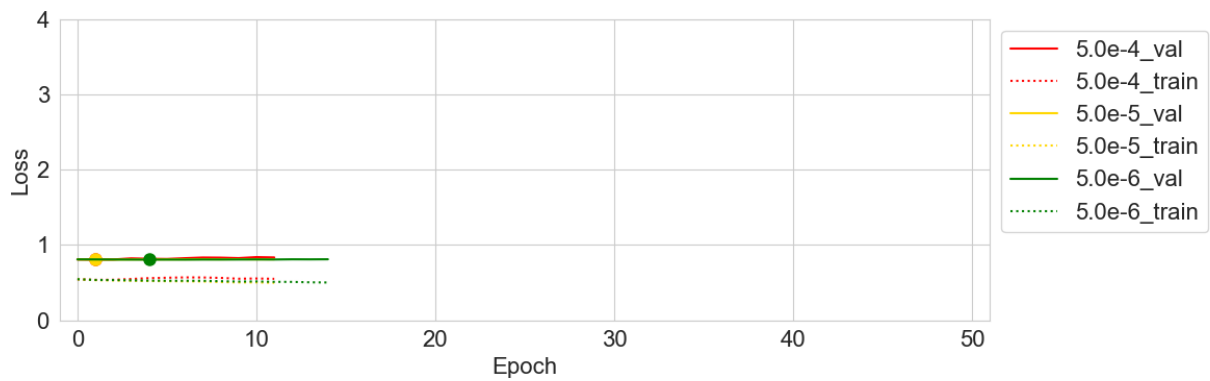
図 4.12: ネットワーク E2E (Randomized) における学習曲線



(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 4.13: ネットワーク E2E (Pretrained) における学習曲線

4.3.3 主観評価

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

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

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

被験者に対するアンケート結果について述べる。年齢層は21歳から62歳に渡り、箱ひげ図にてその分布を示した(図4.14)。性別の内訳は男性32名、女性43名であり、実験に利用した音響機器の内訳はヘッドホンが19名、イヤホンが56名であった。

明瞭性と類似性の評価値について、手法ごとの標本平均と95%信頼区間を表4.9に示す。また、各手法の組み合わせについて、平均値の差の検定(Welchのt検定)を行った結果を表4.10, 4.11に示す。表における (i, j) 成分は、 i 行目の手法に対する評価値の母平均を μ_i 、 j 列目の手法に対する評価値の母平均を μ_j とすると、帰無仮説を $\mu_i = \mu_j$ 、対立仮説を $\mu_i > \mu_j$ と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、表の文字列は以下の意味である。

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

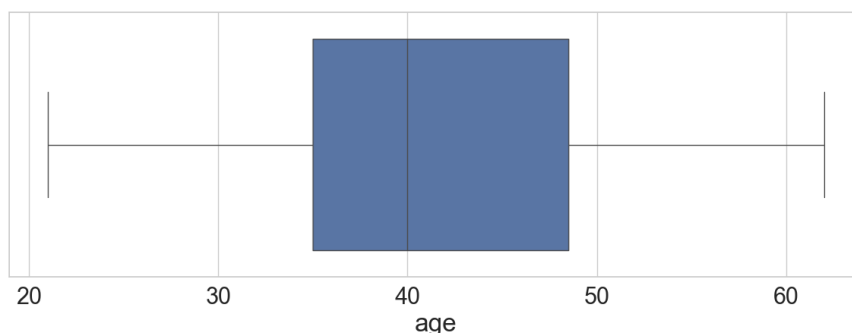


図 4.14: 主観評価実験における被験者の年齢層

表 4.9: 主観評価実験の結果より計算した標本平均と 95%信頼区間

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

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

	AbS	B (Rand, A-S)	B (Rand)	Baseline
GT	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}

また、本実験における有意水準は 5%とした。

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

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

4.4 考察

4.4.1 ネットワーク B (Randomized) について

ネットワーク B (Randomized) は客観評価において、メルスペクトログラムと HuBERT 離散特徴量に対するロジットを入力特徴量としたネットワーク B (Randomized · Mel-HuB) を上

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

	AbS	B (Rand, A-S)	B (Rand)	Baseline
GT	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}

回る性能を示した。これより、ネットワーク B への入力として、HuBERT 中間特徴量はメルスペクトログラムと HuBERT 離散特徴量に対するロジットを組み合わせた特徴量よりも優れていたと考えられる。HuBERT 中間特徴量は、音声波形を HuBERT の畳み込みエンコーダによって変換した結果であった。これは HuBERT の事前学習時に行われる Masked Prediction において Transformer 層の入力となる特徴量であり、Transformer 層では HuBERT 中間特徴量のうち観測できる区間のみを利用して、マスクされた区間における教師ラベルを予測する必要がある。これより、畳み込みエンコーダは、音声波形の系列長を圧縮する過程で、Transformer 層によって行われる文脈的構造の考慮に役立つ情報を、限られた次元の中にできるだけ詰め込むように最適化されると考える。よって、ネットワーク B (Randomized) における HuBERT Transformer 層はランダム初期化されていたものの、HuBERT 中間特徴量自体が文脈的構造の考慮に適した情報を含んでいるから、メルスペクトログラムと HuBERT 離散特徴量に対するロジットを入力特徴量とする場合よりも特徴抽出が行いやすくなり、結果として高い性能に達したと考えられる。

次に、ネットワーク B (Randomized・A-SingleTask) は客観評価と主観評価の両面において、ネットワーク B (Randomized) を上回る性能を示した。これより、HuBERT 中間特徴量を予測するネットワーク A は、メルスペクトログラムと HuBERT 離散特徴量を同時に予測するマルチタスク学習を行わず、HuBERT 中間特徴量のみを予測するよう学習した方が、ネットワーク B に対するより良い入力特徴量を与えられると考えられる。ここで、ネットワーク B (Randomized) とネットワーク B (Randomized・Mel-HuB) 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。よって、HuBERT 中間特徴量と、メルスペクトログラムおよび HuBERT 離散特徴量は、異なる情報を含んだ特徴量だと考えられる。結果として、ネットワーク B への入力として用いる HuBERT 中間特徴量を予測する上では、メルスペクトログラムおよび HuBERT 離散特徴量についての損失の干渉が悪影響を及ぼしたと考えられる。

次に、ネットワーク E2E (Pretrained) では、ネットワーク B (Randomized・A-SingleTask) における重みを初期値としてネットワーク全体を再学習させることでさらなる改善を狙ったが、これは有効でなかった。よって、HuBERT 中間特徴量を継ぎ目として 2 段階で学習することはネットワークの表現力を損なうものではなかったと考えられる。HuBERT 中間特徴量は元々

DNN の中間表現であるから、これを2つのネットワークの継ぎ目としても表現力を損なわなかったと考えられる。

最後に、本実験では損失関数の重み係数 $\lambda_{\text{HuB-disc}}$ について5種類の値によるグリッドサーチを一貫して行い、検討した手法の性能は $\lambda_{\text{HuB-disc}}$ に大きく依存することがわかった。特に、 $\lambda_{\text{HuB-disc}}$ の値を大きくすることによって HuBERT 離散特徴量に対する Cross Entropy Loss の低下が促進される一方で、メルスペクトログラムの MAE Loss が十分に下がらないまま Early Stopping が発生するケースが散見され、マルチタスク学習において両方の損失をバランスよく最小化することの難しさが明らかとなった。実数値をとる重み係数についてのグリッドサーチには限界があるから、これを行うこと自体がモデルの性能の妨げになると考えられる。これに対し、マルチタスク学習の先行研究では、損失の下がり具合に基づいて適応的に重みを調整する方法 [37, 38] が提案されている。これらは、各タスクの損失の勾配のノルムの調整にあたる。また、マルチタスク学習においては各タスクの損失が競合する Negative Transfer [39] が発生する可能性があるが、これに対して各タスクの損失の勾配の向きを調整する方法 [40] も提案されている。こういった手法により、グリッドサーチおよび静的な重み係数を脱却することで、ネットワークの性能改善および実験時間の短縮が期待できる。

4.4.2 ネットワーク B (Pretrained) について

ネットワーク B (Pretrained) は、HuBERT の事前学習済み重みを用いて学習を行ったが、WER を低下させつつ話者類似度を維持することができなかったため、ネットワーク B (Randomized) に劣る結果となった。しかし、これは単に事前学習済み重みが話者性の抽出に適していなかったわけではないと考える。実際、話者識別の先行研究 [41] では、音声波形を HuBERT によって変換した特徴量を入力として話者識別モデルを学習させた場合、窓長 25 ms、シフト幅 10 ms で計算された 40 次元のメルスペクトログラムを入力とする場合よりも、Equal Error Rate が低いことが示されている。ここで、HuBERT は事前学習時のまま重みを固定し、単なる特徴抽出器として用いられた。話者性の含まれるメルスペクトログラムに対し、話者識別をより高い精度で行える入力特徴量を音声波形から抽出できたことになるから、HuBERT が話者性抽出に適していないわけではないと考える。また、本研究では HuBERT Transformer 層の出力に話者ベクトルを結合する構成としたため、仮に話者性が損なわれていたとしてもそれを補完できたのではないかと考える。

これに対し、ネットワーク B (Pretrained) が有効性を示さなかった理由は2つ考えられる。1つ目は、損失関数の重み係数 $\lambda_{\text{HuB-disc}}$ の適切な値が、グリッドサーチした範囲に存在しなかったことである。ネットワーク B (Randomized) では適切な値が含まれていたが、グリッドサーチの候補は決め打ちであるから、その探索範囲に限界があった。この課題に対しては、4.4.1節で挙げた動的な重み係数の調整が有効だと考える。2つ目は、事前学習時との入力特徴量のギャップである。本実験では、HuBERT の事前学習に合わせて HuBERT 中間特徴量を Transformer 層への入力とした。しかし、事前学習時に用いられるマスクベクトルが存在しなかったこと、

原音声から得られる特徴量が存在しなかったことがギャップとなり、転移学習の妨げになった可能性がある。この課題に対しては、ネットワーク A から予測された HuBERT 中間特徴量の一部区間に、マスクベクトルや原音声から得られた HuBERT 中間特徴量を混入させることで、事前学習時の条件に近づける方法が考えられる。具体例として、ネットワーク A から予測された HuBERT 中間特徴量に、原音声から得られる HuBERT 中間特徴量を混入させる方法が考えられる。原音声から得られた特徴量の混入率を p とする。 p が 1 に近いほど原音声の割合が高くなり、ネットワーク B が取り組む問題は容易になると予想される。一方で、 p を学習の進行に伴って徐々に小さくするスケジューリングを行えば、ネットワーク B が取り組む問題の難易度が徐々に高まる。このような方法はカリキュラム学習と呼ばれ、タスクの難易度を段階的に調整することで効率的な学習を可能にすることが知られている。最終的に $p = 0$ として転移学習を完了すれば、初めから $p = 0$ で行った本実験よりも高い精度を達成できる可能性がある。

5 結論

謝辭

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

- [38] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1871–1880, 2019.
- [39] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.
- [40] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, Vol. 33, pp. 5824–5836, 2020.
- [41] Zhengyang Chen, Sanyuan Chen, Yu Wu, Yao Qian, Chengyi Wang, Shujie Liu, Yanmin Qian, and Michael Zeng. Large-scale self-supervised speech representation learning for automatic speaker verification. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6147–6151. IEEE, 2022.