Ver1.0

AIエンジニア・ブートキャンプ

～AIプログラミングを始める前の基礎固め～

<内容紹介 抜粋版>

シニアLSI設計アーキテクト
蛯原 均
Ebihara-Hitoshi@success-int.co.jp

- **自己紹介　蛯原 均（えびはら ひとし）**
  - 1986年3月 電気通信大学大学院 計算機科学専攻科 卒業
- **ソニー(株)　1986.04～2020.2　システムアーキテクト**
  - **研究所 ⇔ 製品設計**　17年
    - OpenGLアクセラレータ　**世界初の商品化**
    - 3次元ビデオ編集機（動画3Dリアルタイム変形）　**世界初**
    - SCE出向) PS2ゲーム開発環境 / GScube開発　**世界初HDゲーム機**
    - ソニエリ出向) auカメラ付き携帯 FW開発
  - **半導体設計**　17年
    - SystemC上流設計環境
      - モバイル向けイメージセンサ設計の標準手法に ⇒ **設計効率3倍達成！**
    - 大規模SoC基幹設計
      - Bravia、PS4、4K/8K業務用ビデオ機器
  - **ソニー人事部 中堅社員研修 技術講座企画&講師**（14年継続）
    - SoCアーキ設計、上流設計、RTL検証、低電力、AI実装 など
- **(株)Retail AI Lab　2020.03～2022.07**
  - **小売向けAI技術開発**（スマートショッピングカート、万引き防止センサ etc.）
    - カメラ検知(RaspPi4)、IoTデバイス(Arduino,ESP-32)
- **サクセスインターナショナル(株) 業務契約 2022.09～**









AIエンジニア・ブートキャンプ Agenda

- **1)AI(Neural Network)が動く仕組み**
  - 中でどんな演算が行われているのか動作原理を解説
- **2)自然言語処理**
  - 単語ベクトル、文章ベクトル、機械翻訳/RNN技術(LSTM/GRU)
  - Transformer/Attention、大規模言語モデルBERT～GPT
- **3)AI最新Topics**
  - 画像生成系AIの著作権問題 など
- **4)画像認識 CNN技術紹介**
  - ILSVRC(画像認識コンテスト) 歴代優勝者がもたらした革新とは？
- **5)AIハードウエア・アクセラレータ**
  - TPU/TensorCore/GPGPU 内部アーキテクチャ解説
- **6)AI技術活用のコツ**
  - 私の体験を元にAI導入に失敗した事例/原因など解説

本講座のコンセプト①

- **AI(ニューラルネットワーク)が動く仕組みは？**
  - 常識として知っておくべきAI技術の基本をわかりやすく解説
- **各AI技術が辿って来た歴史を解説**
  - 今の姿 = 過去技術の良い所を集めた集大成
    - 開発当時の課題認識/革新ポイントを把握しておく事が重要
- **23年版から「自然言語処理の進化」にFocus**
  - **ChatGPT公開** ⇒ 誰もが最新AI技術を使える様になった！

過去を正しく理解しておけばAI技術の未来が見える

AI技術の主役は数年で交代 ⇒ 変化への対応力が重要
新技術は何も無い所から突然出て来る訳ではない

本講座のコンセプト②

問題) AI関連の論文は数式による説明が大半

- 翻訳してみても革新ポイントが良くわからない ← 具体例の説明が無い
  - AI研究者は数学屋さんが多い

<一般公開>

カバー範囲を広げるため あえて詳細説明を省く論文も多い

AI開発者 オリジナル論文 (英文)

＋

<Net上の解説記事>

<日本語のAI解説サイトも増加中>
オリジナル論文を日本語訳しただけのサイトは技術の根幹部分の説明が漠然としていて良くわからない事が多い

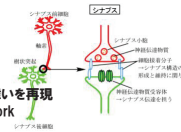
技術の本質を理解 ←この部分がとても大変、膨大な時間と手間がかかります

数式に頼らない、わかりやすい説明図を作成

「厳密さ」は諦め
「シンプルな説明」を優先しています

※本格的なAI学習は書籍/別講座でお願いします

Neural Network 基本構造



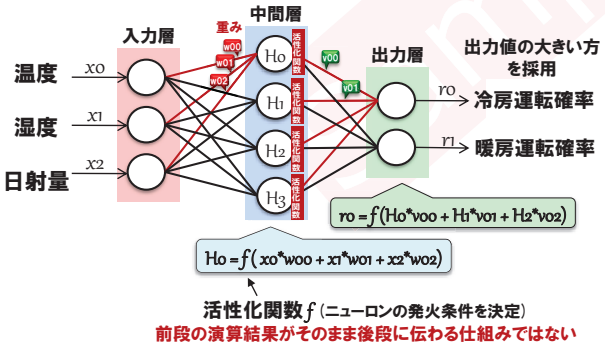
神経ニューロンの振る舞いを再現
→ Neural Network

神経ニューロンの仕組みを模擬

- 入力刺激の総和がクライテリア超え → ニューロン発火、後段に情報を伝える

Neural Networkは重み演算でニューロンの動きを再現

- 入力値に重みを掛けて合算 ⇒ 活性化関数経由で後段に伝達
  - 学習工程で重みを自動決定 → 重要な情報は大きな値に、不要な情報は小さな値に



活性化関数 f (ニューロンの発火条件を決定)
前段の演算結果がそのまま後段に伝わる仕組みではない

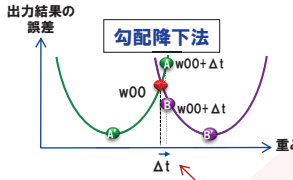
「暑い/寒い/ジメジメ/カラカラ/快適」
と感じたセンサ計測データを
数千～数万単位で集める
↓
Neural Netが人間の感覚を学習
「快適」と感じる領域まで自動運転

Neural Netの重み値を最適化する作業を
『教師データを用いた学習』
と呼ぶ

Neural Network 学習の仕組み

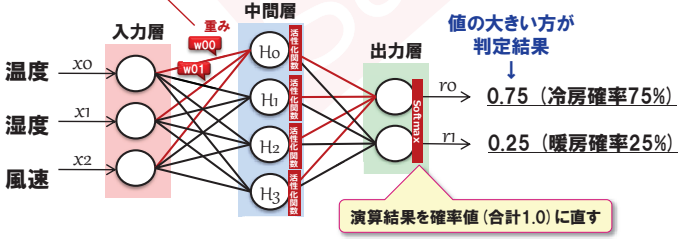
期待する出力結果との誤差が小さくなる様、"重み"調整を繰り返す

- 勾配降下法：重みにΔtを加算　誤差が増大→重みを小さく、誤差が減少→重みを大きく
- 誤差逆伝播法：出力層から入力層方向に重みの最適化を進めて行く

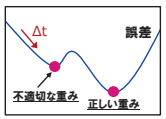


w00+Δtの結果
A：誤差拡大
　w00を小さくして行く
B：誤差縮小
　w00を大きくして行く

膨大な学習データ（数万）を用意して
演算パラメータ（数千）の
最適化を繰り返す
↓
演算量が膨大！
（HWアクセラレータを使わないと辛い）



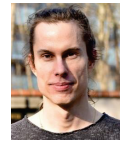
重み探索を失敗するケース



Δt加算後、誤差がそれ以上縮小せず、逆に増えた場合は増える直前の重みが適切な値

単語をベクトル変換 Word2vec

2013年「単語ベクトルを発見した報告」



- Tomas Mikolov氏 2つの論文を発表
  - 周辺単語から中心単語を予測するNeural Net環境を作成
    - その中に『**単語ベクトル**』として使えるデータが存在する事に気付いた
      »単語をベクトル変換する研究の成果発表ではない ← **わかりにくい**

Word2vec技術の中身　最低1種類ずつ入っていればw2vと呼ばれる

- 単語ベクトルを生成できるNN環境 2種類

CBOW or Skip-gram ← 単語を予測する環境の例、2つ存在

- 計算速度改善のため実装の工夫 2種類

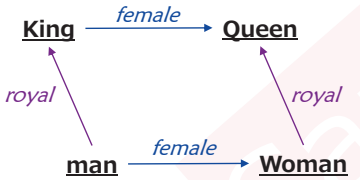
Hierarchical SoftMax or Negative sampling ← 何も工夫しないと重過ぎて動かない

Python gensimライブラリに含まれるのはコチラ

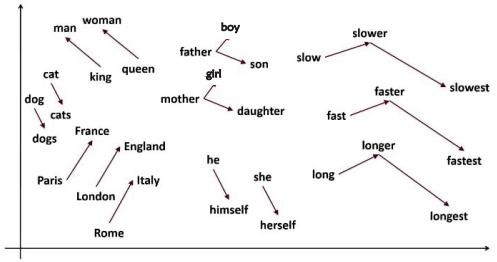
◇Word2Vecの論文
1) Efficient estimation of word representation in vector space 2013 https://arxiv.org/abs/1301.3781
2) Distributed Representation of words and phases and their composition 2013 https://arxiv.org/abs/1310.4546

word2vecで作られた単語ベクトルには面白い性質が存在

- **共起情報(近くにある)だけで作られたベクトルなのに、単語の意味が反映されている**
  - 同じ概念を持つ単語 ⇒ 周辺に出て来る単語に規則性
- **ベクトル差分が単語の意味の違いを表す**
  - 「王族の単語ペア(王様/女王)」と「庶民の単語ペア(男/女)」を比較すると、同じ意味の差は同じベクトル差分になる



「男女の違い」という概念
king − queen = man − woman
「高貴な人」という概念
king − man = queen − woman
たとえば、
princess = prince − man + woman



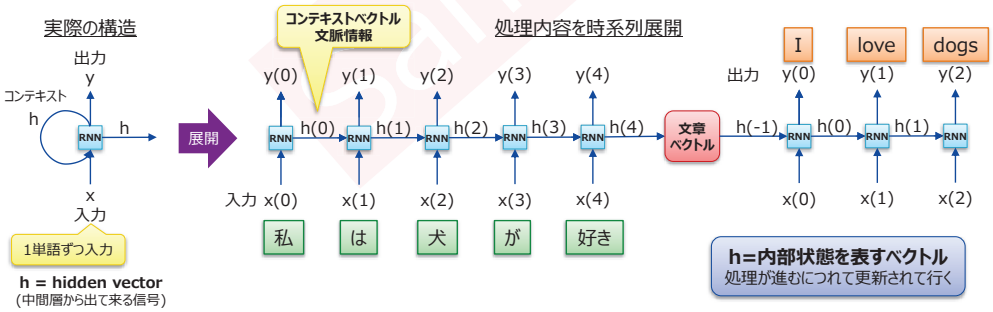
slow→slower→slowest の関係を理解していれば、
fasterはfastのバリエーション、
longestもlongのバリエーション、だと推論できる

faster = fast + (slower − slow)
longest = long + (slowest − slow)

文章理解で重要な事

$(y(t), h(t)) = f(x(t), y(t-1), h(t-1))$

- **文章理解では文脈追跡が必須**
  - 文章後半に出て来る “それ”、”彼”の様な代名詞が何を指すのか？
- **RNN(Recurrent Neural Network)が主流に**
  - 中間層出力が中間層入力に接続される再帰型Neural Network
    - 「前回の状態」が「今回の状態」を決める手掛かりになる



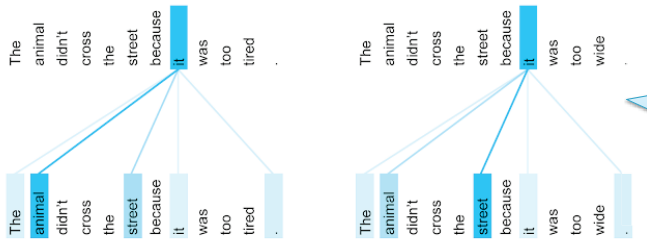
『Transformer』の革新ポイント

- **2017年 Google論文「Attention is all you need」**
  - 正確な機械翻訳を目的とするモデル構造の説明
    - 利点: 計算を並列化/高速化、長い文章の文脈保持、より正確な変換
    - https://arxiv.org/abs/1706.03762
- **従来技術の問題**
  - RNN/seq2seq: 大域的特徴/ニュアンスを捉えにくい、高速化が困難
- **改善ポイント**
  - **Multi-head Attention + Position-Wise feed forward Network**
    - 単語間の照応関係(アライメント)を重視
    - Query-Key-Valueモデル
  - **AttentionベースのEncoder-Decoder型モデル**
    - RNNブロック無し ⇒ 並列演算に有利、文脈を忘れる問題から脱却
- **適用例**
  - BERT、GPT-n、ViT(画像分類)、DALL-E(画像生成)、AI将棋(東大) etc.

Transformer Attention機構の役割

- **ひとつの文章に同時に現れる単語の組み合わせを学習すると**
  - **文脈が理解出来る様になる：「it」が何を指すかは文脈次第**
  - The animal didn’t cross the street because it was too tired.
    ⇒「it」は「animal」
  - The animal didn’t cross the street because it was too wide.
    ⇒「it」は「street」

↑ 文法的な解釈では
何を指すかわからない



animalとtiredは一緒に出現する
(streetとtiredは一緒に出て来ない)

streetとwideは一緒に出現する
(animalとwideは一緒に出て来ない)

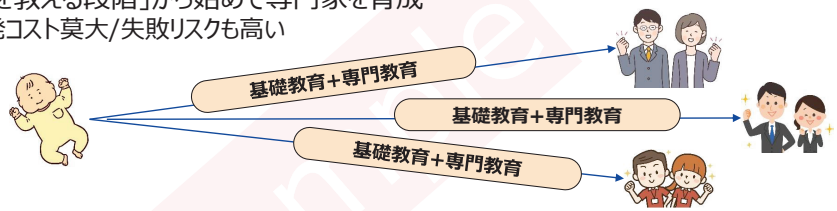
文章中の名詞は
Animal/Street
の２つ
↑
代名詞itが
指すのはどっち？

※1epoch＝10億単語を含む文章データ

事前学習型大規模言語モデル(LLM) 登場

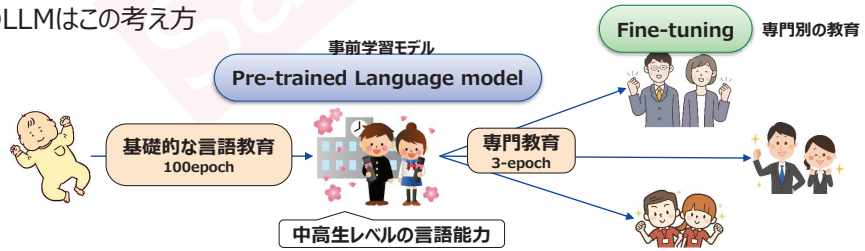
従来のAI開発

- 「言葉を教える段階」から始めて専門家を育成
  - 開発コスト莫大/失敗リスクも高い



言語習得済みLLM ⇒ 追加学習で専門家に育成

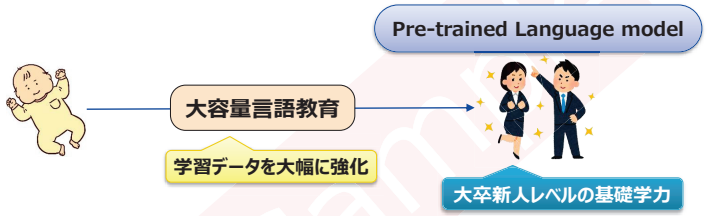
- 初期のLLMはこの考え方



LLMは大規模化する程賢くなると判明

採算無視の巨大化競争が勃発

- GPT-4(100兆パラメータ?)で「大卒新人レベル」に到達



神経ニューロンが多い程
知能は高くなる

巨大化のデメリット

- **追加学習が出来ない** (サーバコストだけで数千万円)
  - Fine-Tuningは無理 → そのまま使う
- **運用テクニックで使いこなすしかない**
  - Few Shot学習
  - Prompt Engineering (プロンプト・エンジニアリング)

大規模言語モデル ビジネス状況

2023年 投資資金を回収すべくビジネス化に舵を切った

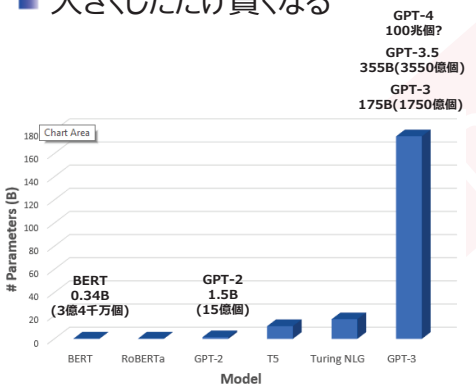
- 数千億円規模のお金を稼げる用途はまだ見つからず
  - **しかし、次の開発投資を停止すればその時点で敗退確定**

社名	UI	LLM 大規模言語モデル	LLM運用形態
Google	BARD (chat形式)	PaLM2 (Lambdaは失敗作)	暫定無償公開中
	Google Cloud アプリ	PaLM2	顧客アプリの中から PaLM2機能をCall
OpenAI	ChatGPT (chat形式)	GPT-3/3.5 GPT-4	GPT-3系は無償 (1億円/日) GPT-4は有償
Microsoft	Bing (chat形式)	Prometheus (GPT-4を包含)	検索エンジンに組み込み Webサイト内容を要約
	Azure Cloud Officeアプリ	GPT-4	GPT-4機能を直接Call

大規模言語モデルの巨大化

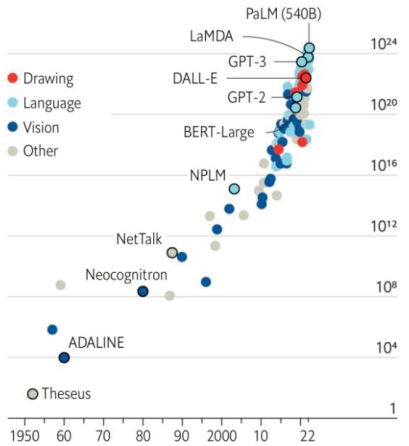
規模限界はまだ先

- 大きくしただけ賢くなる



LLM:パラメータ数の推移

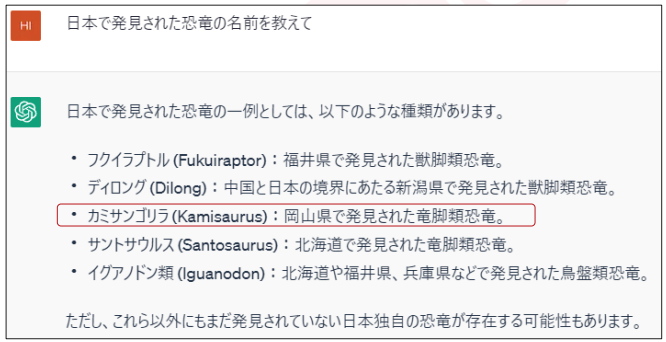
GPT：OpenAI社
PaLM/LaMDA/BERT：Google社



GPT-4は断トツに大きい

ChatGPTで起きる「不都合な現象」

- **Hallucination(幻覚) が発生する事がある**
  - 嘘の答えをそれらしく回答する
    - 正解の中に正々堂々と混ざっているため見過ごす危険大



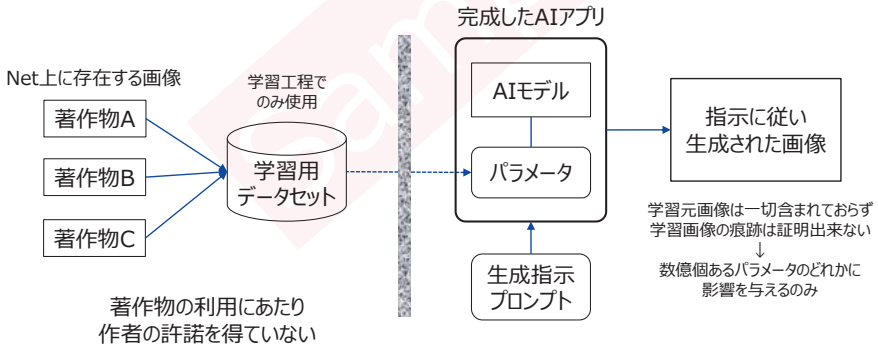
指示文にスムーズにつながる文章パターンを作っているだけ
回答内容が正しいかどうかはそもそも考慮外

大規模言語モデルが抱える本質的な限界

- **文脈は理解出来るが、文章の意味は理解出来ない**
  - 単語出現パターンを学習 → 文脈理解能力を獲得
    - 与えられた文章を別の表現/他言語に変換する事が可能に
      - 日本語、外国語、プログラミング言語など何でもOK
- **典型的な文章パターンを生成する道具に過ぎない**
  - 次単語を予測して順に並べているだけ → LLM出力
- **次単語予測のみで高度な対話が出来る理由は不明**
  - 人格を備えた発言に見える
  - 誰も教えていないのに四則演算が出来る様になる　なぜ？

AI学習と著作権の問題

- **学習元の画像は学習に使われるだけで残らない**
  - Neural Networkのパラメータに反映されて終了
    - 画像情報としてはどこにも残っていない
      - パラメータは数億個あり、学習作業を通して値が少しずつ変化して行く



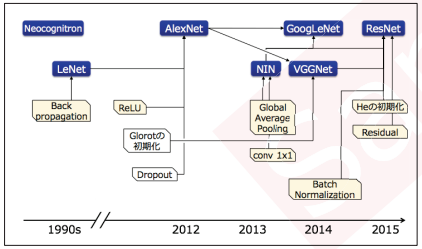
AI生成物に関する法廷闘争

- **画像生成AIで作ったコミック作品を販売**
  - 著作権は誰に帰属？ → 米著作権局の現時点の見解
    - 設定や物語に関する著作権は作者に帰属
    - しかし、画像は人間の作者による作品ではない → 著作権は発生しない
- **AI生成物の著作権が認められれば数十億ドルのお金が動く**
  - AIを使えば誰でも著作権で商売出来る時代が来るかも…
- **画家) 画像生成AIは著作権のある画像を含め学習している**
  - AI生成物に著作権を与える ＝ 窃盗の合法化では？
- **人間が下書きを書き、それをAIが完成させるスタイルでは？**
  - 著作権が生じるのではないか？
- **AIに意識があるなら、AIが作者と言えるのではないか？**
  - AIに人格を認めるか否か？

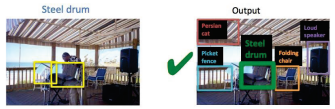
GenerativeAIには『産業革命』並みのインパクト ⇒ ビジネス構造を変えてしまうかも

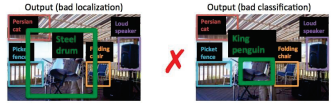
CNN技術に革新 (2012年～ 第3次AIブーム到来)

- **2012年 画像認識コンテストで『AlexNet』圧勝**
  - Neural Net技術がついに旧来手法に勝った
    - Neural Netはスゴイ！
  - ILSVRC ImageNet Large Scale Visual Recognition Challenge
    - 2010年スタートの大規模画像認識競技会

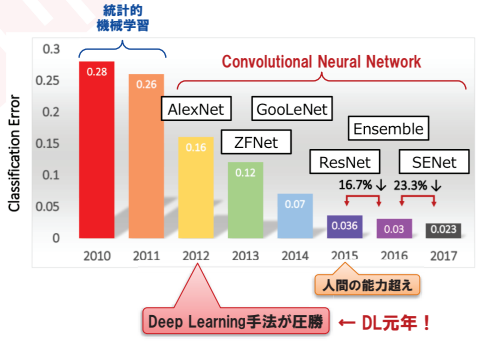


この画像認識コンテストで認識能力が劇的に向上！



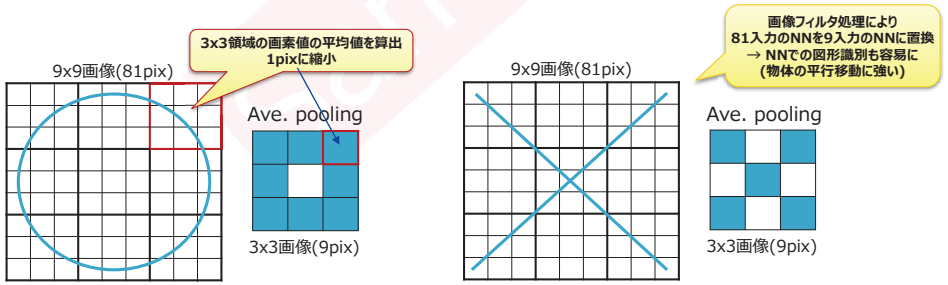


幾つ正しくラベリング出来たかを競う



CNN(Convolutional Neural Network)の仕組み

- **画像をNN全結合層で直接処理するのは効率が悪い**
  - 1次元データ列としてNNに接続 → その際、2次元情報は失われる
- **2次元フィルター演算+pooling処理で画サイズ縮小**
  → 二次元情報を極力残した縮小画像をNeural Netに接続
  - NNが対象物が何かを判定



Average Pooling処理を用いた「○」「×」識別

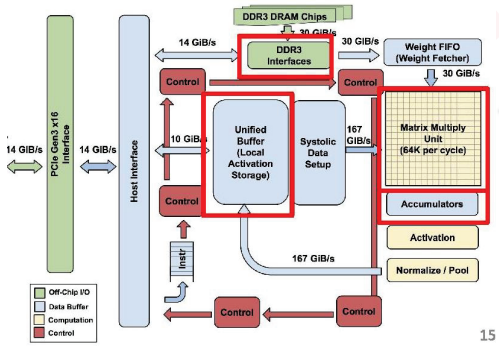
Neural Network専用アクセラレータ登場の背景

- **NN演算は重い ⇒ 加速手段が無いと厳しい**
  - CPUのみだと数日 → GPUを使えば数H
    - 仮想通貨で高性能なGPUが入手し易くなったが、まだ高い！
- **大量データ/リアルタイム処理ニーズの高まり**
  - Google社: 音声認識利用者が増加 → 応答レスポンス改善が急務に
    - GPUは高価、大量導入は無理 ⇒ 専用プロセッサ開発を決断
- **専用HWを作るメリット ⇒ 低電力と高性能の両立**
  - GPU処理は電力効率が悪い (演算時のデータ移動が多いため)
  - 専用のHWアクセラレータなら電力効率を高められる
    - モバイル機器にも搭載可能

AIアクセラレータと呼ばれるもの = 中身はMatrix演算器
(30年前に考案されたシストリックアレイ構造がJust fit！)

Google社がTPU自社開発に踏み切った背景

- **当初 NN演算に NVIDIA社GPU Tesla P100 を使用** 21.2TFLOPS/FP16
  - 音声検索増加でサーバ負荷増 → 処理量2倍 調達コストが許容不能レベルに
- **2014年 独自LSI開発に踏み切る**
  - シストリックアレイ構造を持つTPU(Tensor Processing Unit)を開発
    - ムーアの法則が終了 → このまま待っていてもCPU/GPUの性能飛躍は望めず

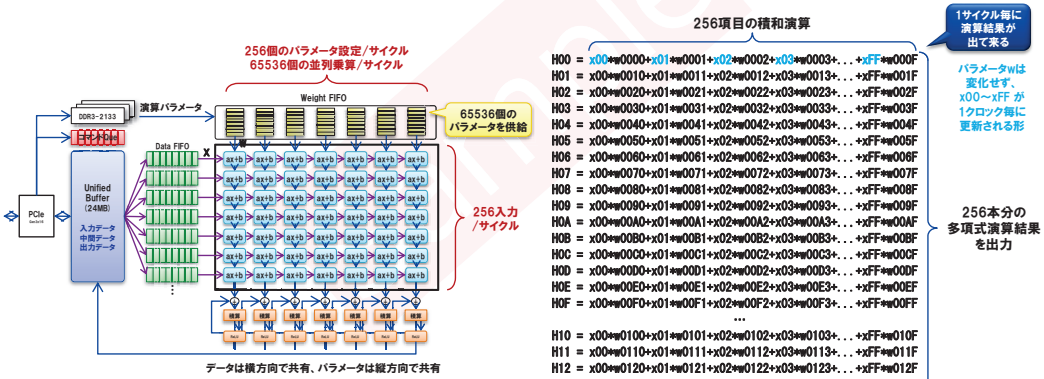


<TPUの優位性>
シストリックアレイ構造は
制御用CPUが不要
↓
CPUは可変性がある分
冗長性が高い部品
サイズが大きく電力効率も悪い

※Systolic=拍動

Google TPUv1 アーキテクチャ

- **256入力x256出力 シストリックアレイ演算器**
  - 8bit整数(入力)x8bit整数(パラメータ)乗算器(65536個)の演算/サイクル



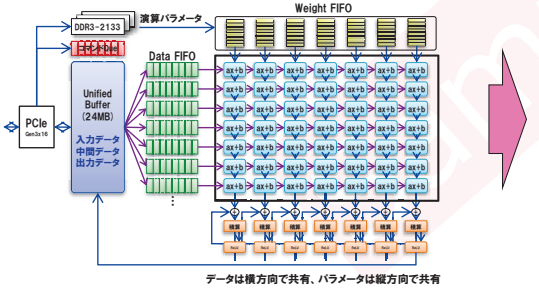
↑この演算結果が1クロック周期で出て来る

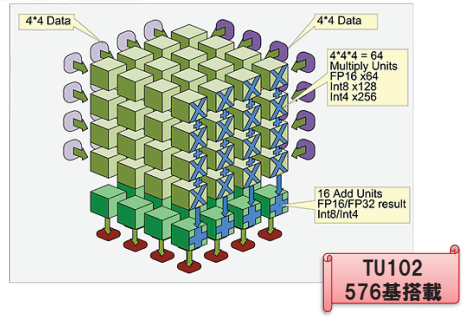
Google TPU と nVIDIA TensorCore アーキの違い

- **Google TPU (2次元)**
  - **256x256 Matrix演算器**
    - **65536個のMAC演算/サイクル**

1000倍違う

- **nVIDIA TensorCore（3次元）**
  - **4x4x4 Matrix演算器**
    - **64個のMAC演算/サイクル**

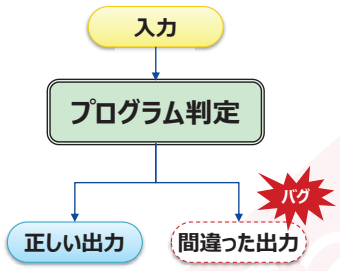




TPUは超大規模回路 ⇒ 巨大なMatrix演算を一度で処理
TensorCoreは小さいMatrix演算器を大量に載せて並列動作させるスタイル

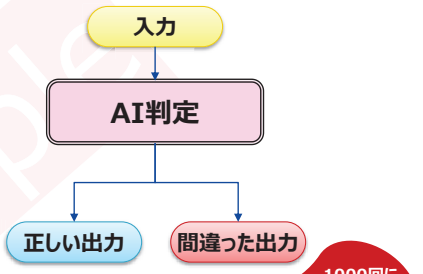
AI活用) AIは間違える前提で使え！

- **「条件式」で判定**



デバッグ済の条件式は
常に正しい答えを出す
↓
誤判定があればバグ⇒必ず直す

- **「NeuralNet」で判定**



間違った判定はゼロにはならない
↓
将来も100%になる事は無い

たまに間違えるプログラムを安全に使う工夫が必要 ← 多くの人は未経験

間違った出力が出ない前提でシステムを作り、後付けで対策しようとして失敗するチームが多い

<End>