設計仕様書

聖徳・平原・土屋チーム

注意

ここの内容は，頭の中で考えたものであり，
うまくいかない可能性がある．(もはや絶対うまくいかない)
適宜仕様変更して

あと，この資料だけでは分からないと思うから聞いて

目次

- ネットワーク概要
- 回路概要
- 決めごと・その他

目次

- ネットワーク概要
- 回路概要
- 決めごと・その他

読む前に

ネットワークを理解する上で参考にして

学習させる時に使ったプログラム
/notebooks/kaomoji_generator_complin.ipynb

ネットワークをNumpyで書き直したもの
/tools/kmj_gen_np.py

ビット精度を検証する時に使ったプログラム
/tools/kmj_gen.py

作るもの

タイトル（仮）

「Kaomoji Generator with Autoencoder」

（タイトルに「ネットワークはMLP Mixerから着想しました」的なの入れたい）

ざっくり説明

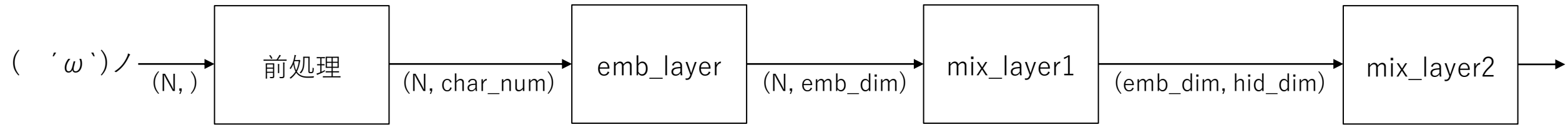
入力：(　´ω`)ノ　→　出力：(　´ω`)ノ

となるものを作りたい

パラメータの定義(ネットワーク)

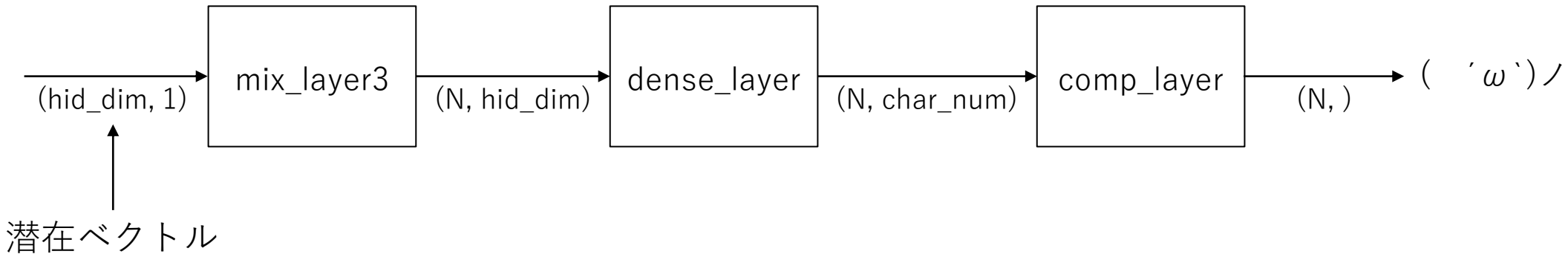
パラメータ名	値（現時点）	説明
N	10	顔文字の最大文字数
char_num	200	使用できる文字種の数
emb_dim	24	文字ベクトルの次元
hid_dim	24	潜在ベクトルの次元

ネットワーク全体図



encoder

decoder



前処理

顔文字をOne-hotベクトルに変換

例：(　´ω`)ノ
①文字ごとに分解（全角は空白＋半角.［　´］→［　］,［´］）
　(　´ω`)ノ　→　[(], [　], [´], [ω], [`], [)], [ノ]

②文字数をNに揃える
　[(], [　], [´], [ω], [`], [)], [ノ]
　→　[(], [　], [´], [ω], [`], [)], [ノ], [<PAD>], [<PAD>], [<PAD>]

前処理（続き）

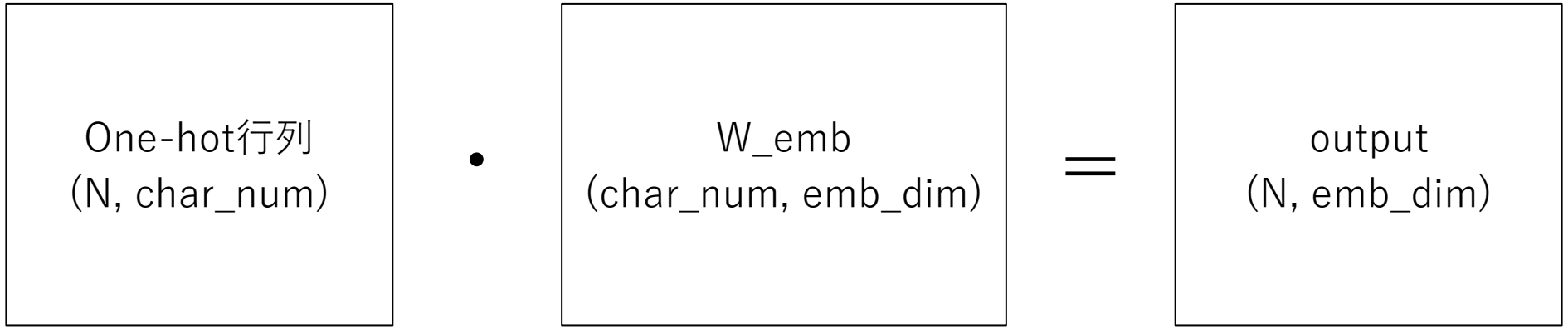
③使用できる文字のリスト(char_list)に照らし合わせて番号付け
　char_list = [<PAD>], [<UNK>], [(], [’], [⁀], [)], ・・・
　例えば，[(]は0から始めて2番目なので「2」

　[(], [　], [´], [ω], [`], [)], [ノ], [<PAD>], [<PAD>], [<PAD>]
　→　[2, 7, 26, 8, 52, 5, 33, 0, 0, 0]

④番号の要素が1のOne-hotベクトル化．最終的に(N, char_num)行列
　「2」→　[0, 0, 1, 0, 0, 0, 0, 0, 0 ･･･]
　「7」→　[0, 0, 0, 0, 0, 0, 0, 1, 0 ･･･]

emb_layer

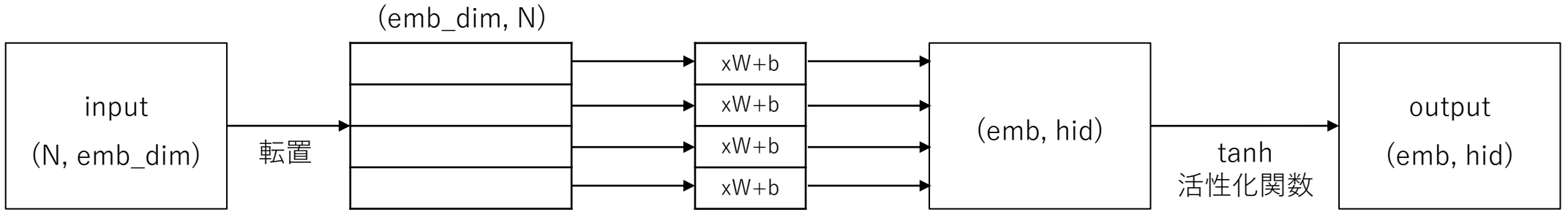
- One-hot行列と重みの行列積．バイアス無し



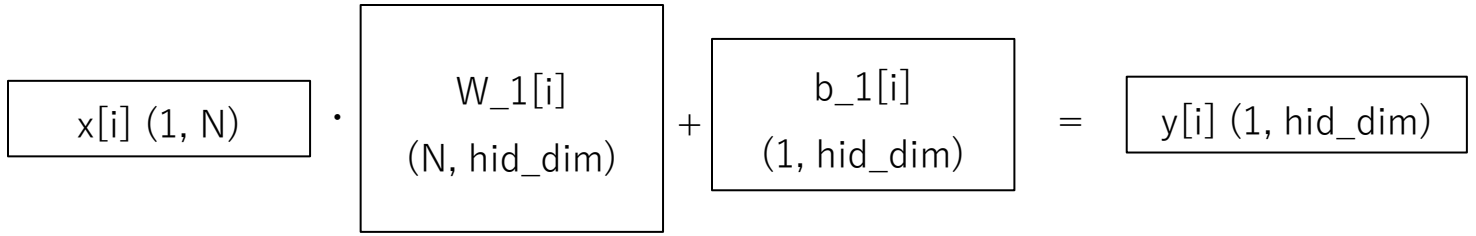
mix_layer1

- 入力を転置，行ベクトル毎に行列積+バイアス，合体，tanh

※一部「_dim」省略



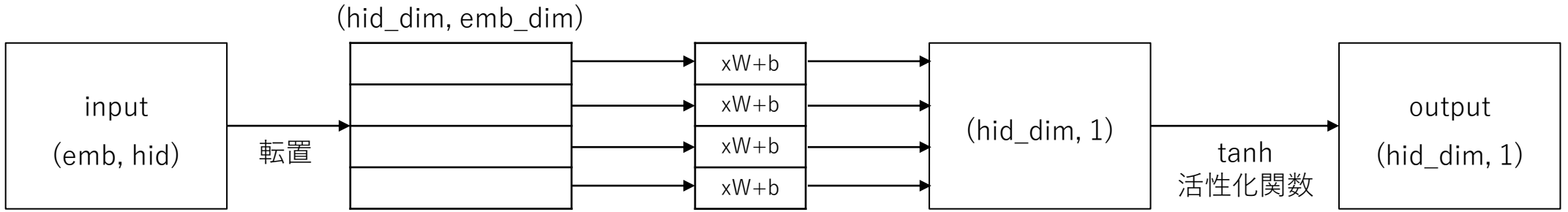
xW+bの処理：行ベクトル x ごとに異なる重みとバイアスを用いる（0 ≦ i < emb_dim）



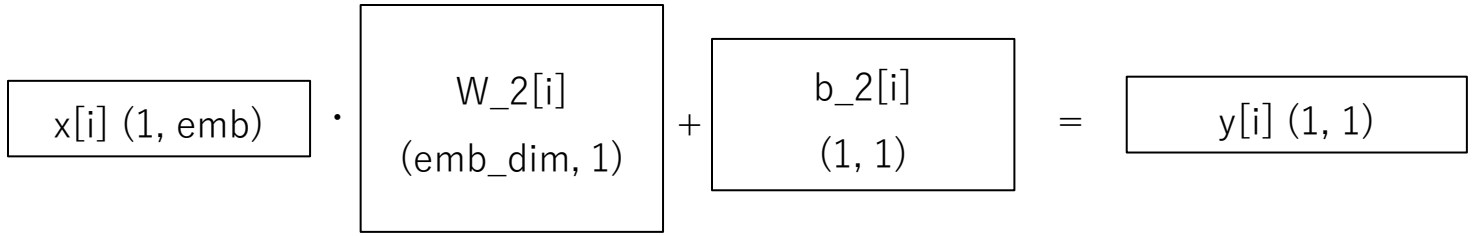
mix_layer2

- パラメータ以外mix_layer1と同様

※一部「_dim」省略



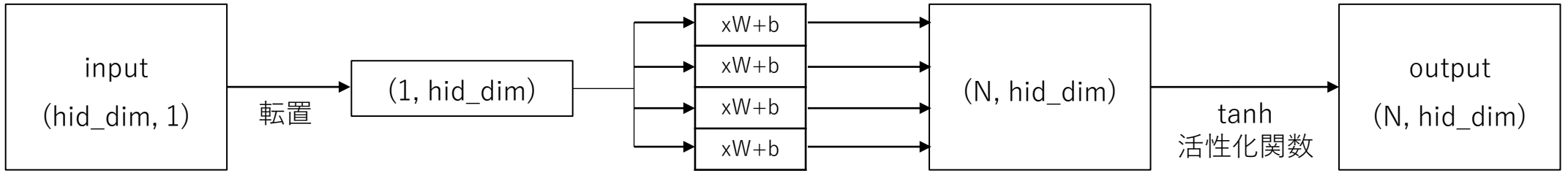
xW+bの処理：行ベクトル x ごとに異なる重みとバイアスを用いる（0 ≦ i < hid_dim）



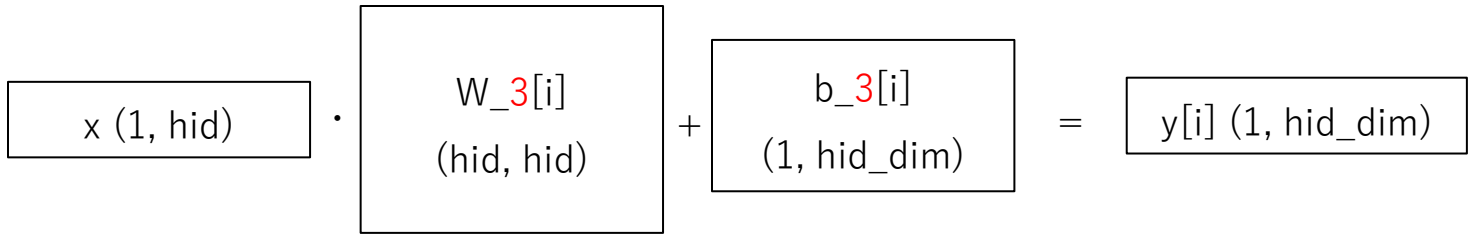
mix_layer3

- 同様だが，入力がベクトルなので同じ値を用いる

※一部「_dim」省略．重みは(バイアスも)「decoder_W_1」．ここではW_3と書いている

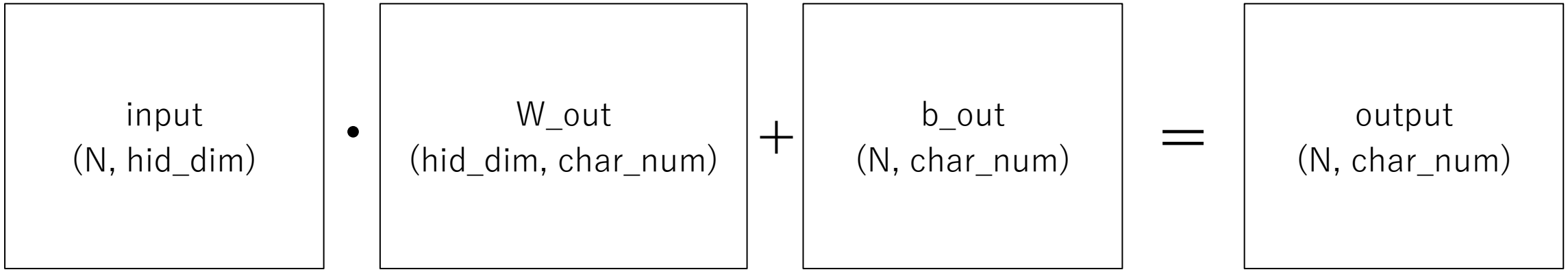


xW+bの処理：ベクトル x に対し，複数の重みとバイアスを用いて行列にする（0 ≦ i < N）



dense_layer

- 入力に行列積+バイアス．バイアスも2次元



comp_layer

- 行ベクトルの最大値の番号を取得

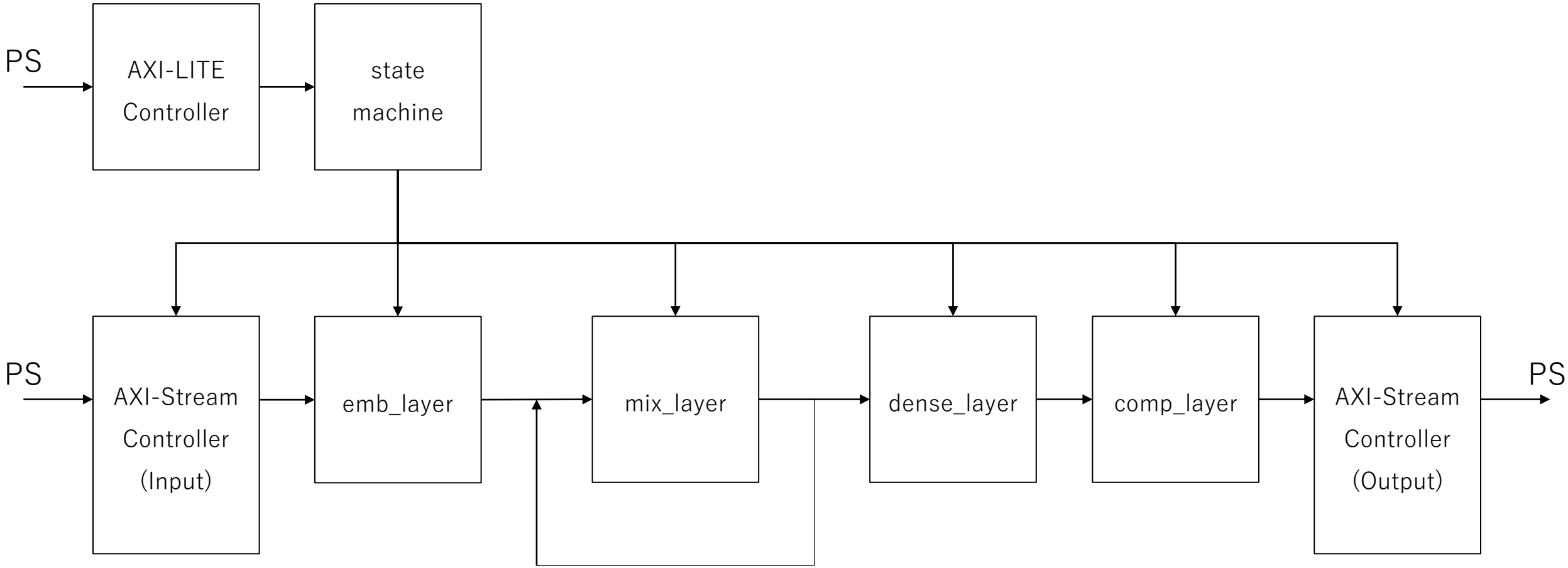
例えば，[0.1, 0.2, 5.0, -1.0, ・・・] で最大値が 5.0 だとすると，
5.0 は2番目だから「2」
char_list(前述)と比較して，文字を選択する.
「2」→「(」
これを，N回繰り返すと結果的に，

(N, char_num) → (N,) → (´Д`)ノ となる(全く同じのはなかなかできない)

目次

- ネットワーク概要
- 回路概要
- 決めごと・その他

回路全体図


PS
AXI-LITE Controller
state machine
PS
AXI-Stream Controller (Input)
emb_layer
mix_layer
dense_layer
comp_layer
AXI-Stream Controller (Output)
PS

パラメータの定義(FPGA)

パラメータ名	値（現時点）	説明
`N	10	顔文字の最大文字数
`CHAR_NUM	200	使用できる文字種の数
`EMB_DIM	24	文字ベクトルの次元
`HID_DIM	24	潜在ベクトルの次元
`I_LEN	6	数値の整数部のビット数
`F_LEN	10	数値の小数部のビット数
`N_LEN	16	数値のビット数．整数部＋小数部
`STATE_LEN	4	ステートを表す数値のビット数
`CHAR_LEN	8	char_listに照らし合わせた番号のビット数

AXI-LITE Controller（聖徳）

- AXI-LITEの通信用．回路全体の制御を行う．

属性	タイプ	データ長	名前	内容
interface			AXI_LITE	AXI-LITEのインターフェース色々
内部	reg	[31:0]	slv_reg0	rst_n, run, set の制御
	reg	[31:0]	slv_reg1	modeの制御
output	wire		clk	クロック信号
	wire		rst_n	リセット信号．負論理(0の時にリセット)
	wire		run	開始信号
	wire		set	ステートをセットする信号
	wire	[1:0]	mode	生成モード信号(後述)
	reg	[`STATE_LEN-1:0]	set_state	セットするステート

AXI-Stream Controller(input)（聖徳）

- AXI-Streamの通信用．内部FIFOでデータを受信.

属性	タイプ	データ長	名前	内容
interface			AXI_STREAM	AXI-STREAMのインターフェース色々
output	reg	[`N*`CHAR_LEN-1:0]	q	出力データ

AXI-Stream Controller(output)（聖徳）

- AXI-Streamの通信用．内部FIFOでデータを送信.

属性	タイプ	データ長	名前	内容
interface			AXI_STREAM	AXI-STREAMのインターフェース色々
input	wire	[`N*`CHAR_LEN-1:0]	d	入力データ

state machine（聖徳）

- ステートマシン.

属性	タイプ	データ長	名前	内容
input	wire		clk	クロック信号
	wire		rst_n	リセット信号．負論理(0の時にリセット)
	wire		run	開始信号
	wire		set	ステートをセットする信号
	wire	[`STATE_LEN-1:0]	d	セットするステート
output	reg	[`STATE_LEN-1:0]	q	現在のステート

state machine（聖徳）

- ステートの種類.

名前	値	内容
`IDLE	0000	アイドル．何もしない
`RECV	0001	FIFOからデータを取り出す
`EMB	0010	emb_layerの計算
`MIX1	0011	mix_layer1の計算
`MIX2	0100	mix_layer1の計算
`MIX3	0101	mix_layer1の計算
`DENS	0110	dense_layerの計算
`COMP	0111	comp_layerの計算
`SEND	1000	FIFOにデータを格納

emb_layer（土屋）

- emb_layerの計算を行う.
  計算は行列積だがOne-hotなので重みを取り出すだけで良い

属性	タイプ	データ長	名前	内容
input	wire		clk	クロック信号
	wire		rst_n	リセット信号．負論理(0の時にリセット)
	wire		run	開始信号
	wire	[`N*`CHAR_LEN-1:0]	d	入力データ
ouput	reg		valid	終了信号
	reg	[`N*`EMB_DIM*N_LEN-1:0]	q	出力データ

mix_layer（平原）

- mix_layerの計算を行う
  入出力の形状は層によって変わるが，最大値`HID_DIMに合わせた

属性	タイプ	データ長	名前	内容
input	wire		clk	クロック信号
	wire		rst_n	リセット信号．負論理(0の時にリセット)
	wire		run	開始信号
	wire	[`STATE_LEN-1:0]	state	現在のステート
	wire	[`HID_DIM*`HID_DIM*`N_LEN-1:0]	d	入力データ
ouput	reg		valid	終了信号
	reg	[`HID_DIM*`HID_DIM*`N_LEN-1:0]	q	出力データ

dense_layer（土屋）

- dense_layerの計算を行う
  行列積+バイアス.

属性	タイプ	データ長	名前	内容
input	wire		clk	クロック信号
	wire		rst_n	リセット信号．負論理(0の時にリセット)
	wire		run	開始信号
	wire	[`N*`HID_DIM*`N_LEN-1:0]	d	入力データ
ouput	reg		valid	終了信号
	reg	[`N*`CHAR_NUM*`N_LEN-1:0]	q	出力データ

comp_layer（聖徳）

- comp_layerの計算を行う
  行ベクトルの最大値の番号を取得し，整数値を出力

属性	タイプ	データ長	名前	内容
input	wire		clk	クロック信号
	wire		rst_n	リセット信号．負論理(0の時にリセット)
	wire		run	開始信号
	wire	[`N*`CHAR_NUM*`N_LEN-1:0]	d	入力データ
ouput	reg		valid	終了信号
	reg	[`N*`CHAR_LEN-1:0]	q	出力データ

modeについて

以下の4つのモードで動作させたい

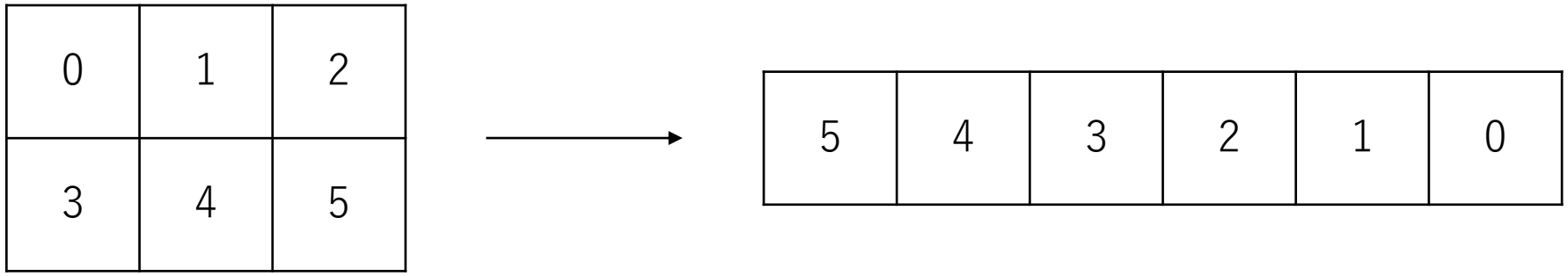
- 順伝播（まずはこれだけ実装）
  顔文字を入力，同じ顔文字を出力
- 類似生成
  顔文字を入力，潜在ベクトルに乱数を加え，入力と似ている顔文字を出力
- 新規生成
  入力は無し．潜在ベクトルを乱数で生成し，存在しない顔文字を出力
- 合体
  2つの顔文字を入力，潜在ベクトルを足し合わせ，新たな顔文字を出力

目次

- ネットワーク概要
- 回路概要
- 決めごと・その他

行列とデータ順

- 入出力時に行列は1次元になるので要素の順番を決める
  2×3行列→6次元ベクトル



フォルダ構造の統一

- フォルダ名や保存先を合わせましょう

名前	内容
data	データセットやパラメータファイル
include	Verilogの定数とかをインクルードして使う
notebooks	機械学習用プログラム
pdf	この資料とか
src	Verilogソースコード
tb	Verilogテストベンチ
tools	固定小数点変換とかのその他プログラム

各自でフォルダ作らずに
1回ブランチを作り直した方が
楽かもしれない

Verilogのコメント

- Verilogは「//」でコメントアウトできる.
- Vivadoのエディタは日本語がShift-JIS(多分)で，他のエディタは大抵UTF-8だから，日本語は文字化けする(2020.1時点)
- なので，コメントは英語で書きましょう
  （ローマ字でも良いよ）

コードを書いていく上で

- 各自専用のブランチにプッシュする
  ↓
- 聖徳が確認してメインブランチにマージする

って感じでやるからメインブランチにプッシュしないでね

ここまで読んだら，あとは手を動かすのみ！