社会実装とデータサイエンス4
（深層学習向けプロセッサ）

牧野淳一郎
神戸大学理学研究科惑星学専攻

話の構成

- ニューラルネットワークの (すごく短い) 歴史
- これまでの計算機の進歩と現状の困難
- 「ポストムーア」
- ニューラルネットワークの進歩と計算機の進歩
- まとめ

牧野はどういう人か？

- 神戸大学理学研究科惑星学科専攻教授
- どういうわけで「データサイエンス・人工知能のための高性能プロセッサについて概説」するのか？

牧野はどういう人か？

- 元々、銀河系や星団の、コンピュータシミュレーションを使った研究をしていた (今もしている)
- 「シミュレーション専用計算機」の開発を 1990 年くらいから始める。
- 2004 年から、汎用並列計算機開発にもかかわる。
- 2011 年から、理研の「ポスト京」プロジェクトにも。
- 2016 年から、 AI ベンチャーの Preferred Networks と共同で深層学習向けプロセッサを開発。

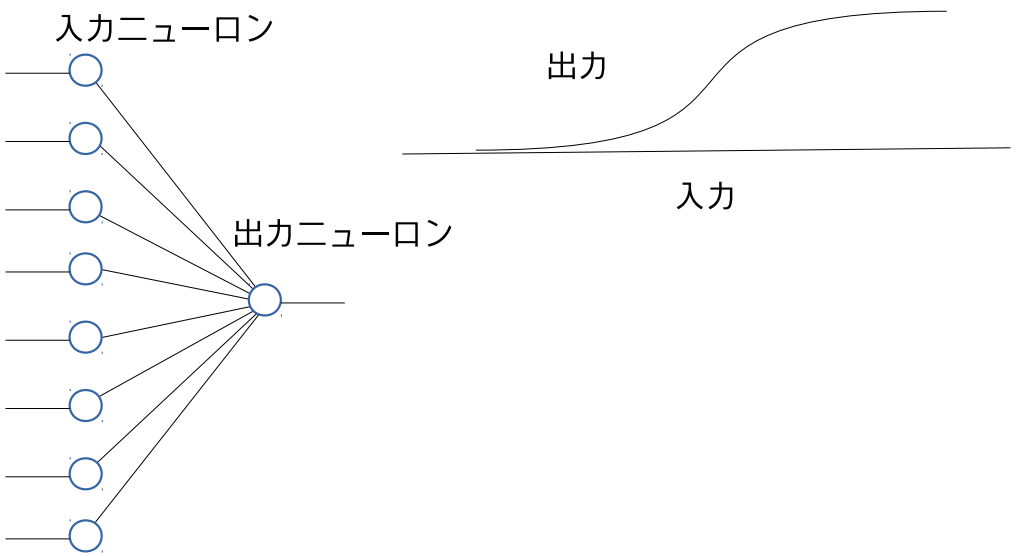
PFN 紹介ビデオ

ニューラルネットワークの (すごく短い) 歴史

「三度目の正直」?

- 1950 年代: パーセプトロン
- 1980 年代: 誤差伝搬学習
- 2010 年代 (今): 多層ネットワーク (「深層学習」)

パーセプトロン



パーセプトロン

- 入力の多数のニューロンと出力ニューロン
- 1つのニューロンは、入力を $[0,1]$ の範囲に変換する「関数」
- 複数の入力があるニューロンは、それらに係数をかけて合計してから関数で変換
- 「学習」ができる:「正解」と出力の差から係数を変える

パーセプトロン

- 1950-60 年代に色々研究された。
- 表現できる機能に限界があることが指摘されて、人工知能研究としては下火に。

パーセプトロンの限界

例えば単純な2項論理演算のうち、排他的論理和 (XOR) は表現できない。

A	B	積
T	T	T
T	F	F
F	T	F
F	F	F

A	B	和
T	T	T
T	F	T
F	T	T
F	F	F

A	B	XOR
T	T	F
T	F	T
F	T	T
F	F	F

何故限界があるか

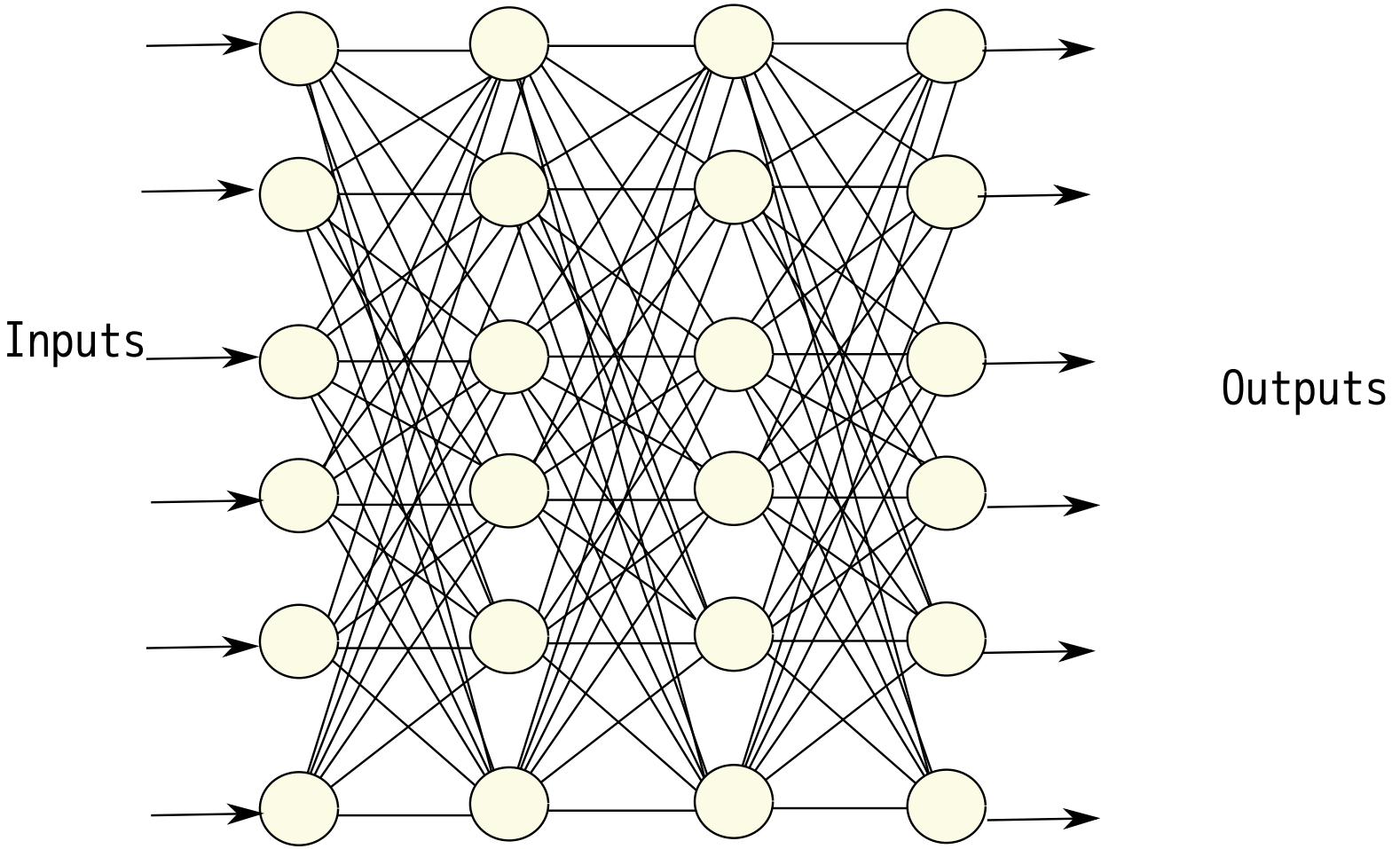
(厳密な証明は今日は省略)

- 入力ニューロンの出力を (係数はあるけど) 単に加算。
- 入力のベクトル空間 (2 入力なら平面) で、出力ニューロンの入力が一定になる (具体的には、出力が切り替わる) 値はある超平面 (2 次元なら直線) 上にある。
- 超平面 (2 次元なら直線) で分けられるものしか表現できない

限界を超える方法

- 多段ニューラルネットワークにすればいい、というのは論理的には明らか。
- 例えば論理演算であれば、任意の数の入力の任意の形の論理式は、全て、3層のニューラルネットワークで表現できる。
- これは命題論理でいう積和標準形にすればいいため
- 極めて形式的というか理屈としては、論理演算さえできれば数値計算もできるので、3層のニューラルネットワークは「万能」。なんでも計算できるはず

多層ネットワーク



ではなぜ1960年代に下火になったか?

- 多層ネットワークに学習させる方法が分からなかった
- そもそも計算機の能力的に多層ネットワークは無理だった

2層ネットワーク: 入力ニューロンが n 個だと、必要な記憶容量も計算量も n 程度。
3層ネットワーク: 入力層、中間層がそれぞれが n 個だと、必要な記憶容量も計算量も n^2 程度。
ニューロンが1000個あると、3層ネットワークは2層ネットワークの1000倍大変。

1980年代における復活

- 理論の進歩: 3層ネットワークに学習させる方法が発見された
- 計算機の進歩: だいぶ速くなったので小規模な3層ネットワークならなんとか。

学習させる方法

「誤差逆伝搬学習」
難しそうな名前がついているが、数学的な考え方は単純。

ある層の係数を、ニューラルネットの出力と「正解」の差が一番大きく減る「方向」に少しだけ変化させる。

(「最急降下法」と呼ばれる)

もうちょっとだけ説明

- 多層ニューラルネットのある層の係数それぞれについて、
- 今与えている入力は固定して
- 係数をちょっと変えたら出力がどう変わるかを計算
- 係数の変え方をなるべく小さくして、出力が正解に大きく近付くようにする

数学的には、、、、係数をベクトル $\mathbf{w} = (w_1, w_2, ...w_n)$, 正解を y_t、出力を y と書く時、 y_t が y より大きいなら

$$\left(\frac{\partial y}{\partial w_1}, \frac{\partial y}{\partial w_2}, \cdots \frac{\partial y}{\partial w_n}\right)$$

の方向、逆ならその逆方向に「少し」動かす。
(学習させたい入力が沢山ある時どうするかとか出力が沢山あるとどうか一杯色々な話があるが今日は省略)

「誤差逆伝搬」

- $\frac{\partial y}{\partial w_1}$ 等を求める手順が、出力層で計算した誤差をニューラルネットを逆に辿っていって計算する形になるのでこういう名前。
- 原理的には、3層でなくてもっと多層でもなんでもこの方法で学習させられる。
- 「万能」

ではなぜ80年代のニューラルネットブームも下火に？

- あまり大きなネットワークはやはり計算機のメモリ容量や速度的に無理だった。入力数千ニューロンがやっと。画像だと 30×30 くらい。
- そうするとあんまり色々なことはできない。画像認識等も、人間が色々考えて作った方法に勝つまでにはなかなかいかなかった。

2010年代

- 本質的に新しいアルゴリズムとかができたわけではない
- ものすごく大きなネットワークを、大量の入力データを使って力任せに学習させることが可能になった。
- 入力 300×300、 150層とか
- 細かいテクニックは一杯生まれた。ニューロンの関数の単純化、畳み込みネットワーク、バッチノーマリゼーション、その他色々

その結果

これまでなかなかできないかった色々なことができるようになってきた

- 細かい画像認識: 例えば画像のこの領域は「犬」とか「人」といったラベルをつける
- 音声認識
- 自動翻訳
- 囲碁等のゲーム

特に画像認識は自動運転のキーとなる技術の一つ

ニューラルネットワークのこれまでの進歩

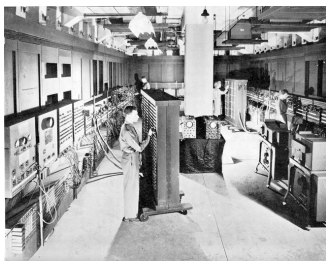
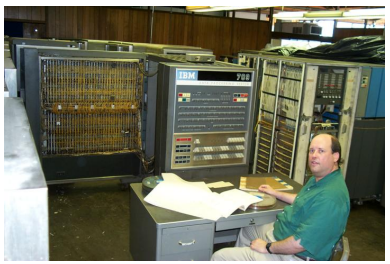
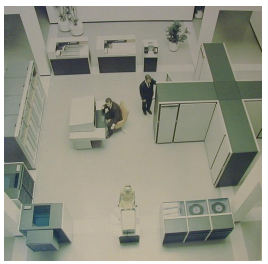
- 大体 30 年おきに「大きな発展」
- これは、計算機の速度でいうと 100 万倍
- 計算機は、1950 年代から今まで大体 10 年で 100 倍速くなってきた

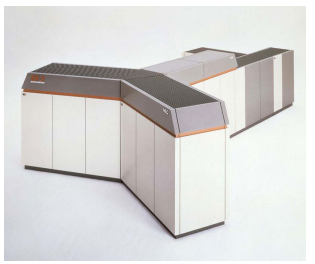
というわけで、次の話題:計算機の進歩のこれまでとこれから

これまでの計算機の進歩

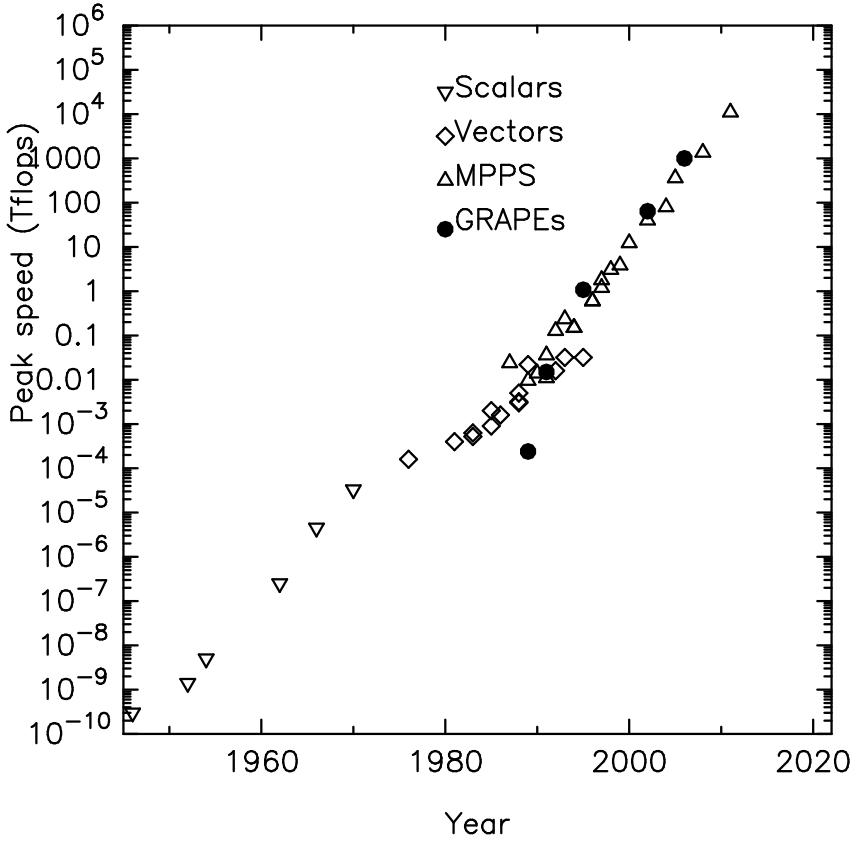
1940年代から現在までの70年間、ほぼ10年で100倍。何故そのような指数関数的進歩を長期に続けたのか?(今後はどうか?)
基本的な理由:

- 使うスイッチ素子が高速になった
- 使うスイッチ素子が小型、低消費電力になって、沢山使えるようになった
- 使うスイッチ素子が安くなって、沢山使えるようになった(スパコンの物理的大きさは70年代が最小。そこまで段々小さくなって、そこからまた大きくなった)

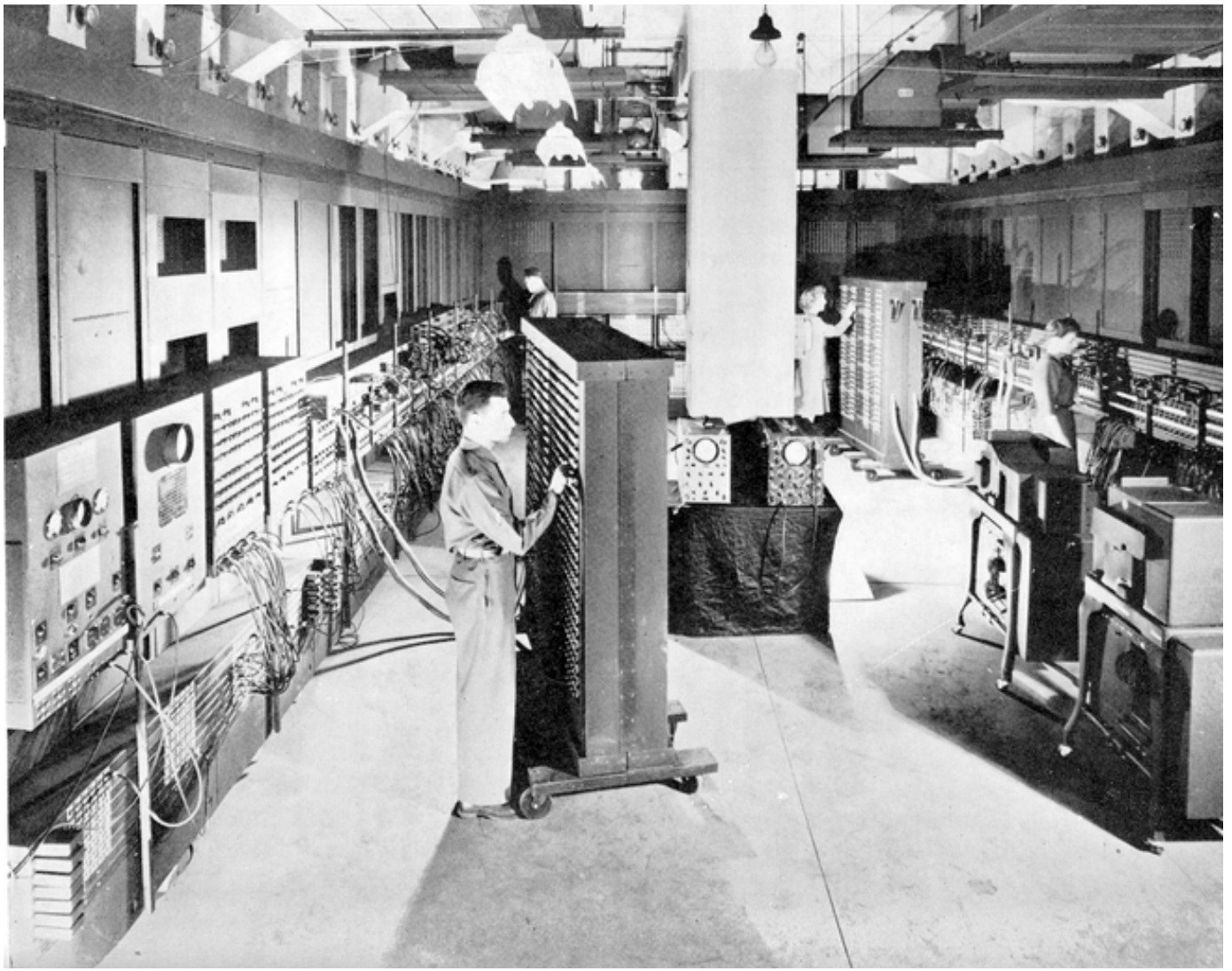



スパコンとは何か



- 単純には、「その時代で最高速クラスの計算機」
- 70 年間で 14 桁速くなった＝ほぼ 10 年で 100 倍
- 何故これほど進歩したか？は「スパコンとは何か」そのもの

1940年代の「スパコン」



ENIAC

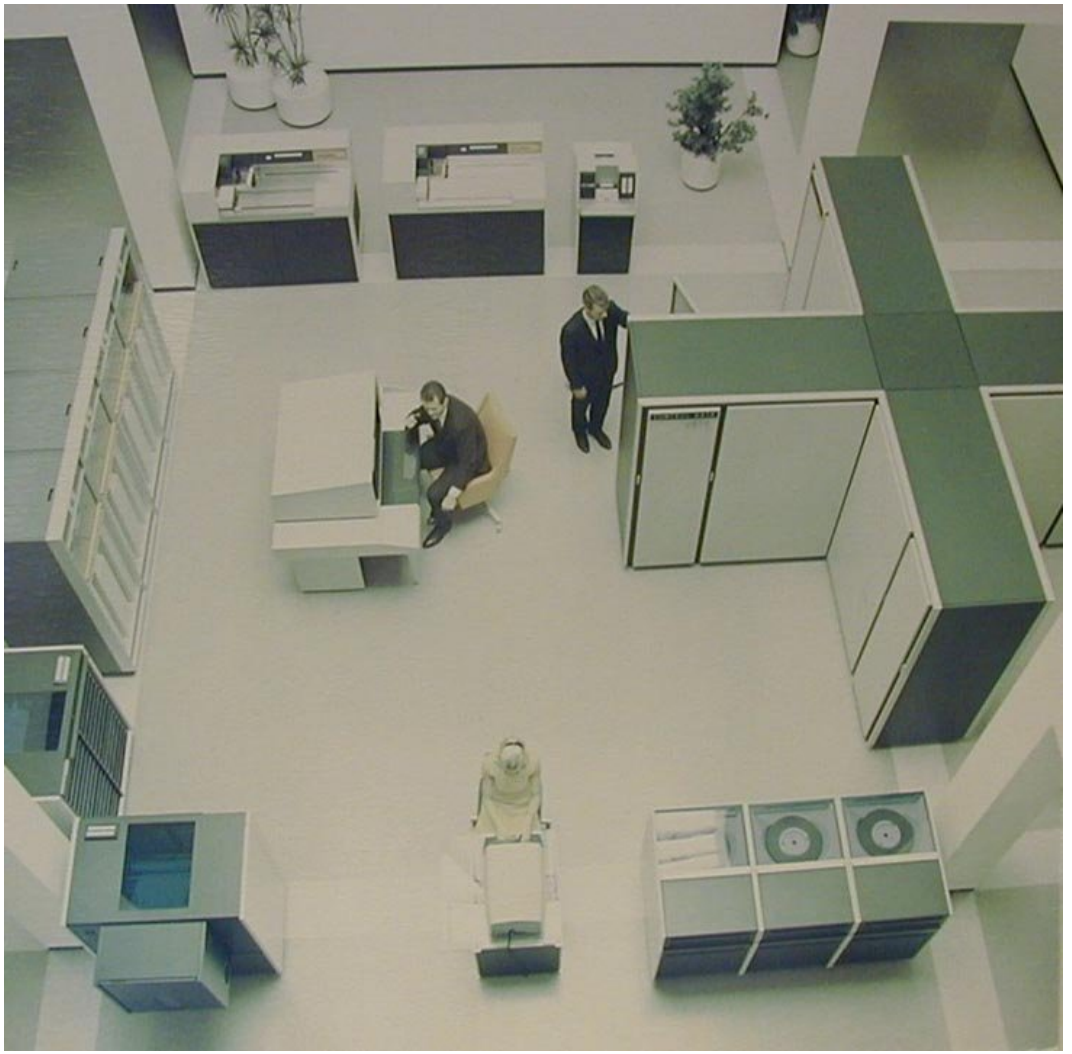
ENIAC

- 第二次世界大戦中に、アメリカ陸軍が弾道計算のために開発。完成は 1946 年
- 18,000 本の真空管、消費電力 150KW、10 進法 10 桁の数が基本
- 「プログラム」はスイッチ、ケーブルの変更で行う
- 加減算が 1 秒 5000 回、乗算が 400 回くらい (らしい)

1950年代の「スパコン」

IBM 709 — IBM 最後の真空管計算機

1960年代の「スパコン」



CDC 6600 1Mflops

CDC6600

- 1964 年から出荷。シリコントランジスタを利用。
- 10MHz クロック、1Mflops 程度の性能
- 近代的計算機アーキテクチャの原型。
  - ロード・ストアアーキテクチャ
  - 複数の演算器の「アウト・オブ・オーダー」実行
- シーモア・クレイが設計

1970年代の「スパコン」

CRI Cray-1 160Mflops

Cray-1

- 1976 年から出荷。IC (ECL 素子による小規模 IC) を使用
- 80MHz クロック、160Mflops
- 最初に成功したベクトル計算機
  - ベクトルレジスタ
  - 半導体メモリ
- シーモア・クレイが設計

ベクトル計算機って？

- 「スカラー計算機」と「ベクトル計算機」と分ける
- 「ベクトル処理」をするかどうか
- スカラー計算機 (あるいは普通の計算機の基本命令): 命令 1 つで演算 1 つ。例えば掛け算 1 つとか。
- ベクトル計算機: 1 命令で複数の演算を処理。ベクトルの、要素同士の四則演算が基本。それだけでは足りないので色々追加機能がある。
- 必ずしも並列処理するわけではない。**Cray-1** では「パイプライン処理」。長さ n のベクトル命令の実行に 定数 $+ n$ クロックかかる。

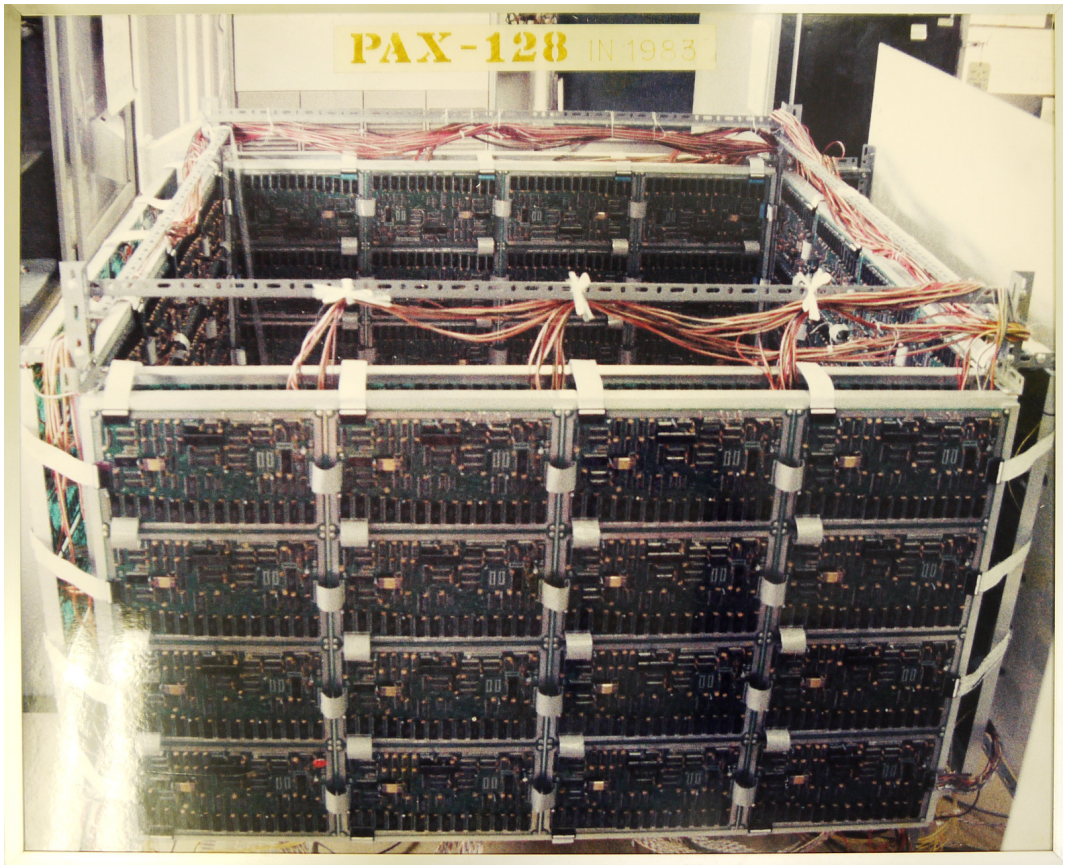
1980年代の「スパコン」

NEC SX-2 1.3Gflops

NEC SX-2

- 1985 年から出荷。CML 論理素子による LSI を使用。1000 ゲート程度 (Cray-1 の IC の 100 倍程度)
- 6ns クロック、掛け算、加減算パイプラインをそれぞれ 4 本
- 富士通、日立が同様なマシンを出荷、NEC は最後
- Cray は共有メモリマルチプロセッサに (Cray XMP, YMP)

1980年代にはこんなのも



PAX-128 4Mflops。筑波大学、星野ら

1990年代の「スパコン」

ASCI-Red, Intel Pentium Pro, 1.8TF

ASCI-Red

- 1996年に完成。200MHz Pentiun Pro プロセッサ 9216台。
- 3次元メッシュネットワーク (ほぼ2次元、、、)
- 核爆弾のシミュレーション用

2000年代の「スパコン」

IBM BG/L 360TF

IBM BG/L

- 2004年完成、カスタムプロセッサを最大131072個使用。3次元トーラスネットワーク
- ピーク性能360TF (もっと大きい構成もあった模様)
- 後継の BG/P, アーキテクチャを一新した BG/Q の開発後、プロジェクト解散

2010年代の「スパコン」

ポートアイランドの「京」コンピュータ、10PF

「京」

- 2012 年完成、カスタムプロセッサを 8 万個使用。6 次元トーラスネットワーク
- ピーク性能 11PF
- 商用版富士通 FX10, 「ポスト FX10」が開発された。今年度から「富岳」(ポスト京) への入れ換えが始まる。

現在 (2000 年以降) の普通のスパコン

- Intel のプロセッサを使用
- アプリケーションにあってれば NVIDIA の GPU を使用
- 沢山並べる
- 高速ネットワークを使う、あるいは Cray 社のシステムを買う
- 独自プロセッサとかはあまり使われない。
- あんまり芸がないので写真は省略

過去のスパコンの進化

何の話をしたかったかというと：何故計算機はどんどん速くなったのか？
基本的な理由:

- 使うスイッチ素子が高速になった
- 使うスイッチ素子が小型、低消費電力になって、沢山使えるようになった
- 使うスイッチ素子が安くなって、沢山使えるようになった (スパコンの物理的大きさは70年代が最小。そこまで段々小さくなって、そこからまた大きくなった)

素子の高速化

- といっても、真空管でもそれなりに速かった。
- スイッチング速度が重要でないわけではないが、配線を信号が伝搬する速度のほうが昔から重要。
- 昔は信号はほぼ光の速さでつたわった。
- 最近の LSI 上の配線は非常に細く (抵抗が大きく)、キャパシタンスを充電しないといけないことによる RC 遅延のため、信号が伝わる速度は光速度よりはるかに低い。
- 太い配線に大電流を流せば速いが、大量の電力消費になる

素子の小型化

- 真空管 → トランジスタ →IC という進化は 1970 年代までは重要
- サイズだけでなく、消費電力が下がることが重要
- 80 年代から重要になったのは CMOS LSI の微細化。10 年でサイズが 1/10 になる
- CMOS 素子では (2000 年くらいまでは) 微細化すると電圧を下げることができ、消費電力が下がり、速度は向上した。いわゆる CMOS スケーリング。
- 2000 年頃からは電圧が下がらないので、電力はちょっと下がるが速度は上がらなくなった。いわゆる CMOS スケーリングの終焉。
- そろそろ微細化も困難になってきた。また、トランジスタの構造・製造工程が複雑になり、微細化するとかえって価格上昇するようになった。いわゆるムーアの法則の終焉。

CMOS スケーリングって何？

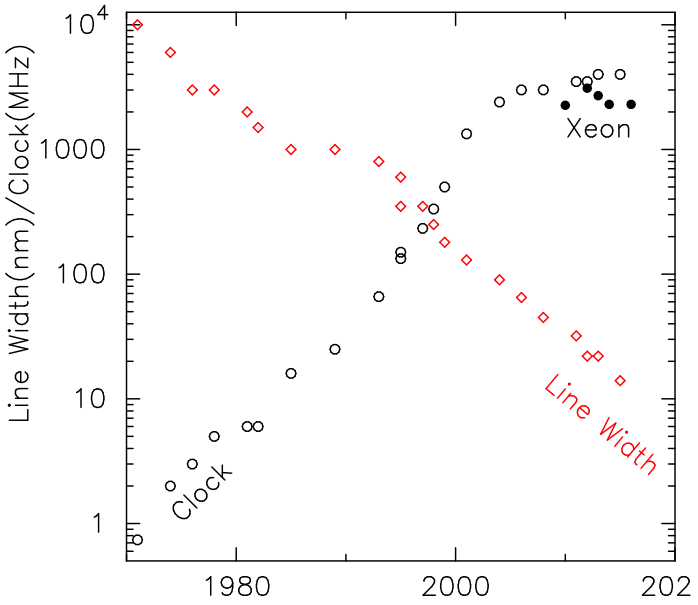
「Dennard Scaling」とも。

トランジスタのサイズ (3 次元的にすべて)、電源電圧を $1/k$ にし、不純物濃度を k 倍にすると、トランジスタの速度は k 倍、スイッチングあたりの消費電力は $1/k^3$ になる。

これが成り立つなら、

動作周波数 $\propto 1/$線幅
動作電圧 $\propto$ 線幅
性能 $\propto 1/$線幅^3 (電力、面積一定で)

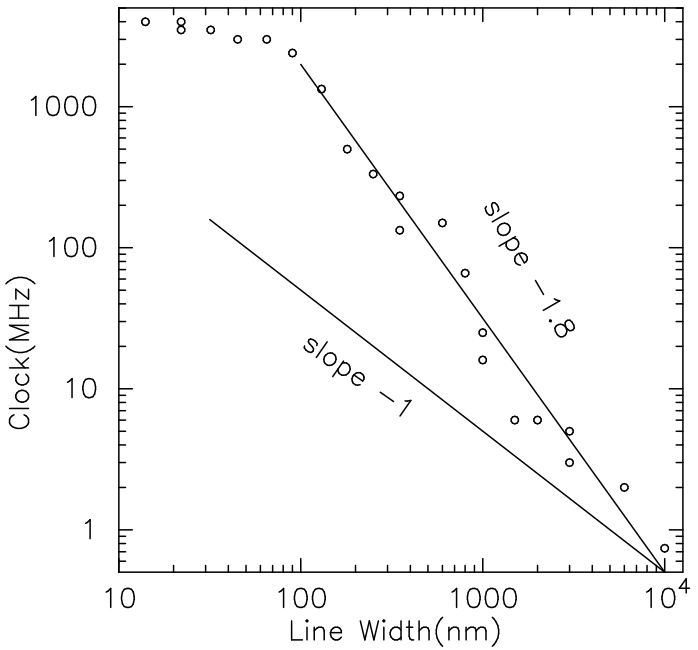
現実は



40 年間の Intel マイクロプロセッサの線幅 (トランジスタサイズ) と動作クロック

線幅は 40 年間で 3 桁縮小。クロックは 25 年間で 3 桁上昇、そのあと停止。(コア数の多い Xeon は段々クロック下がる)

線幅とクロック



線幅とクロックの関係

90nm までは

クロック $\propto$ 線幅$^{-1.8}$

そのあとはほぼクロック一定

(**Xeon** は下がる、、、、)

つまり

- ムーアの法則 (トランジスタサイズは時間の指数関数) は確かに成り立っている
- いわゆる **CMOS** スケーリング (速度が線幅に反比例) は実は成り立っていない

何が起こったのか？

- **90nm** までは電圧をあまり下げず、パイプライン段数を増やすことでクロックをあげた
- これは非常に消費電力増やすので、**1** チップ **100W** になったところで限界
- そこからは、コア数やコア内演算器数を増やすことで性能向上

ちょっといきあたりばったり観あり。というわけで、、、

素子の性能向上、サイズ低下と スパコンの性能向上

「スパコンの進歩」は4つの時期に分けられる

I スパコンが完全パイプライン化した乗算器をもたない時期 (1969年まで)

II スパコンは1つ以上の演算器をもつが、1チップにはまだ演算器1つが入らない時代 (CMOS では1989年まで)

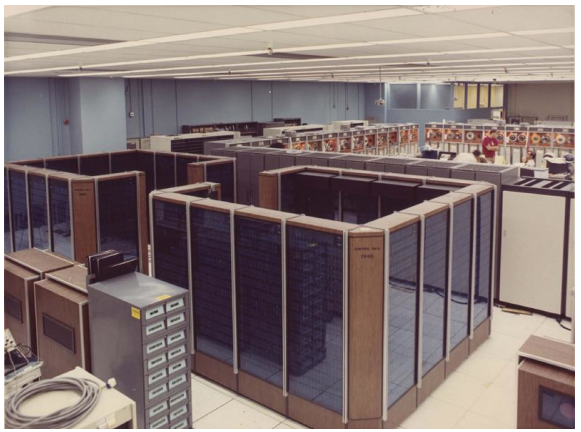
III1チップに複数の演算器がはいるが、微細化すると動作クロックが上がり、電力が下がった時代 (2001年頃まで)

IV チップに多数の演算器がはいるが、微細化してもクロックが上がらず、電力がへらない時代 (2001-)

このそれぞれで、素子の性能向上がどう使われたかは違う。

I期: 1演算器未満の時代 ～ 1969

- この時期には、メモリアクセスのほうが演算より速い
- 演算器の性能向上が最重要課題
- CDC 6600 くらいまで。
- 計算機の構成方法もまだ手探り。
- CDC 7600 で1演算器はいるようになる



CDC7600 36.4MHz clock

1969

IC はまだ使ってない

II期: 1演算器以上の時代 ～ 1990

- 1つ以上の演算器を有効に使うことが課題
- パイプライン化 (ベクトル命令)、複数の演算ユニットの SIMD and/or MIMD 並列実行が必要になる
- 演算器の数が増えると、メモリとどうつなぐかが課題になる。
- 演算器 16-32 個で破綻する (1992 年頃): メモリと演算器の間のデータ移動回路が大規模・複雑になり過ぎるため
- 末期の計算機: Cray T-90, 日立 S-3800
- 富士通 VPP500 では、分散メモリにすることで当座しのごうとした:ある程度の成功
- 他の方向: 1チップマイクロプロセッサでの分散メモリ。プロセッサ間の通信は細い線でいいことにする。これが 90 年代以降の主流になった

III期: 1チップ高速化の時代 ～ 2001

- 1チップマイクロプロセッサを分散メモリで多数つなぐのは II期末期から
- 1チップに1演算器以上がはいるが、ありあまるトランジスタを演算器を増やすのにはつかわないで動作クロックの向上、キャッシュメモリの大型化に使った時代
- 1990年代。 牧野の意見としては「失われた10年」
- 計算機全体としては II期に起こった問題がチップレベルでも起こるのを先送りにした
- メモリとの接続は速度が不足になった(memory wall): キャッシュでごまかす方針
- 迷走したプロセッサ開発も多い:各社の複数チップ共有メモリプロセッサ。(失敗プロジェクト: ASCI Red 以外のASCI マシン)
- 末期の計算機: Intel Pentium 4, DEC Alpha 21264

IV 期: 1 チップ多演算器化の時代 ～ 2020?

- 引き続き、1 チップマイクロプロセッサ、分散メモリ。
- デザインルール 130 nm あたりから、動作電圧低下、速度向上に限界
- マルチコア化、コア内 SIMD 化を同時に進めている
- 80 年代のベクトルスパコンの辿った道を追いかけている
- ということは、16-32 コアで破綻がくるはず
- 破綻がくることの予見: Intel Xeon Phi (60 コア)
- GPGPU はちょっと別。

まとめると

- 半導体技術は 2000 年頃に CMOS スケーリングが終焉、さらに微細化自体のスローダウンが進行中であり、もはや微細化が技術進歩を牽引していない。
- 昔の Cray-1 みたいな「スパコン」の進歩は、プロセッサコア 32 個くらいの並列度で限界、破綻した
- 現在のマイクロプロセッサは、それくらいのコア数に達してきており、破綻が近い (あるいは既に破綻している)
- この 2 重の困難をどう解決するか、という展望が必要。
- 「ポストムーア」は本当の問題ではない。

何故ポストムーアは問題ではないか？

現代の「最先端のマイクロプロセッサ」の設計は驚くほど非効率的

- チップ上のトランジスタの $\mathbf{97\%}$以上
- 消費電力の (多分)$\mathbf{90\%}$ 以上

は、「演算論理以外」に使われている (**Xeon Phi** の場合。**Xeon** だともっと非効率)。

何故それほど非効率なのか?

根本的な理由: 「計算機の設計」はまだ科学にも工学にもなってない。

- 熱機関の設計: 熱力学第一、第二法則が理論限界=理想の熱機関を与える
- 航空機の設計: 誘導抗力・摩擦抗力・それ以外 (「圧力抗力」) への分離が「理想の航空機」を与える

つまり: 科学的設計=理論限界に近づけること。

ところが:(トランジスタではなくて) 計算機の理論限界、という考え方自体がまだ存在していない。

流線形航空機

B. Melville Jones, The Streamline Aeroplane, Journal of the Royal Aeronautical Society, 33(1929)

THE STREAMLINE AEROPLANE

BY B. MELVILL JONES, A.F.C., M.A., F.R.A$_E$.S.

Ever since I first began to study Aeronautics I have been annoyed by the vast gap which has existed between the power actually expended on mechanical flight and the power ultimately necessary for flight in a correctly shaped aeroplane. Every year, during my summer holiday, this annoyance is aggravated by contemplating the effortless flight of the sea birds and the correlated phenomenon of the beauty and grace of their forms.

We all possess a more or less clear ideal of what an aeroplane should look like; a kind of albatross with one or two pairs of wings—depending on whether we live in Germany or Britain. In our more sanguine moments we even—like Alice and the cat—see the wings without the albatross. But progress towards this ideal, so far as the general purposes craft is concerned is, we must all admit, painfully slow. It has seemed to me that a contributory factor to the slowness of this evolution has been the lack of any generally understood and easily visualized estimate of what could be achieved were the difficulties in the way of realizing the ideal form overcome.

訳

航空力学の研究を始めてからずっと、私は機械的飛行で実際に消費される動力と適切に設計された航空機の飛行のために究極的に必要な動力の間に存在する莫大なギャップに悩まされてきた。毎年、夏の休暇の間、海鳥の努力なしであるかのような飛行と、彼らの形態の美と優雅さとの関係に思いをはせるごとに、この悩みはより深いものになった。
我々は誰でも、航空機がどのような形であるべきかについてそれなりに明確な理想をもっている。アホウドリのような形で、一対または二対―ドイツにいるか英国にいるかによって―の羽根をもっている。より楽観的な時には―アリスと猫のように―アホウドリなしで羽根だけを見ることもある。しかし、少なくとも汎用の航空機に関しては、この理想にむけた進歩は苦痛を伴うほどに遅いものであったことは我々全てが認めざるを得ない。この進歩の遅さの主な理由は、私の見るところでは、理想的な形態を実現するための困難が克服された時に何が達成できるかについての広く理解されることができ容易に描くことができる見積もりが存在しないことである。

アホウドリ

ソッピース・キャメル

(第一次大戦時のイギリスの戦闘機)

確かに見かけは違う

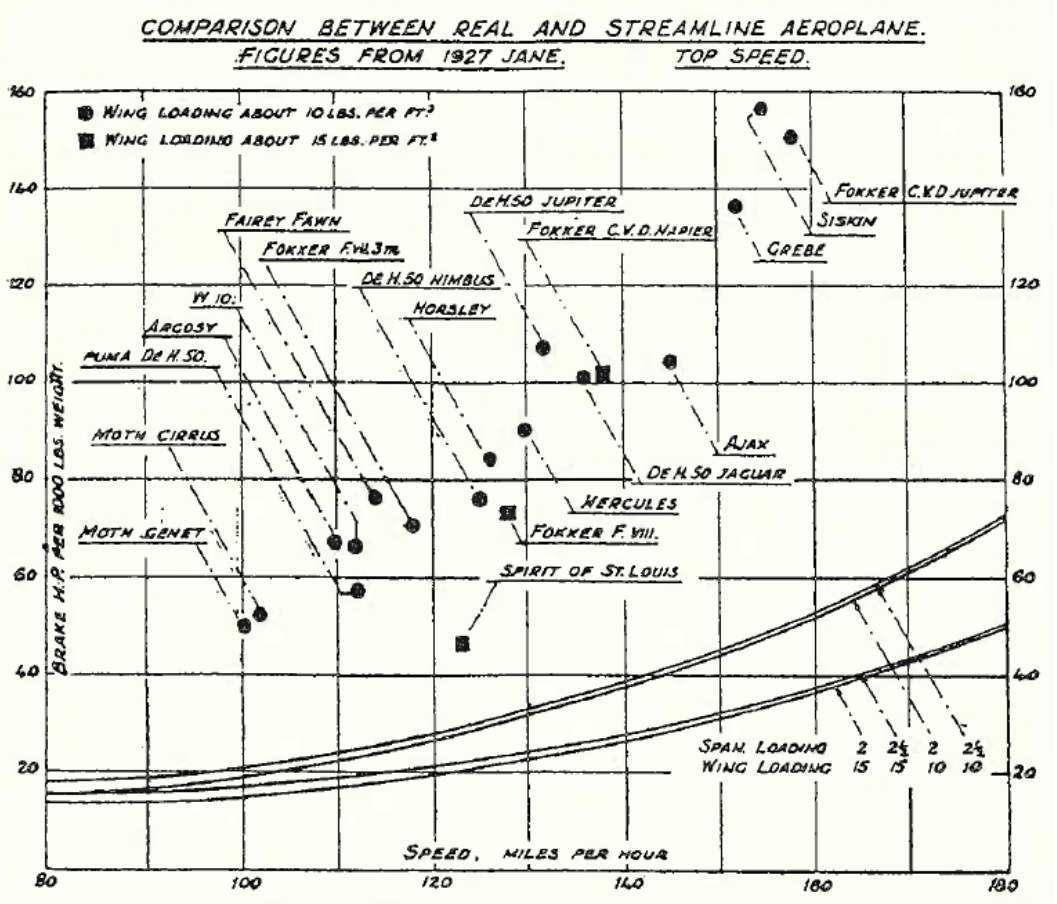
アホウドリは無駄なさそう。すっきりしている。ソッピース・キャメルはなんだかゴテゴテ色々なものが、、、

違いを定量化する

「なんとなく無駄がなさそう」では科学にも工学にもならないので、、、
問題: 空気抵抗をどこまで減らすことができるか
(原理的な) 解答:
減らせるものと減らせないものがある。

- 抗力
  - 誘導抗力 — 有限幅の翼ではゼロにはならない
  - 有害抗力
    - 圧力抗力 — 原理的にはどこまでも減らせる
    - 摩擦抗力 — 表面積で決まる限界あり

定量化した結果



横軸:速度

縦軸:重量あたり馬力

下の線:理想値。翼面荷重と翼幅荷重が違う４種

点は実機。一番下(良い)のはリンドバーグの **Spirit of St.Louis**

Spilit of St. Louis

流線形化のため操縦席から正面には窓がない。
良いといっても理想の3倍の抵抗
エンジンカウリング、引っ込み脚、片持ち翼、、、

現代の航空機

左: グライダー 右:ボーイング 787
実用機の B787 も随分スマート、理想に近いものになっている。

というわけで

計算機設計者が本来していないといけないこと:

1. 理想の計算機=理論的限界を明らかにする
2. 現実の設計をそれに近づける

現状やっていること

1. 現在ある設計を、現在ある色々な制約の範囲で色々いじってみる。思い付きのアイディアもいれたりする。
2. よさげなものを作ってみる
3. 会社がつぶれていなければ上を繰り返す

現状批判はいいとしてどうしたものか？

明らかにするべきこと:

> 理想の計算機を明らかにし、それに近づいていくべく努力する

では、理想の計算機とは？
飛行機、エンジンの類推では:同じ電力で一番沢山計算できる計算機 (「省エネ性能」)

「省エネ性能」の科学的定義

- 「論理設計」の効率を問題にする＝半導体技術は与える、ないし規格化する
- (細かいことをいいだすと物理設計も問題だがこれはちょっとおく)
- トランジスタは無限に沢山使ってもいいことにする (熱力学における準静的変化に対応)

この仮定の下で、ある問題をあるアルゴリズムで解くのに必要な電力消費の最小値は存在するはず。これを理論限界＝理想の性能とすればよい。
こういう方向での研究開発が今後重要。
では深層学習では？

深層学習と行列乗算

- 深層学習は多段ニューラルネット。ということは
  - 計算の $\mathbf{99.9\%}$ くらいが行列乗算 (BP 学習でも)
  - GPU が使われてるのは結局単精度行列乗算専用マシンとして
  - NVIDIA がものすごく成長したくらいの需要。
- 学習は多段ニューラルネットに対して BP が最適かどうかは (私には) 良くわからない。まあでも多分行列乗算ができればいいような方法にはなる。

深層学習向けの計算機

- ニューラルネットワークの1層は「行列ベクトル積」
- 特に、現在主流の「畳み込みネットワーク」では「行列行列積」
- 計算精度はあんまりいらないらしい。最近は16ビット表現が使われることもある。

現在の最先端

- **NVIDIA Tesla V100 (2017** 年夏発表)
- **16** ビットで 4×4 の行列同士の乗算の専用回路をもつプロセッサ
- **1** チップで **120 Tflops** 程度を実現。(「京」の **1** チップは **128Gflops** なので、**1000** 倍近く速い)
- 通常の **64** ビットだと **8Tflops** 程度。
- **2** 年たってもまだ最先端。計算機としては珍しいが、「ポストムーア」時代になったことの現れでもある。

他のところの状況

- **Google** は **TPU, TPU2** を開発、社内で利用。**TPU2** は **32** ビットの行列乗算専用回路を内蔵している模様 (詳細はまだ不明)
- **Intel**、**IBM** は色々なものを発表しているが、、、
- 日本メーカーも富士通、ルネサスその他
- 牧野も **Preferred Networks** と共同研究開発中。以下その話を少し。

MN-Core/GRAPE-PFN2とは?

- **PFN** と牧野のとこで開発している、主なターゲットは深層学習なプロセッサ。
- 深層ニューラルネットワーク **(DNN)** の学習で世界最高の電力あたり性能を実現することが目標。
- **FP16** (ライク) フォーマットでピーク **524TF** (予定のクロックで動けば、、、)
- 電力消費 (チップレベル) **500W** 以下、**1.2TF/W** あたり (目標)
- **(NVIDIA Volta: 300W, 132Tops, 0.44 Tops/W)**

時系列

- **2016/2** 牧野が **PFN** に呼ばれて (まだ本郷にあった) 色々議論
- **2016/6** 共同で **NEDO** のあんまり大きくない予算 (**4000** 万 × **2** 年) に応募 (その辺で **PFN** は現在の大手町に)
- **2016/7** 公式にプロジェクト開始。**NEDO** のお金では大きくチップ作れないので **40nm** のシャトル (**GPFN1**)、それにちょっと遅れて実機を **TSMC 12FFC** で (**GPFN2**、**PFN** 自己資金)。
- **2018/12 PFN** から正式発表

PFN の深層学習チップ開発へのコミットメント

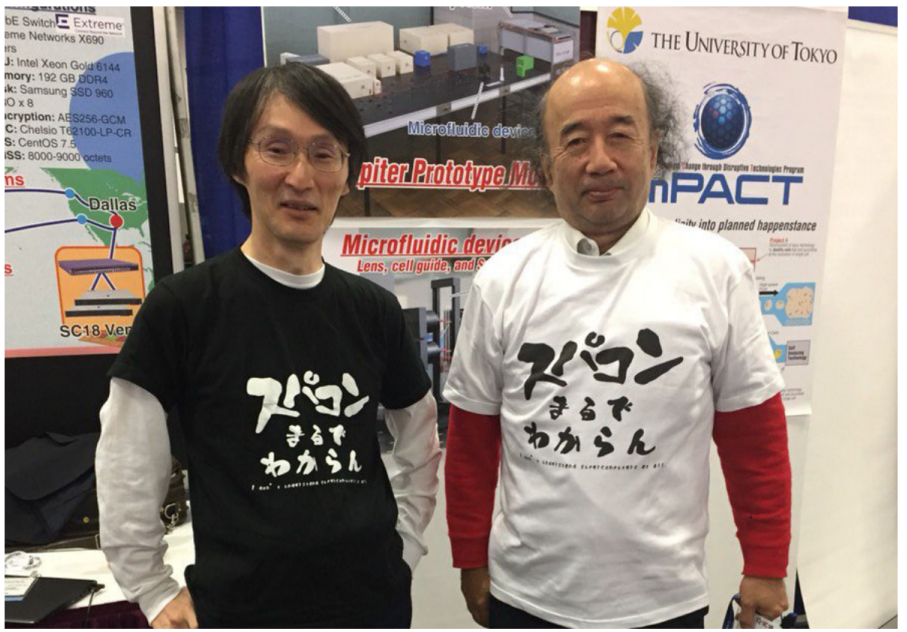
結構巨大プロジェクト。

- 現在の関係グループ: 30 人くらい (2016 年ほぼ 0 からスタート)
- リーダーの IBM(名村さん) とか、過去に 牧野と共同開発した人も。他にも転職してきたおっさん多数。PFN にしては平均年齢高い。
- 12FFC 開発費用: 15-20 億くらい？
- 次世代チップ計画始まった (GPFN3)

PFN “chip team”

MN-Coreは、NEDOの公募プロジェクトに採択されたところからはじまりました。

そのプロジェクトで制作したプロセッサでは、理化学研究所の村主 崇行氏、坪内 美幸氏をはじめとする、牧野教授の研究チームメンバーと共同で研究開発を行いました。MN-Coreは、そのときの知見を活かして設計開発しています。

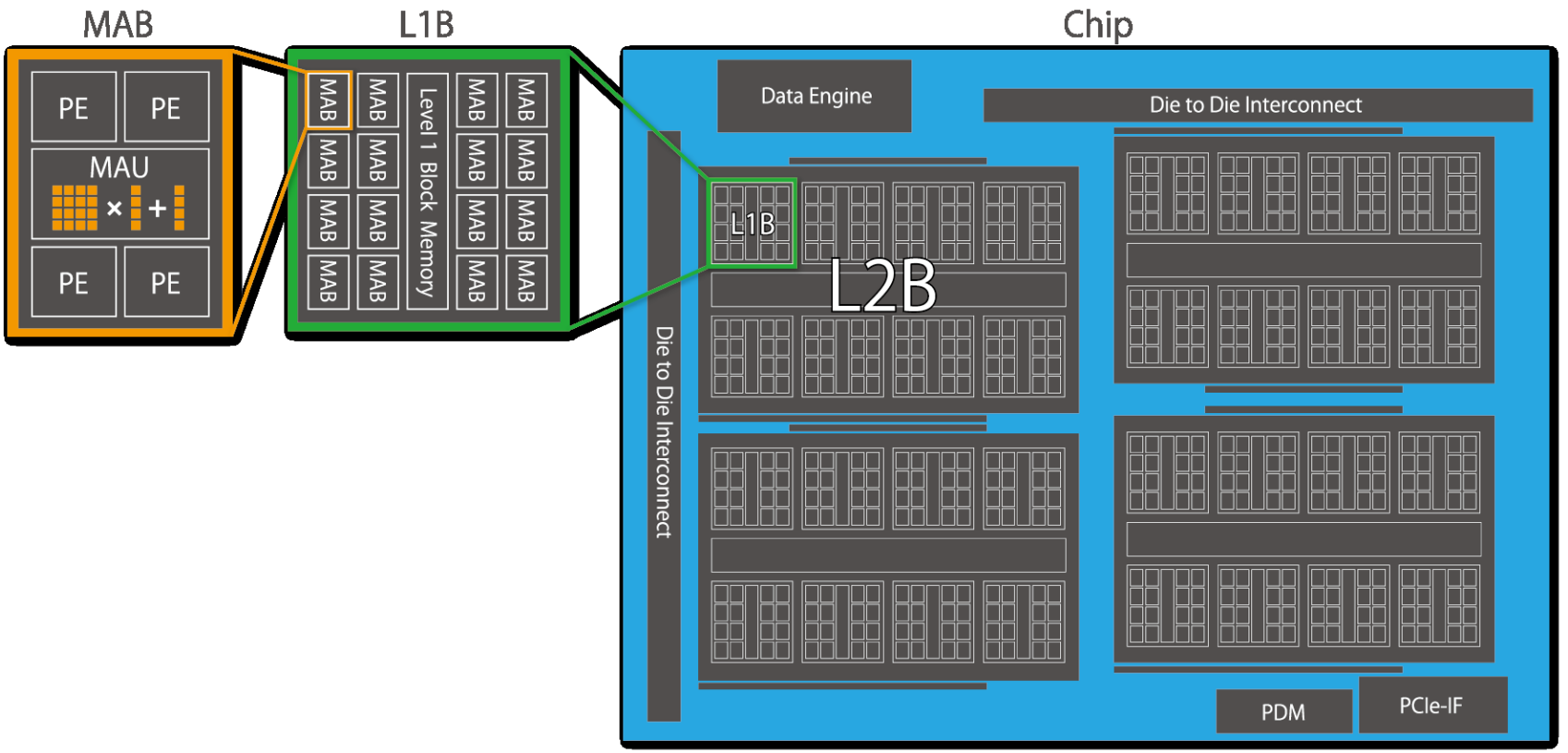


(左：牧野教授、右：平木教授。写真提供：稲葉真理 東京大学 准教授)

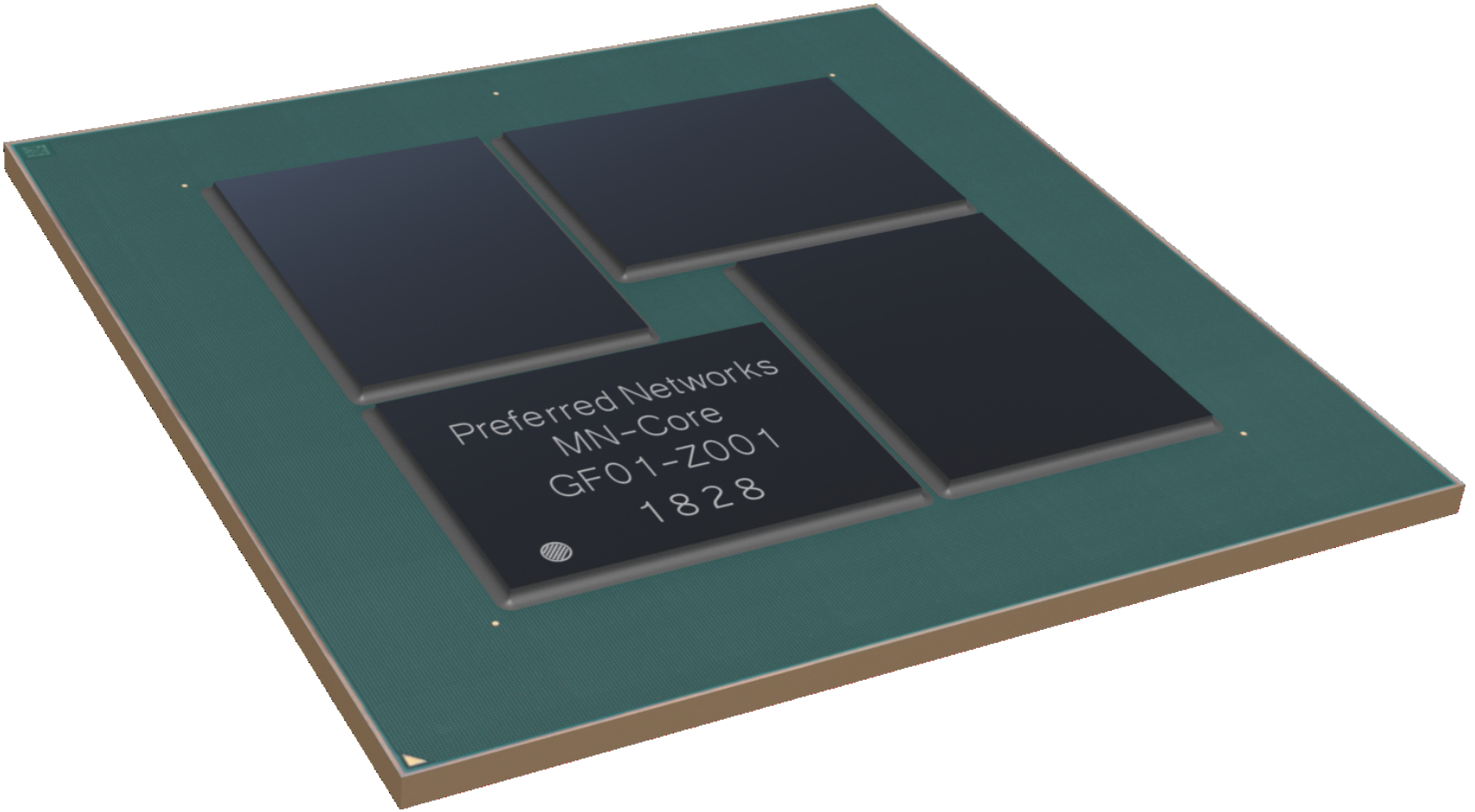
GPFN2 概要

- 1 カード: 1「モジュール」
- 1 モジュール: 4 ダイ MCM
- 1 チップ: PCIe (gen4, x16), xDDRx メモリ (内緒) 4 “Level-2 放送ブロック” (L2Bs)
- 1 L2B: 8 L1Bs
- 1 L1B: 16 MABs (Matrix Arithmetic Blocks)
- 1 MAB: 4 Processor Elements。これに対して 1 つ「行列乗算ユニット」
- FP64:FP32:FP16 性能比は 1:4:16
- カード全体が SIMD で動く。命令ストリームは 1 つ。

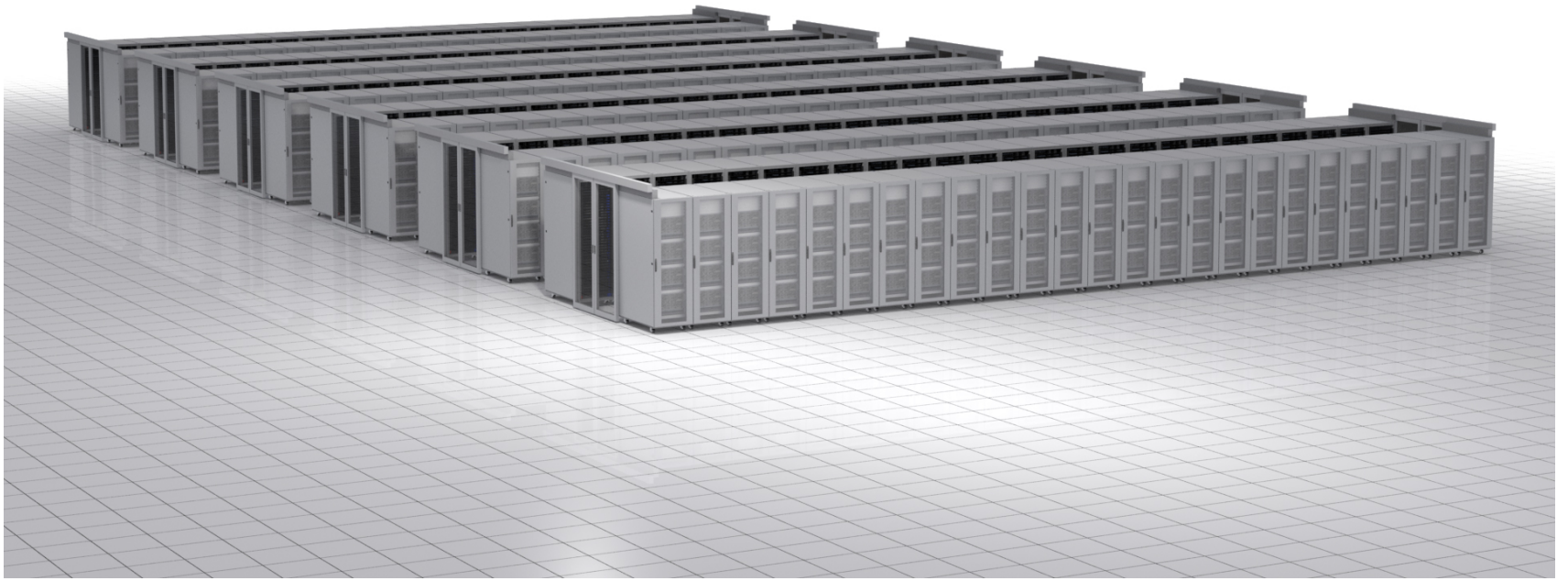
GPFN2 ダイアグラム



MN-Core



MN-3



まとめ

- 「深層学習」は、原理は1950年代のパーセプトロン、1980年代の3層ネットワークと変わらない
- が、計算機がすさまじく高速化したことで、巨大なネットワーク、巨大なデータを扱えるようになり、非常に強力になった。
- 計算機の進歩について、「ポストムーア」は今度は本当。シリコン半導体技術の限界は確実にきている。
- しかし、計算機設計という観点から見ると、問題はそこではなくて、非効率的な設計のほう。
- まず、熱機関や航空機なみの「効率」の概念を定義することが必要＝デザイン以前に科学が必要
- 深層学習については世界中でそういう方向に向かう。