Silicon Studio

エンド・トゥ・エンド AI描画に至る道

高度構造化入力のための ニューラルネットワーク構成や インフラの検討

シリコンスタジオ株式会社
フレドリック・ヘルツベルユ

Brand New!

講演内容

Silicon Studio

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

講演内容

Silicon Studio

1. 前置き
    1. 自己紹介
    2. このセッションについて
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

1-1. 自己紹介

Silicon Studio

- フレドリック・オット・マックス・フォルケ・ヘルツベルユ
- スウェーデン出身
- シリコンスタジオ株式会社
  - 入社6年目、研究開発室所属
- 前職：北京红剑公司，量子化学のGPGPU開発（北京）
  Ericsson（ストックホルム）

1-2. このセッションについて

Silicon Studio

- エンド・トゥ・エンドAI描画に至る道
  - 神経回路網でのリアルタイム描画を目標とする
- リアルタイムを前提として、手法について議論したい
  - デファクトスタンダードはまだ定まっていない
  - アーキテクチャの選択ができる貴重な機会
- 第一原理計算のアプローチで
  - 基本の命題から出発して、正しいアプローチであることを確認したい
  - 結果論ではない

1-2. 命題

- 下記の趣旨を主張したい：

① （現状描画に使われている）
神経回路網の構成にはGPU使用率の非効率がある　…3章

② リアルタイム描画なら非効率をなくせる可能性がある　…4章

③ そのための「ハッシュ分業」アプローチを提案する　…5章

– タイル内容のハッシュによって神経回路網の重みを選択して読み込む

講演内容

Silicon Studio

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

講演内容

Silicon Studio

1. 前置き
2. 背景
    1. やりたいこと
    2. 従来手法の問題
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

2-1. AIによる人間の顔の描画

Silicon Studio

- 人間は顔については非常に感受性が高い
  - 不自然だとすぐに気づいてしまう
  - 自然な顔なら映像の魅力が大幅にUp
- 顔の正確な表現は難しい
  - 肌の質感、瞳の表現
  - 超高精細なアセットが必要
- 人間の顔の実写映像は大量にある
  - 教師データに使える
- 顔の描画をAIで行うことは非常に有望

2-2. 従来のAI画像生成

- 近年、AIによる画像生成技術は劇的に進歩
  - 非リアルタイムのアプリケーションでは本物の写真と見分けがつかない
- しかし、リアルタイム描画は困難
  - 計算時間がかかる
  - 元になる画像が必要
  - 顔の造形やライティングなどの細かい指定ができない
- リアルタイムでの応用はまだあまり考えられていない
  - アーキテクチャから見直す必要があると考える

Face swapping
[Naruniec et al., 2020]

講演内容

Silicon Studio

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
    1. 非効率の概要
    2. GPUのメモリ構造
    3. 畳み込みカーネルの非効率
    4. チャンネル数とスケーラビリティ
    5. GPUの同期の構成
    6. データ依存性
    7. データ依存性と同期
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

3-1. 非効率の概要

- 神経回路網はレイヤー間のデータ依存性によって同期が必要
- 前のレイヤーの出力をまた次の入力として
  GDDR6メモリから読み込む
- 当該ピクセルに無関係なチャンネルが
  計算処理性能やメモリ容量を食い尽くす

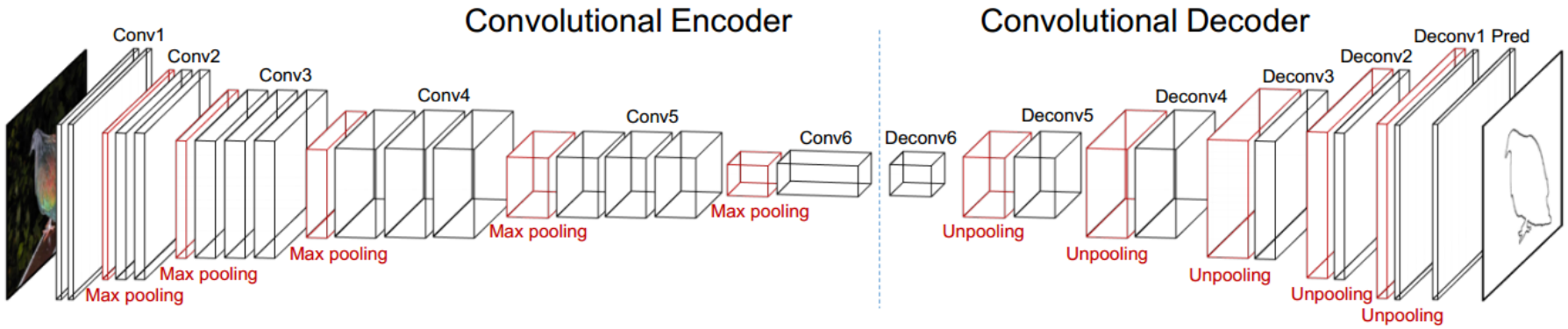


Figure 2. Architecture of the proposed fully convolutional encoder-decoder network.

https://nsarafianos.github.io/icip16

3-1. GPUのニーズ

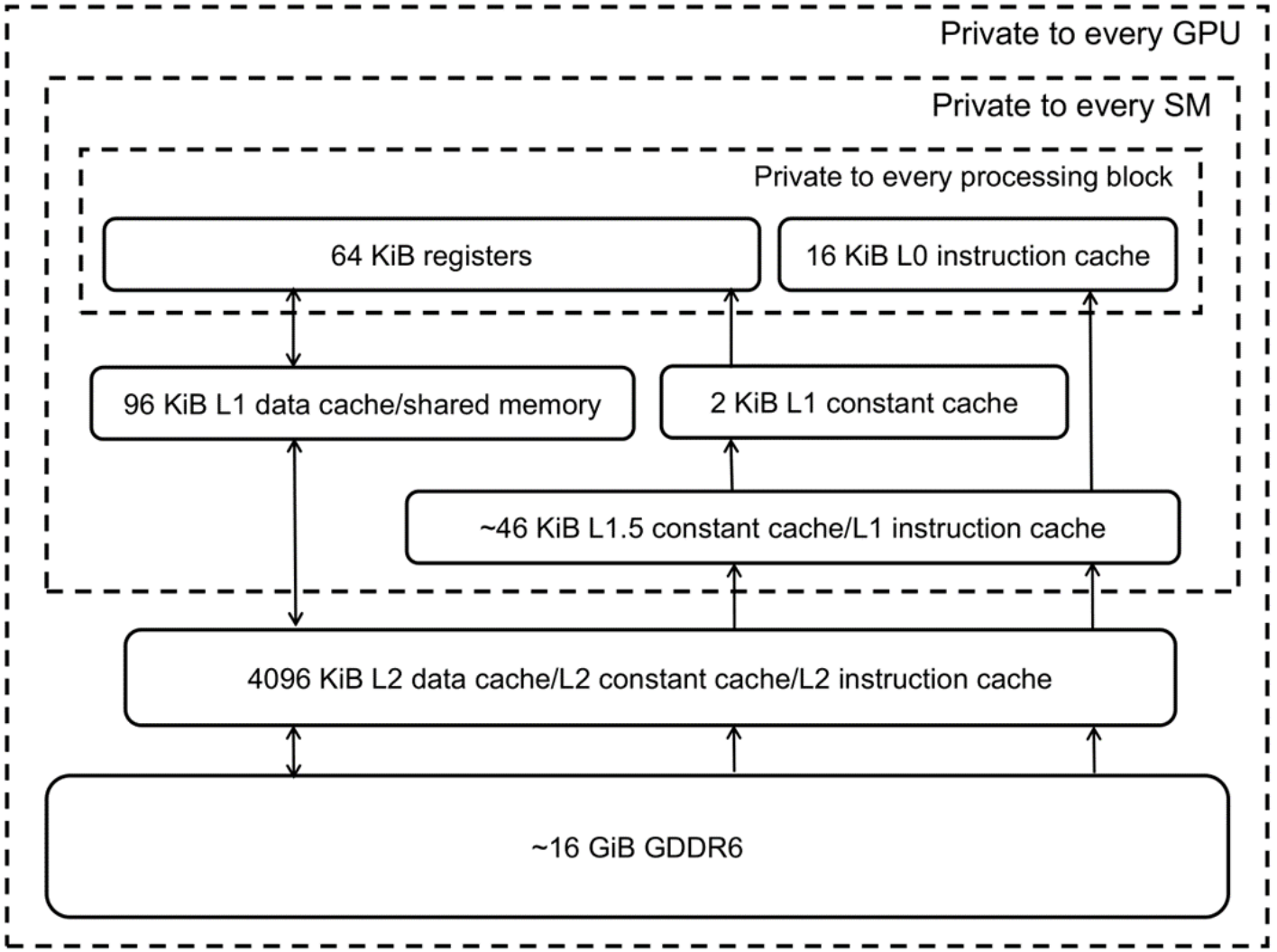
- 作業データセットはオンチップメモリに
  適合したサイズ(<32KB)が望ましい
- 独立したスケジューラユニット内で
  同期が行われることが望ましい
- 特徴量マップは、ハードウェアの積和演算
  (multiply-add)で容易に得られることが望ましい

3-2. GPUの構造１：メモリ

- SM当たり
  4つの命令スケジューラ
- 命令スケジューラ当たり
  64KBのレジスタ
- メモリ読み込み遅延：

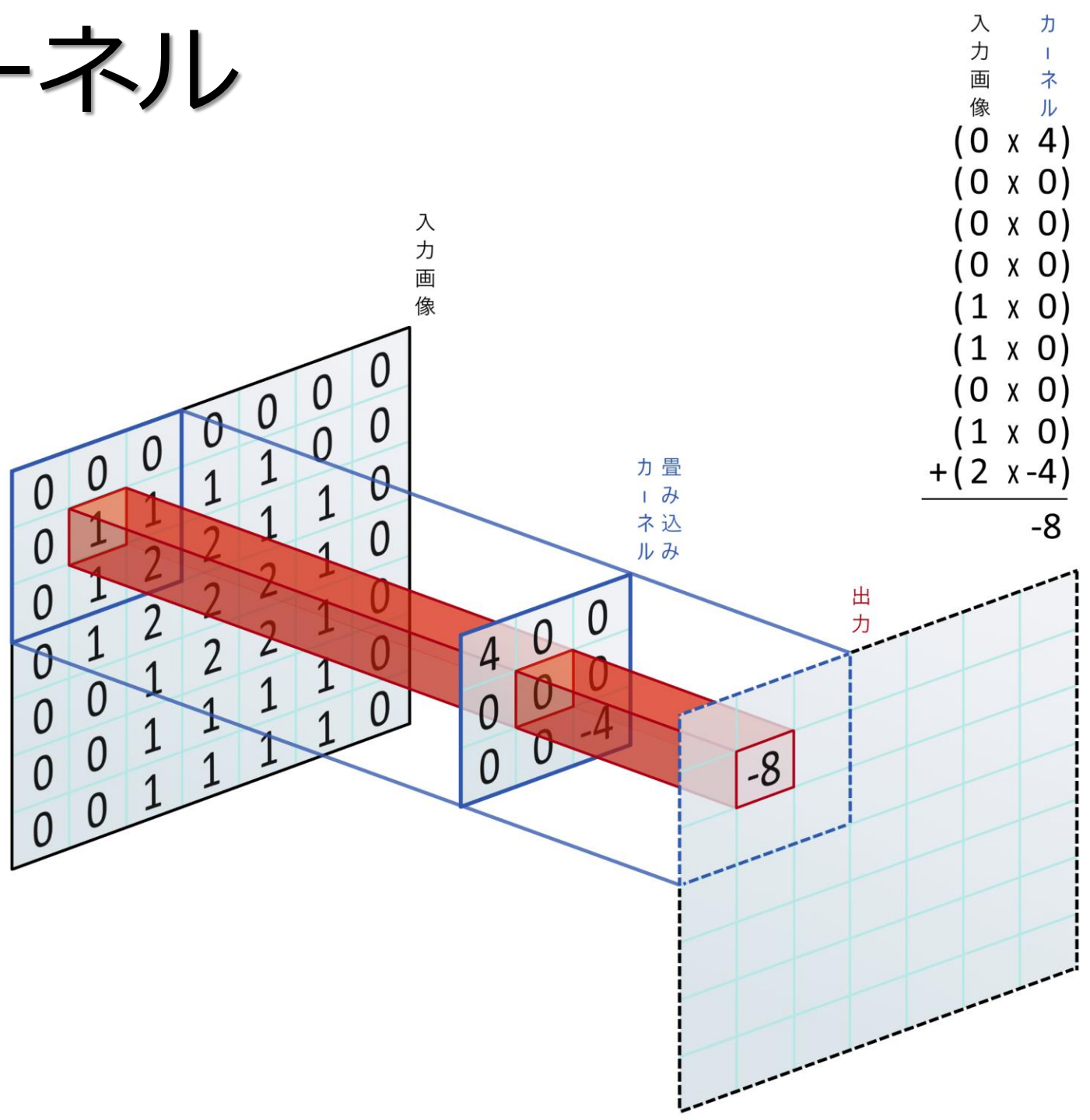
GDDR6	- 616 clock cycles
L2	- 188 clock cycles
L1	- 32 clock cycles
SMEM	- 19 clock cycles
Register	- 0 clock cycles

Zhe Jia, Marco Maggioni, Jeffrey Smith and Daniele Paolo Scarpazza. "**Dissecting the NVidia Turing T4 GPU via Microbenchmarking**“, 2019



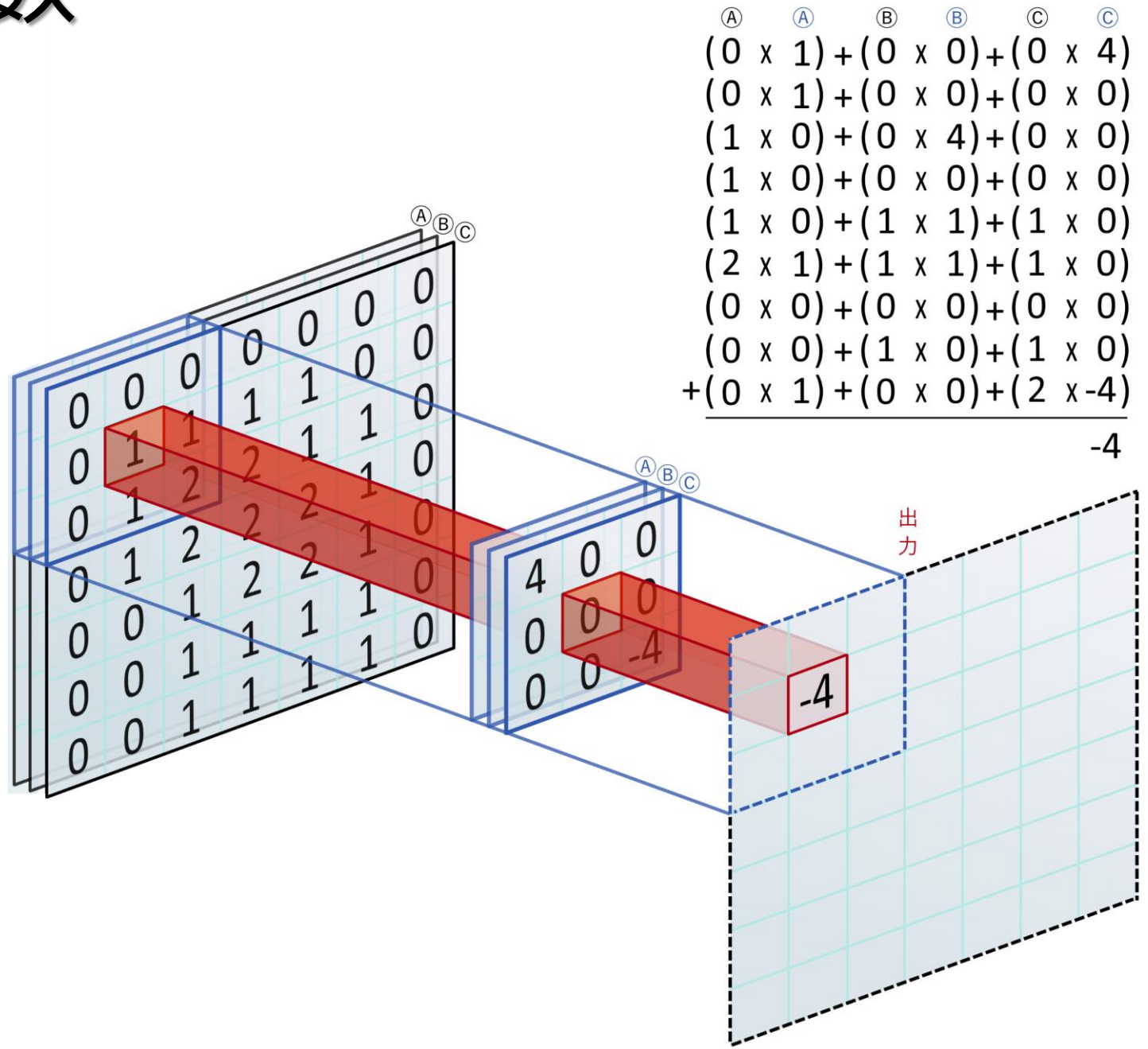
3-3. 畳み込みカーネル

- 畳み込みカーネルが
  各ピクセルに適用される
- カーネルサイズが≥2なら
  入力が重ね合わされる
- 画像に示す3 x 3のカーネルが
  標準的に用いられている



3-3. チャンネル数

- チャンネルが多いほど
  計算処理が増える
- 例えば、入力が3チャンネルなら
  (3x3x3)のカーネルになる
- チャンネルが多いほど
  ピクセル当たりの
  メモリ使用量が増える



3-3. 表現力

- 各チャンネルがそれぞれ異なる情報を持っている
- 例えば、
- tableに属しているか、chairに属しているかなど
- 神経回路網に表現力を持たせるため、多数(512 など)のチャンネルが用いられる
- しかし、各ピクセルにおいてはほとんどのチャンネルが無関係(値が0)になっている場合が多く、非効率

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, Antonio Torralba.
"**GAN DISSECTION: VISUALIZING AND UNDERSTANDING GENERATIVE ADVERSARIAL NETWORKS**“, 2018

3-4. チャンネル数とスケーラビリティ

Silicon Studio

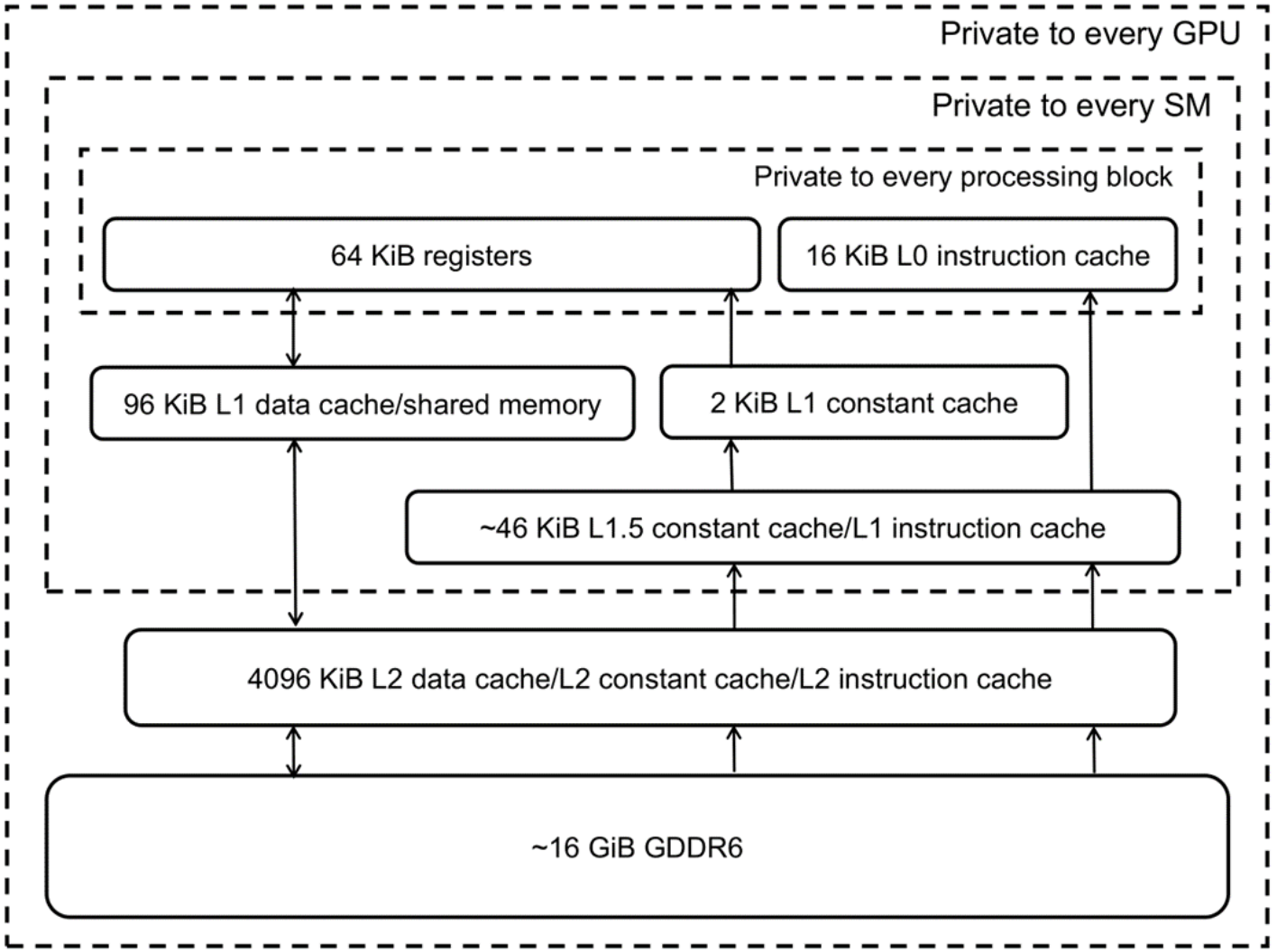
- 画像の品質を求めるほどチャンネル数が増える
- 表現力を求めるほどチャンネル数が増える

- チャンネル数による非効率化はスケーラビリティのボトルネックになる

3-5. GPUの構造２：同期

- 内部のシンクロは速い
- 外部のシンクロは遅い
- 別SMとのシンクロは
  基本的にシェーダ再起動
- 依存性の同期(概数):

  GPU内　- 20,000 clock cycles
  SM内　- 100 clock cycles
  Warp内　- 20 clock cycles
  Thread内 - 0 clock cycles



Zhe Jia, Marco Maggioni, Jeffrey Smith and Daniele Paolo Scarpazza. "**Dissecting the NVidia Turing T4 GPU via Microbenchmarking**“, 2019

3-6. データ依存性

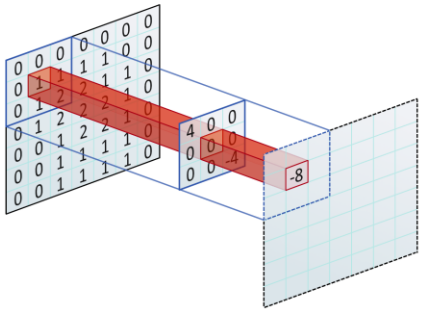
Silicon Studio

- 認識力のためにはデータ依存性が必要だと考えられる
- 最適化のためにはデータ依存性は最低限にしたい
- まず依存性の原因を見せたい

3-6. データ依存性の拡大

- 畳み込みカーネルによるレイヤーを多数重ね合わせてゆく
- 畳み込みカーネルの出力は次のレイヤーの入力になる
- 重ね合わせるためデファクトの入力領域が徐々に広がってゆく

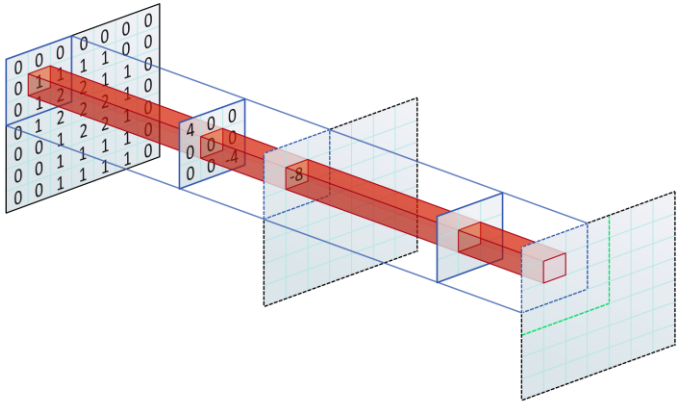
- 「緑」はデータ依存性ありの入力領域を意味する



3-6. データ依存性の拡大

- 畳み込みカーネルによるレイヤーを多数重ね合わせてゆく
- 畳み込みカーネルの出力は次のレイヤーの入力になる
- 重ね合わせるためデファクトの入力領域が徐々に広がってゆく

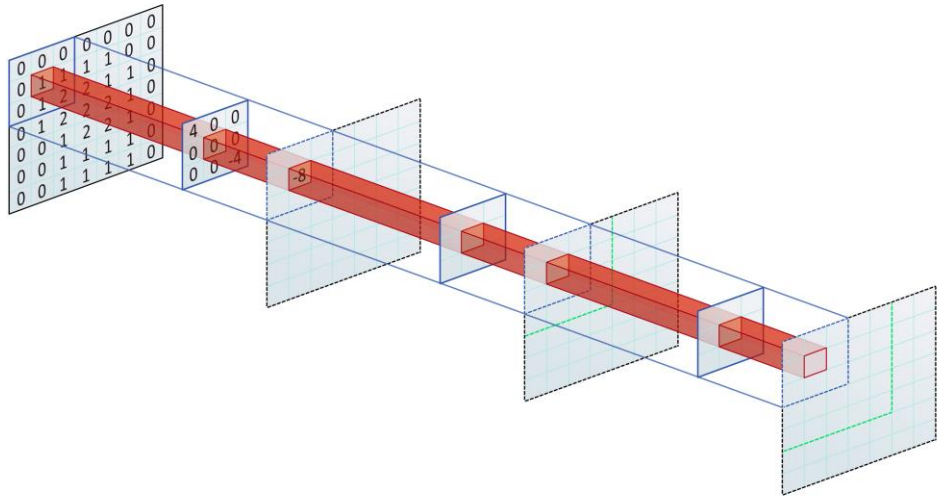
- 「緑」はデータ依存性ありの入力領域を意味する



3-6. データ依存性の拡大

- 畳み込みカーネルによるレイヤーを多数重ね合わせてゆく
- 畳み込みカーネルの出力は次のレイヤーの入力になる
- 重ね合わせるためデファクトの入力領域が徐々に広がってゆく

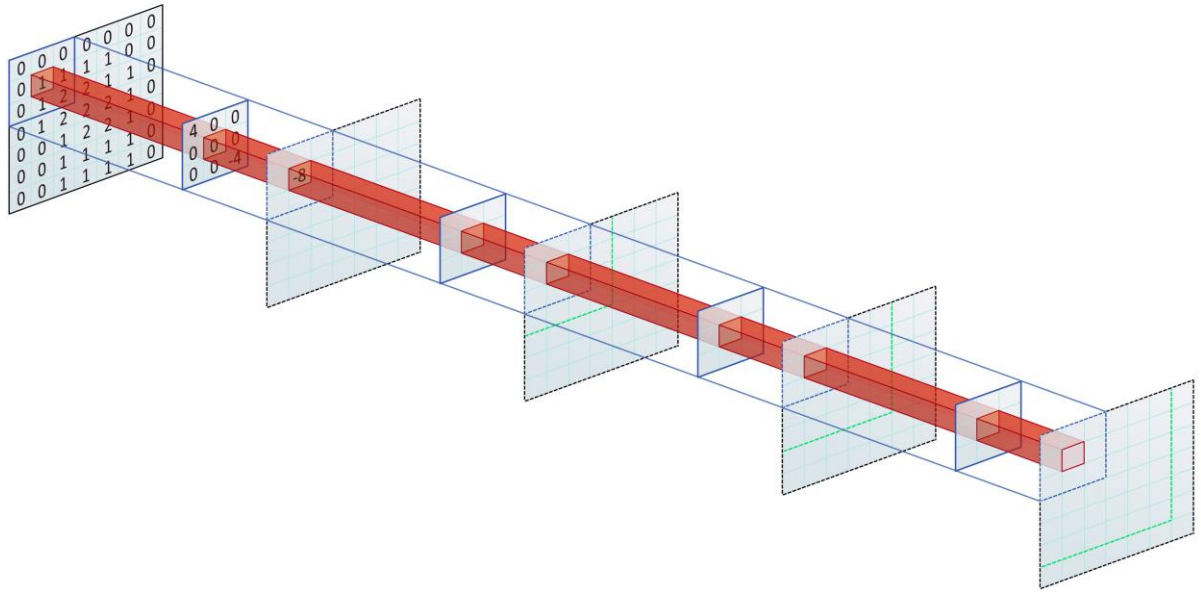
- 「緑」はデータ依存性ありの入力領域を意味する



3-6. データ依存性の拡大

- 畳み込みカーネルによるレイヤーを多数重ね合わせてゆく
- 畳み込みカーネルの出力は次のレイヤーの入力になる
- 重ね合わせるためデファクトの入力領域が徐々に広がってゆく

- 「緑」はデータ依存性ありの入力領域を意味する



3-7. データ依存性と同期

- 大域的なデータ依存性があると、各レイヤーの推移毎にシェーダを再起動する必要がある
- 同期として、シェーダ再起動は遅い（ほぼ１０μ秒？）
- ドライバーとAPIの設定があればシェーダ再起動より少し速い方法もある（CUDAのみ？）
- 独立したスケジューラ内の同期がもっとも速い（ほぼ１０ナノ秒？）

3. 非効率さのまとめ

Silicon Studio

- 神経回路網は
  レイヤー間のデータ依存性によって同期が必要
- 前のレイヤーの出力をまた次の入力として
  GDDR6メモリから読み込む
- 当該ピクセルに無関係なチャンネルが
  計算処理性能やメモリ容量を食い尽くす

講演内容

Silicon Studio

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
    1. 大域的なデータ依存の必要性
    2. リアルタイムレンダリングの事情
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

4-1. 大域的なデータ依存の必要性

Silicon Studio

- 求めている情報は当ピクセルの中になければ
  他のピクセルも参照する必要がある（＝依存性）

- 色情報(RGB)のみの入力では、
  当該ピクセルの情報としては基本的に足りない

- 画像認識や画像変換の場合なら
  データ依存性は仕方ないことと言える

4-2. リアルタイム描画では事情が異なる

- 入力は内容不明の写真ではなく、
  カメラやメッシュなどのバイナリデータから生成される
  - そのため、ピクセルに入れる情報は自由であり、
    必要であれば追加も可能
- この利点を活用するには手法の革新が必要

講演内容

Silicon Studio

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
    1. 提案手法の概要
    2. 畳み込みカーネル
    3. ハッシュ分業
    4. 高度構造化入力
    5. 教師データ生成
    6. トレーニング
    7. システムの実装
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定

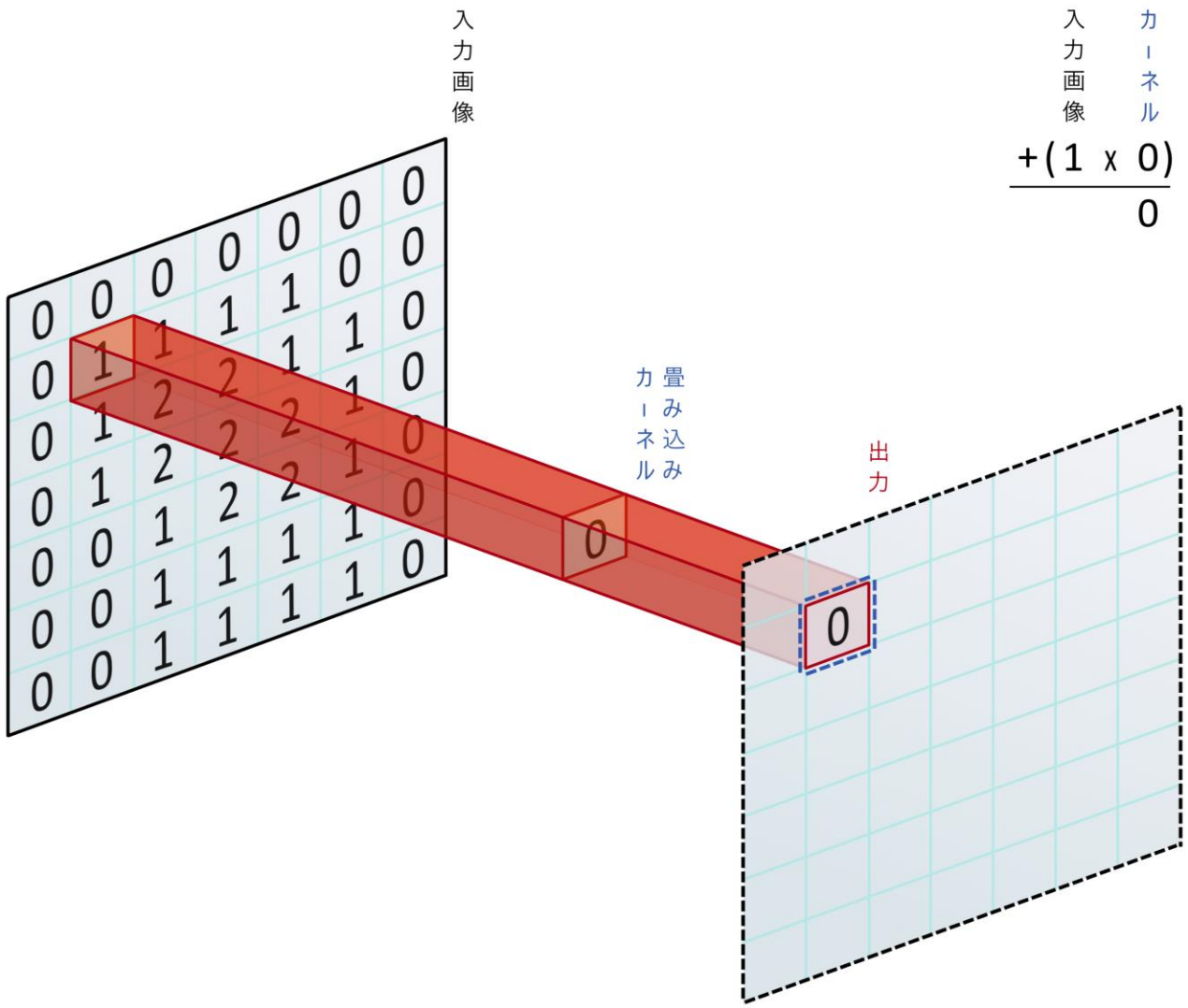
5-1. 提案する対策

- 下記の対策を講じる：
  - ① １ｘ１の畳み込みカーネルとすることでデータ依存性の問題を解決
  - ② １６ｘ１６の独立タイルとすることでデータがレジスタに収まり、レイヤー推移の同期の問題を解決
  - ③ タイルで応用される神経回路網が、そのタイルに関係あるチャンネルだけを利用する

5-2. 1 x 1 の畳み込みカーネル

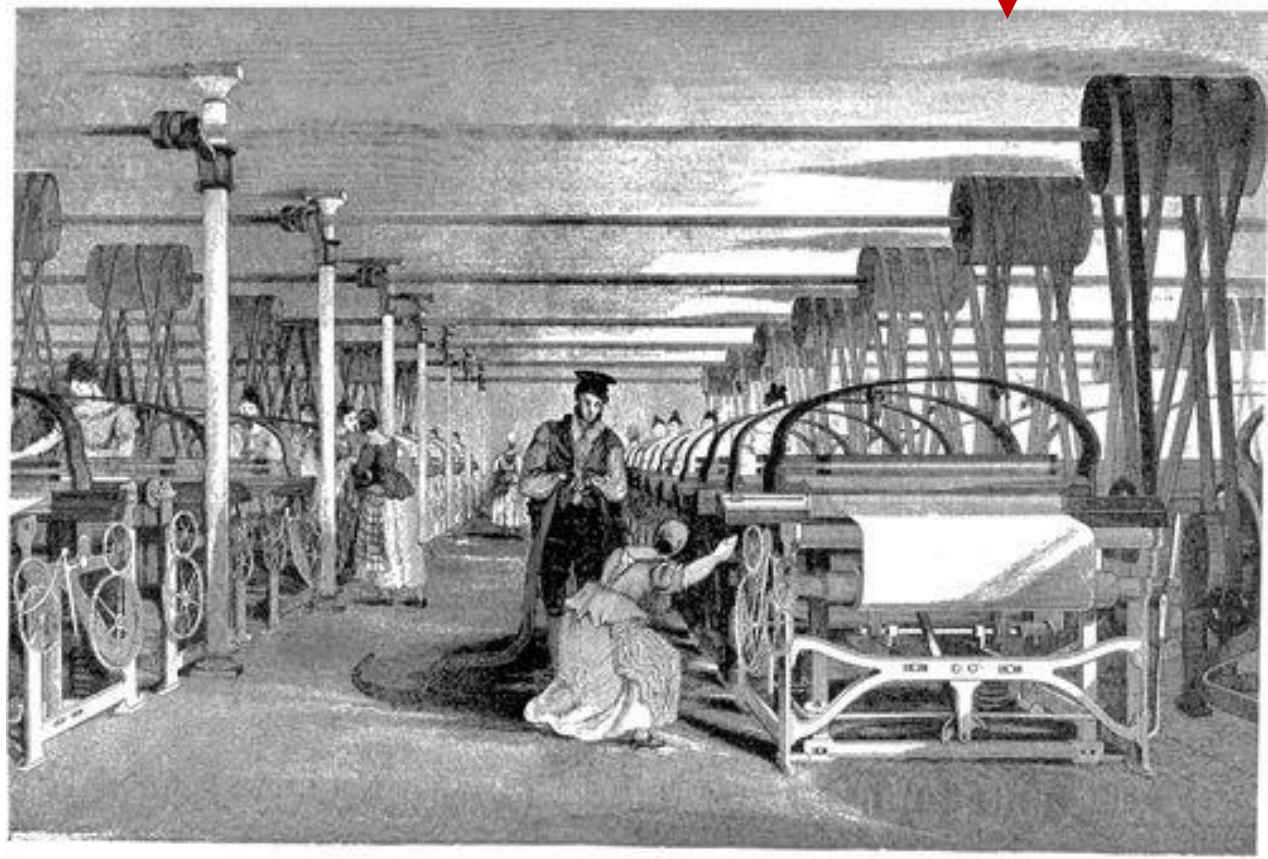
- 計算処理が減少する
- データ依存性がなくなる
- ピクセルのチャンネルを吟味する必要がある

- 周囲の情報を用いない単純な神経回路網
  - 顔の、ある特定の位置を描画することにのみ特化



5-3. 分業

- 産業革命の時、全体的に優秀な職人が狭い知識の専門家に差し替えられた
- 神経回路網にもそのようなDeskilling(脱スキル化)を応用する
- トレーニングの入力と応用の入力が合っているなら狭いドメインでも問題ない



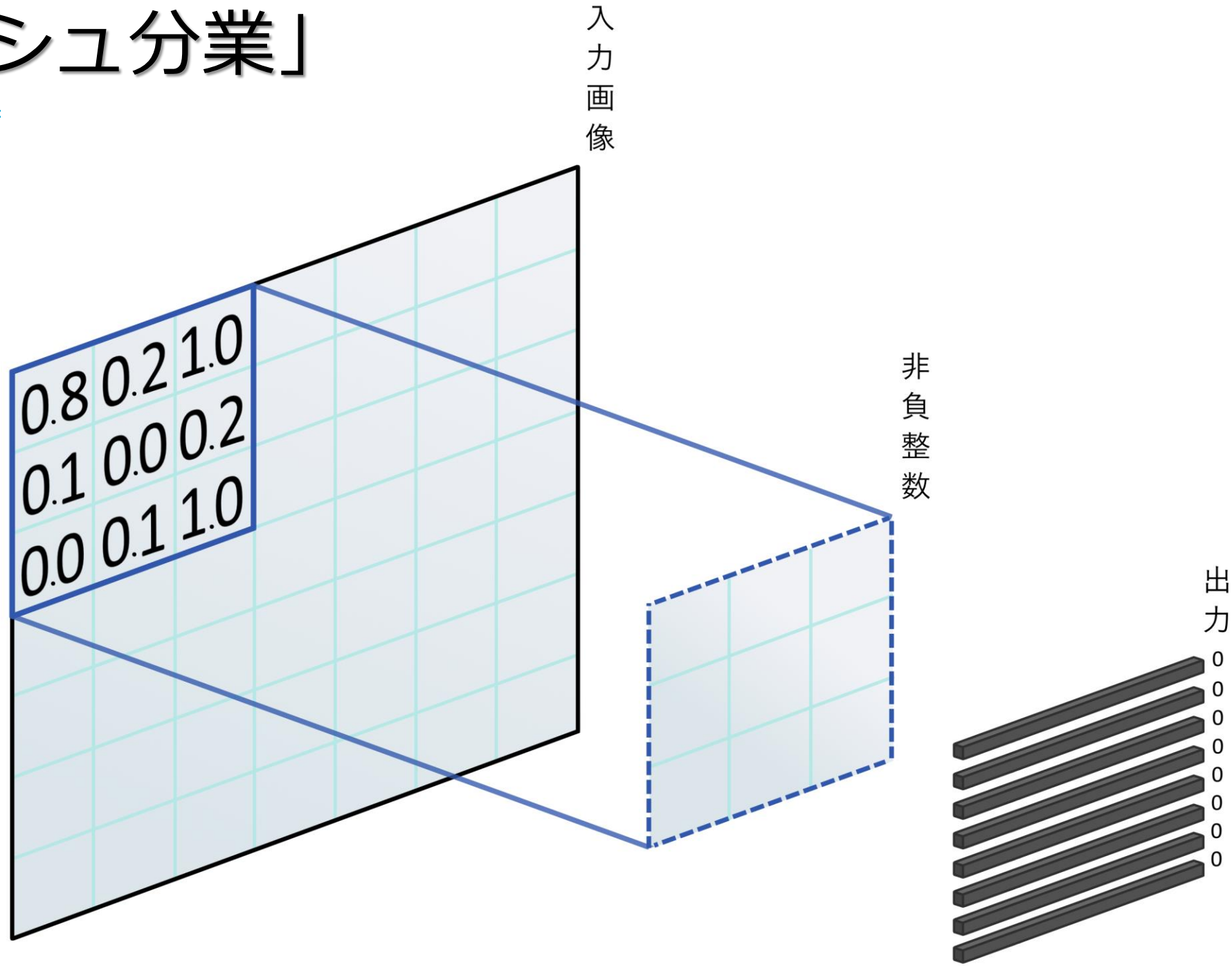
5-3.「ハッシュ分業」

- ドメインを管理するため、
  タイルのピクセル内容からハッシュを計算
- 疑似コード：

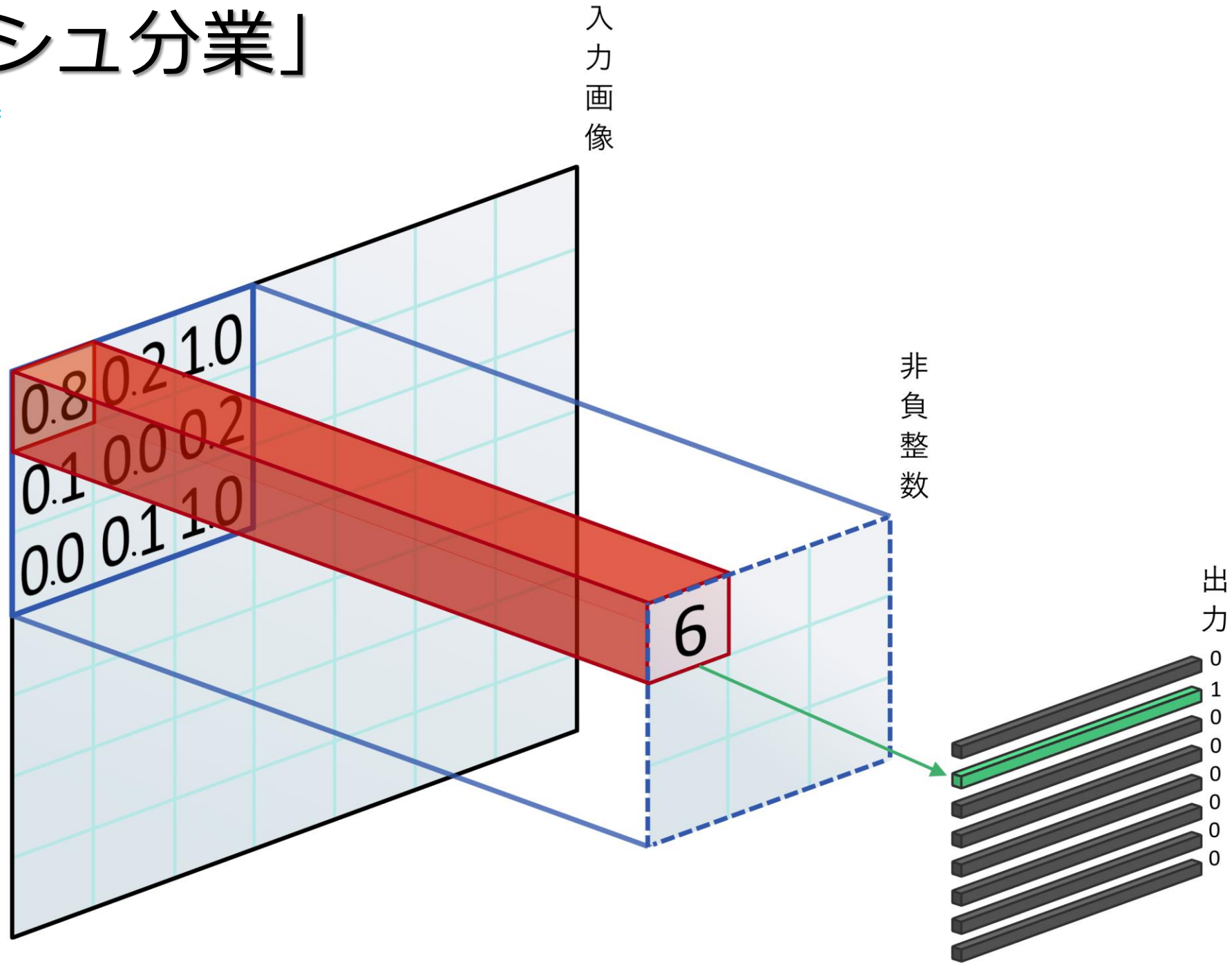
```
for( pixel in tile ) { hash_u |= (1<<pixel.u) }
for( pixel in tile ) { hash_v |= (1<<pixel.v) }
```

- ３Dモデルのもつテクスチャ UVは理想的
  - 基本は０→１
  - その上にスケーリングする（例えば８倍）
- 「配列index＝ハッシュ」で重みを効率的にロードできる

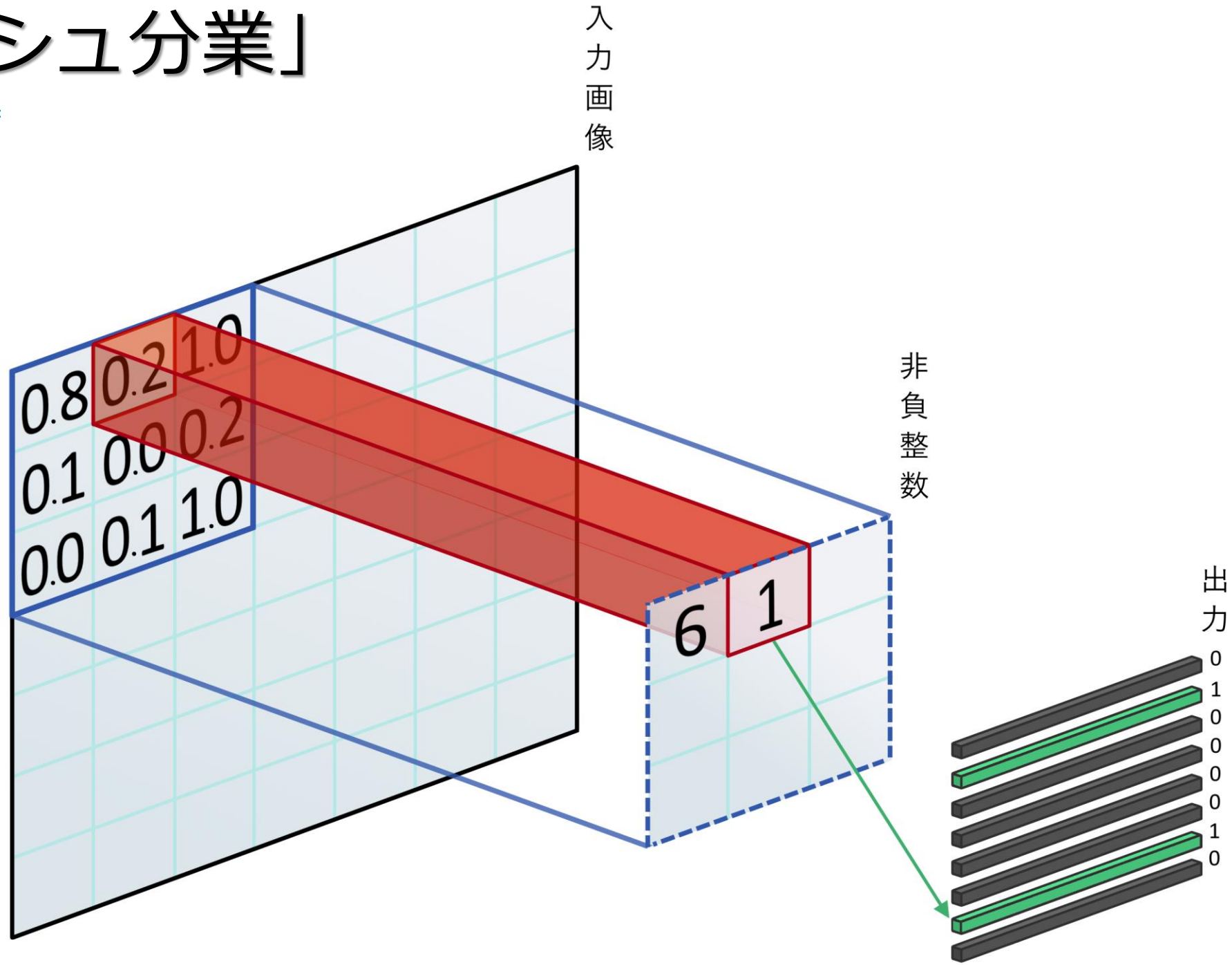
5-3.「ハッシュ分業」



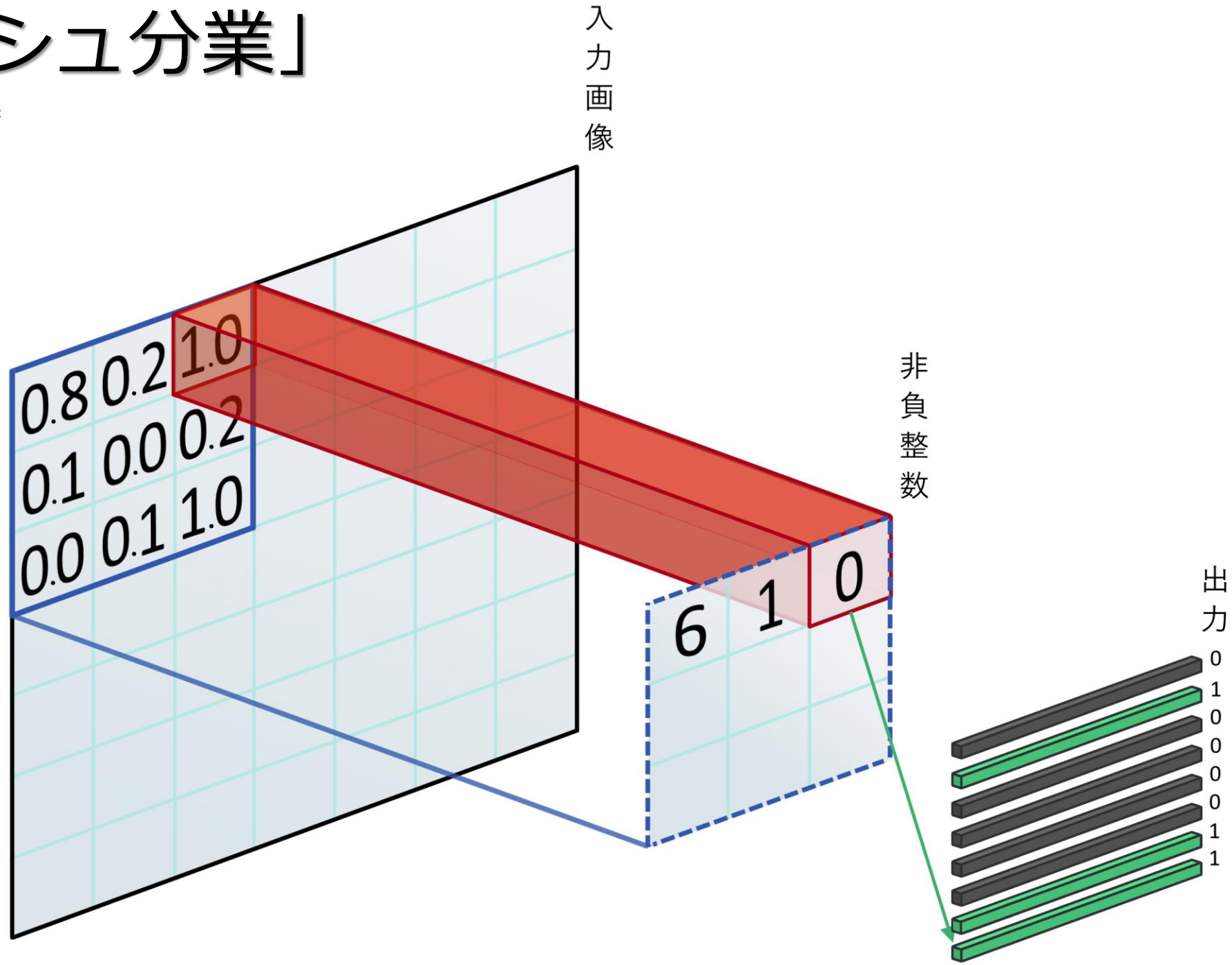
5-3.「ハッシュ分業」



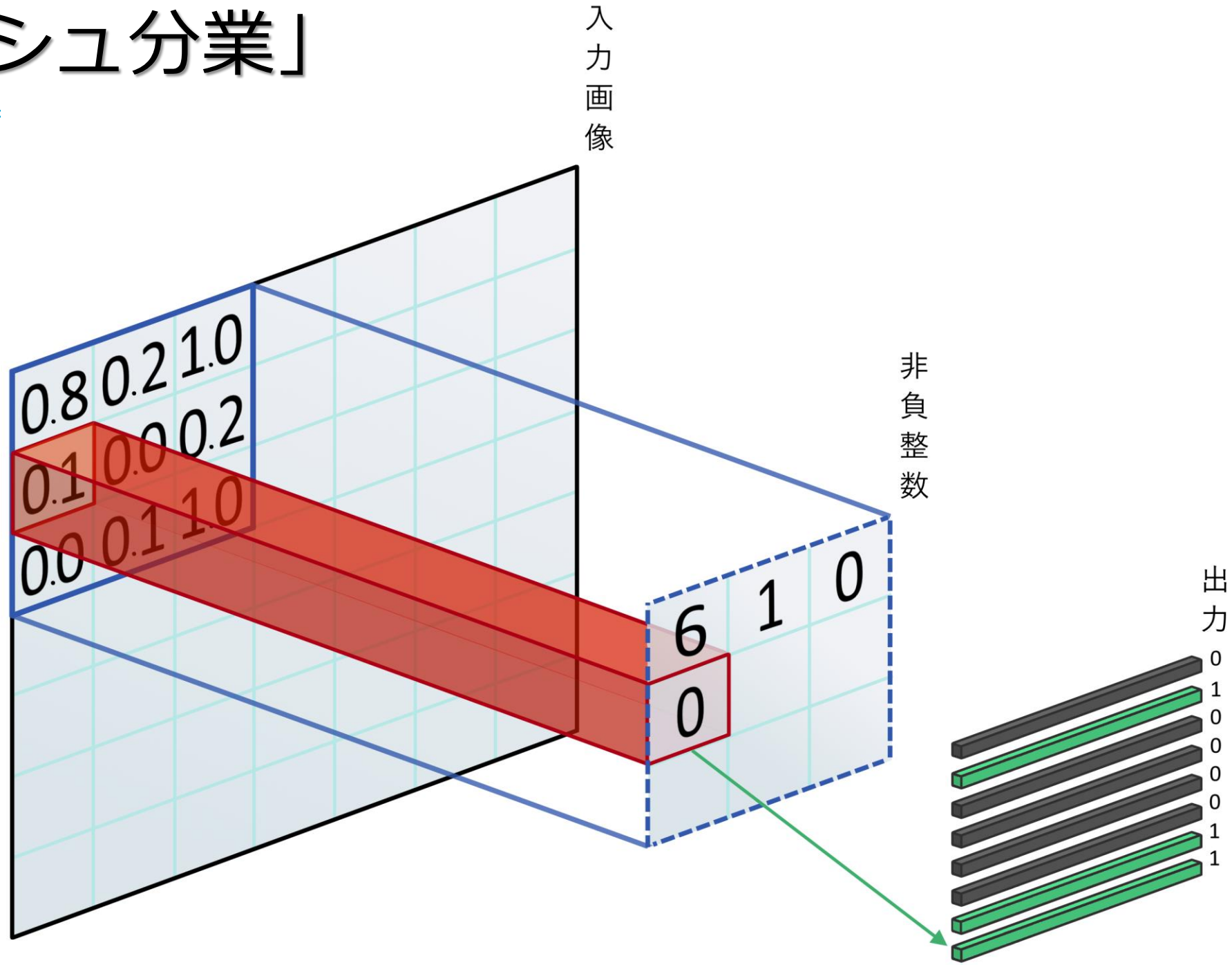
5-3.「ハッシュ分業」



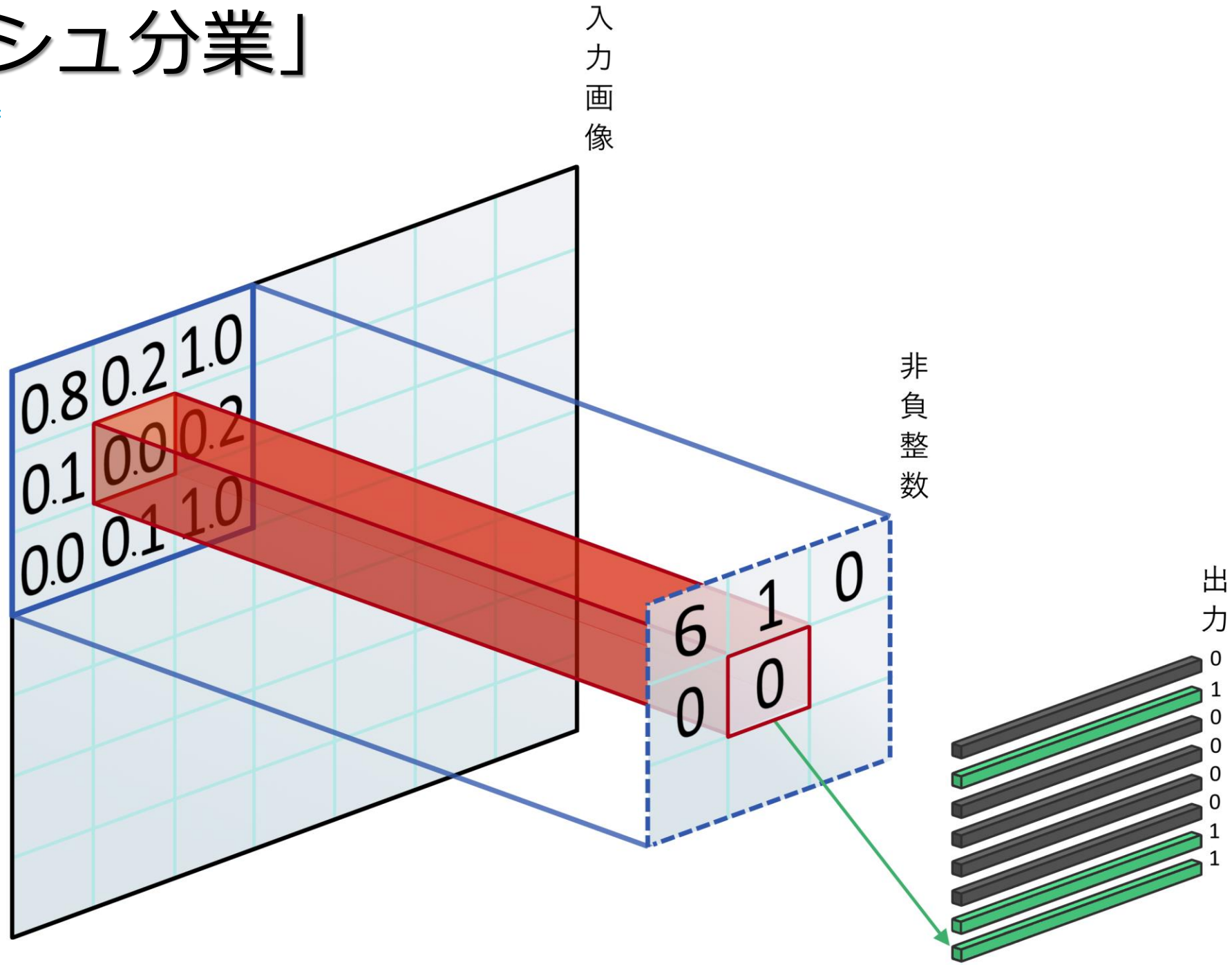
5-3.「ハッシュ分業」



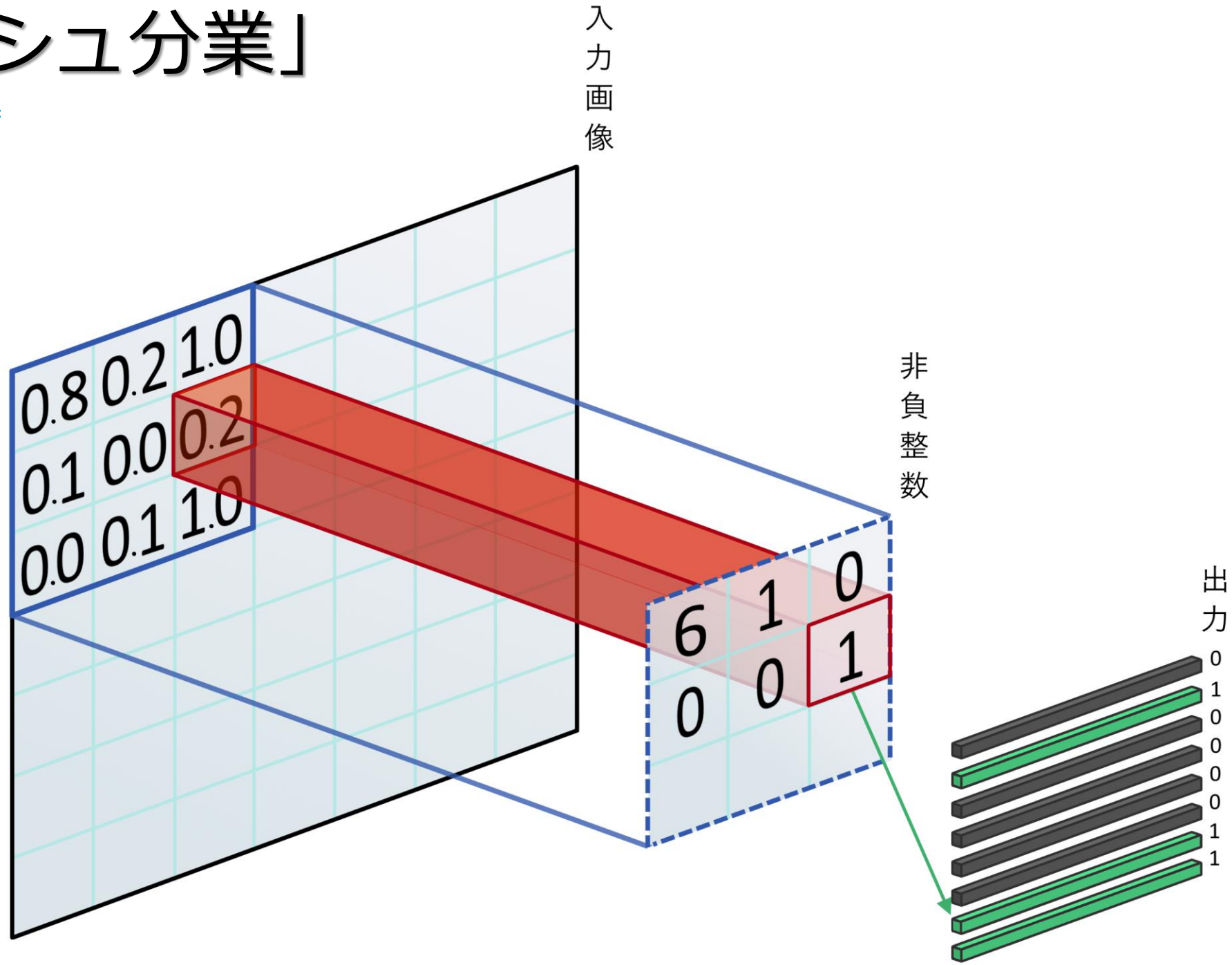
5-3.「ハッシュ分業」



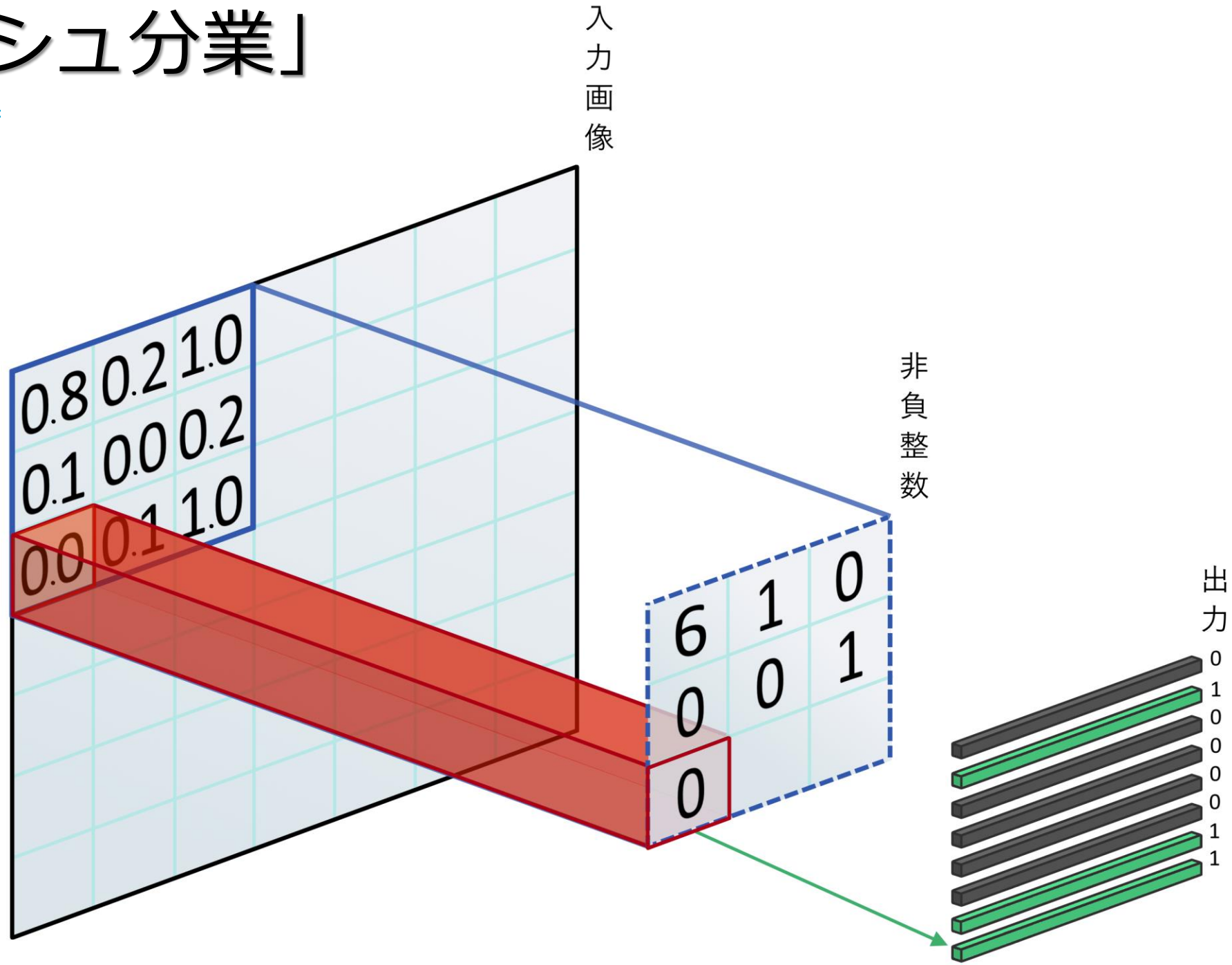
5-3.「ハッシュ分業」



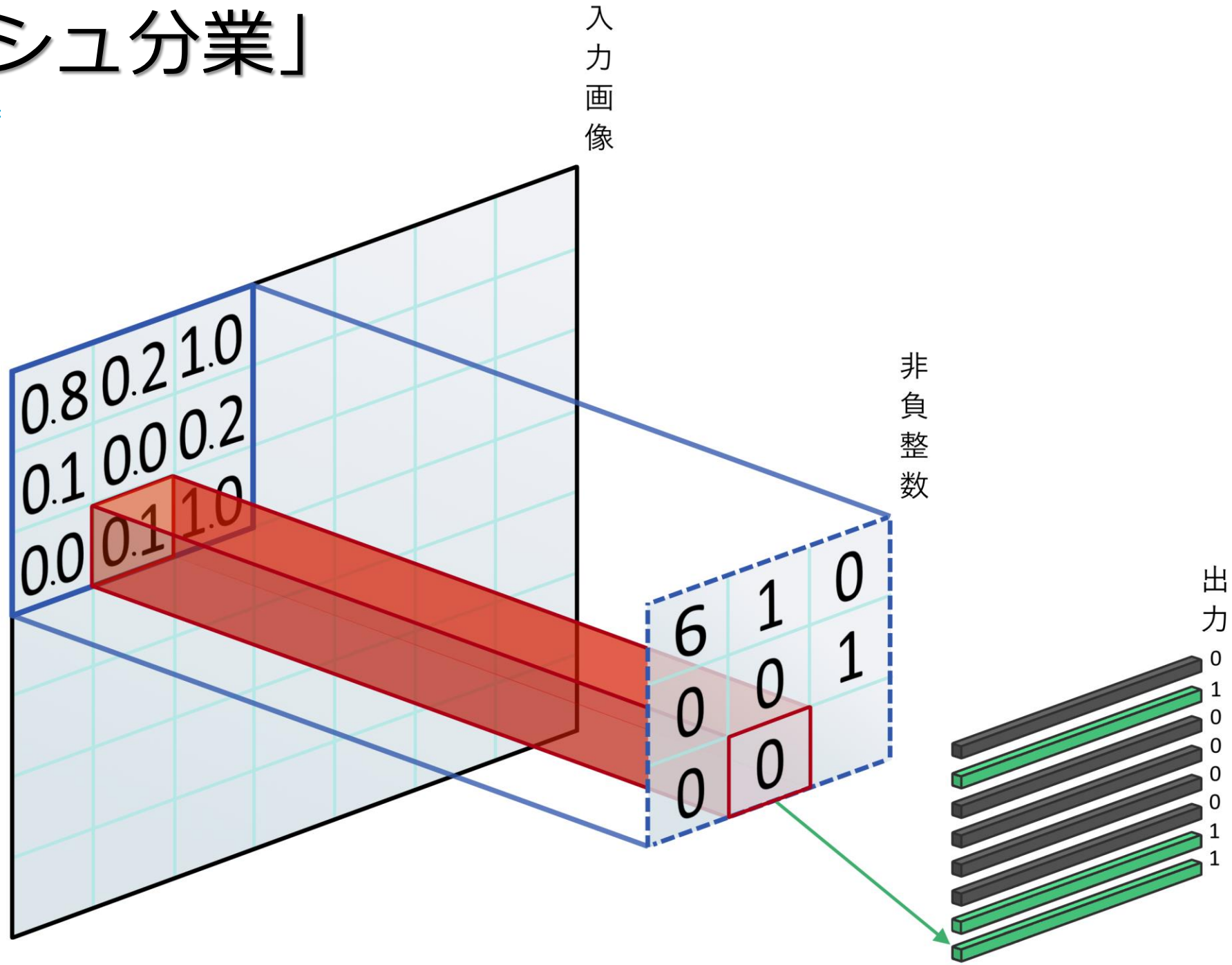
5-3.「ハッシュ分業」



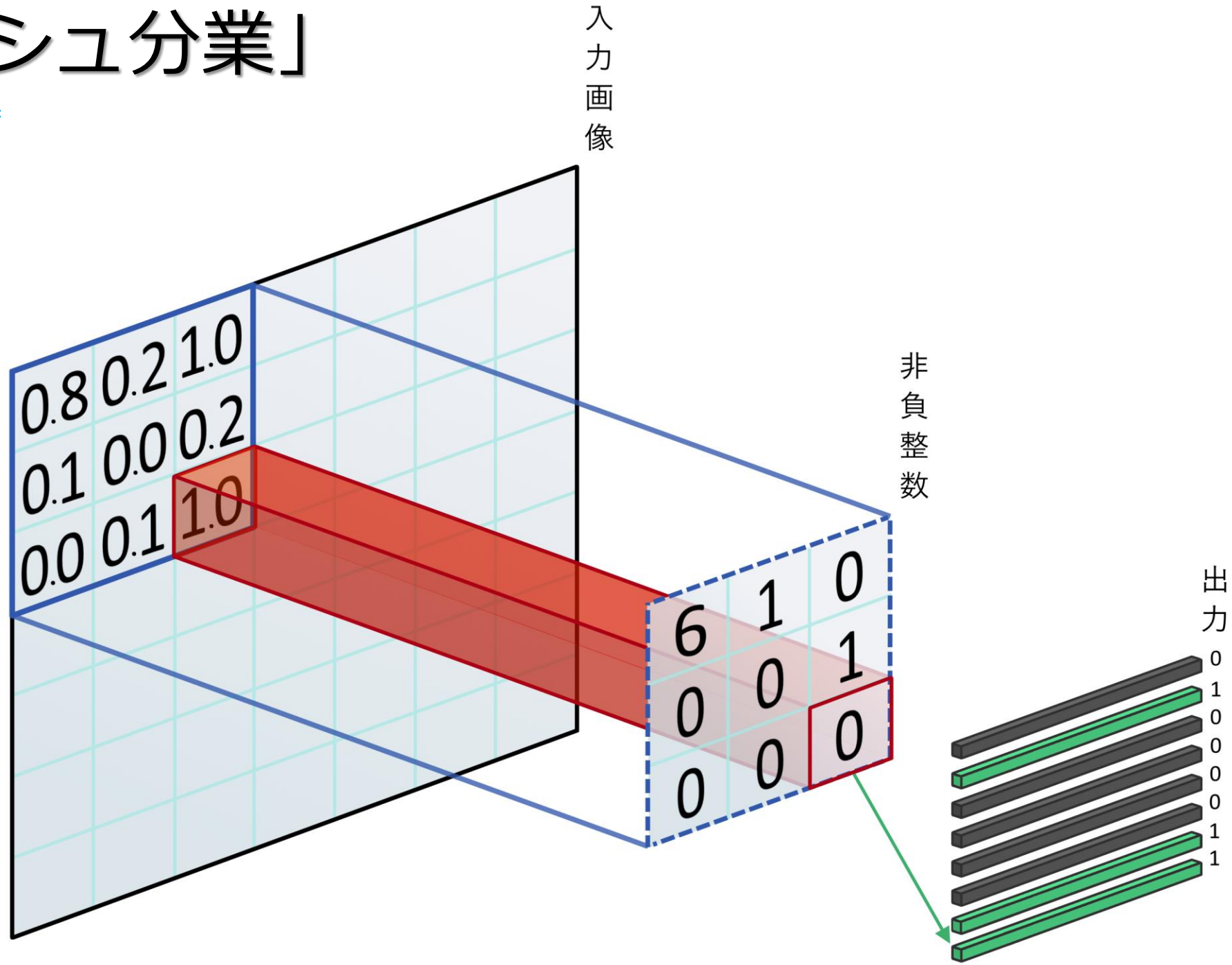
5-3.「ハッシュ分業」



5-3.「ハッシュ分業」



5-3.「ハッシュ分業」

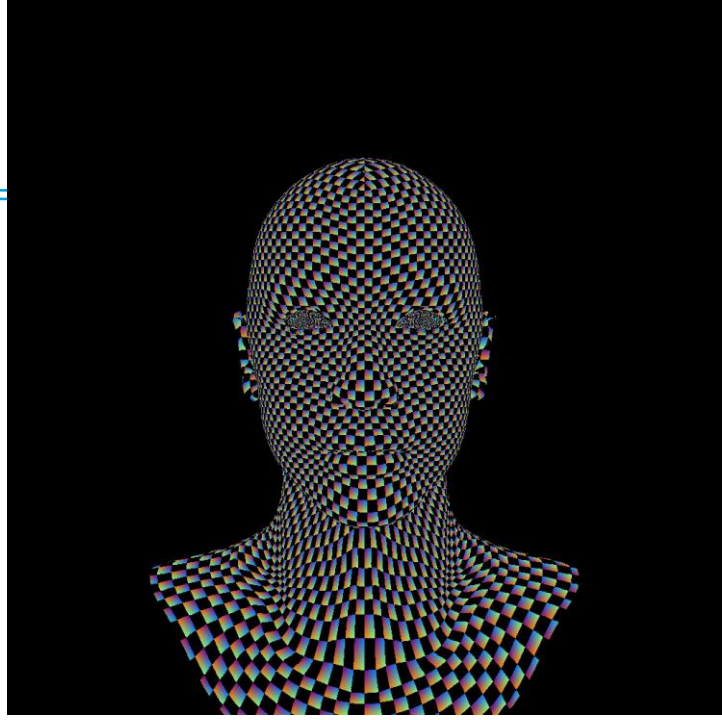


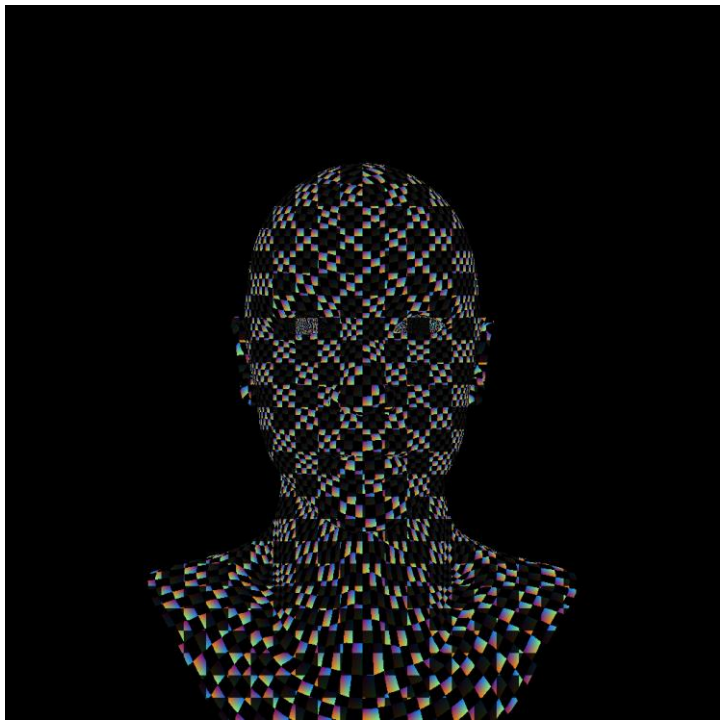
5-3. 独立のタイル

- ピクセルが独立でも、
  神経回路の重みは共有

- 効率のために１６ｘ１６ぐらいの
  ピクセル（１タイル）が協力して
  一緒に計算される

- 重みは共有メモリに保存される
  （Tensor Coreと好相性）





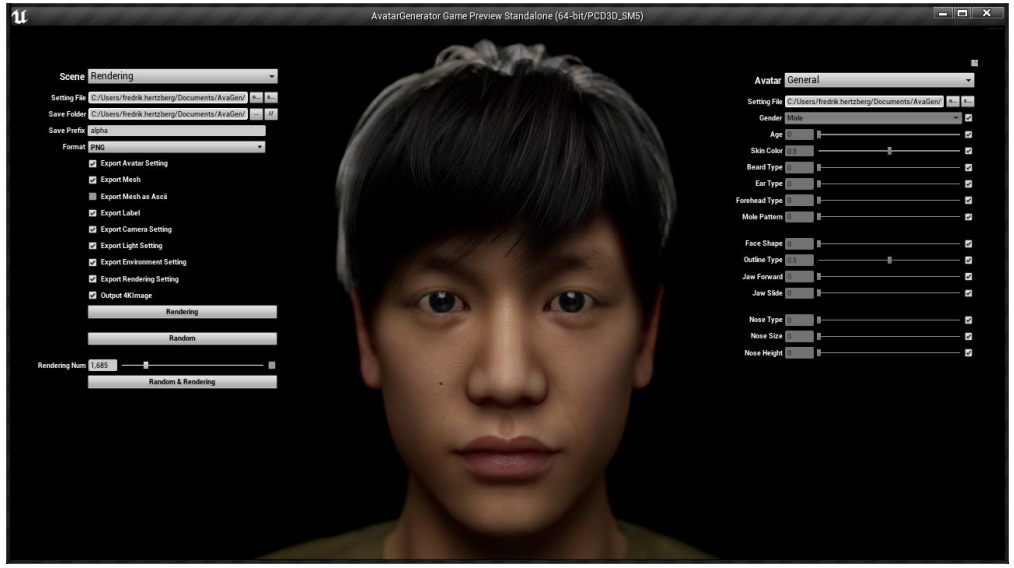
5-4. 高度に構造化した入力フォーマット

- ラスタライゼーションまたはレイトレーシングを使用
- 画面上のすべてのピクセルについて、マテリアル、表面のUV座標、法線、および光源情報を事前計算
  - ディファードシェーディングのGバッファのような内容
- 入力が構造化されているため、各ピクセルは独立しており、単一のシェーダ呼び出しで計算を完結できる
- これらを明示的に事前計算で行うことで、フレーム間の不安定性につながる潜在的な原因が1つ少なくなる

5-5. 教師データ

- 提案手法では、
  教師データ収集が大きな課題
  - 実写映像にはUV座標や法線などの情報がない
- 教師データもCGで生成
  - アーキテクチャの検証のため
  - 実写映像から学習するアイデアについては最後に紹介
- 弊社の顔画像生成ツール
  (Avatar Generator)を使用
  - パラメータを自在に変更して
    様々な顔画像を容易に生成可能
    - 異なる人種、年齢
    - 眉毛、目、鼻、口、耳の各パーツの 大きさ、位置
    - 髪の長さ、ウェーブ、ボリューム等
    - メイクアップ
    - フェイシャルパラメータによる様々な表情
    - 光源やカメラの位置、レンズの設定

Avatar Generator
Tools for creating various 3D avatars



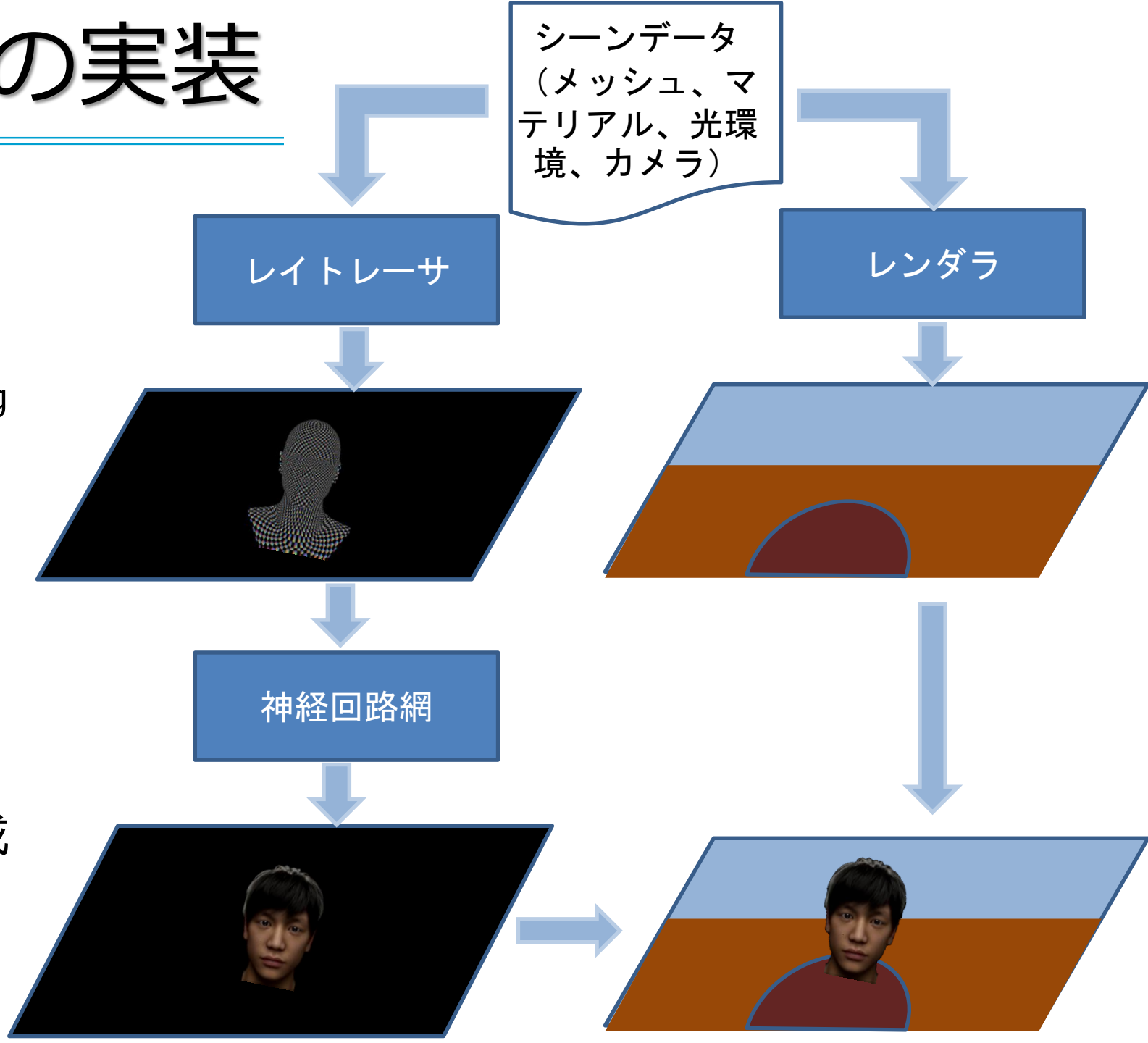
https://www.siliconstudio.co.jp/products-service/ai-cgdata/avatar/

5-6. トレーニングについて

- タイル内容によってバイナリファイルを別フォルダに
  - フォルダ毎に独立に学習させる
- Batch Normalizationの影響で、
  高解像度出力にはメモリ使用量が大きくなる問題
  - ハッシュ分業の開発はある程度その問題の対策
  - 分業の神経回路網は重みが比較的少ないため、
    メモリのネックがなくなる

5-7. システムの実装

- 各ピクセルにおいて
データを収集し、
神経回路網に渡す
入力フォーマットに変換
  - Nvidia の Vulkan Ray-Tracing Extension を利用して実装
- 神経回路網のコンピュート
シェーダを実行、
RGBの色情報を出力
- 顔以外の部分は
従来どおりのレンダリング
- 最後に2つの結果画像を合成



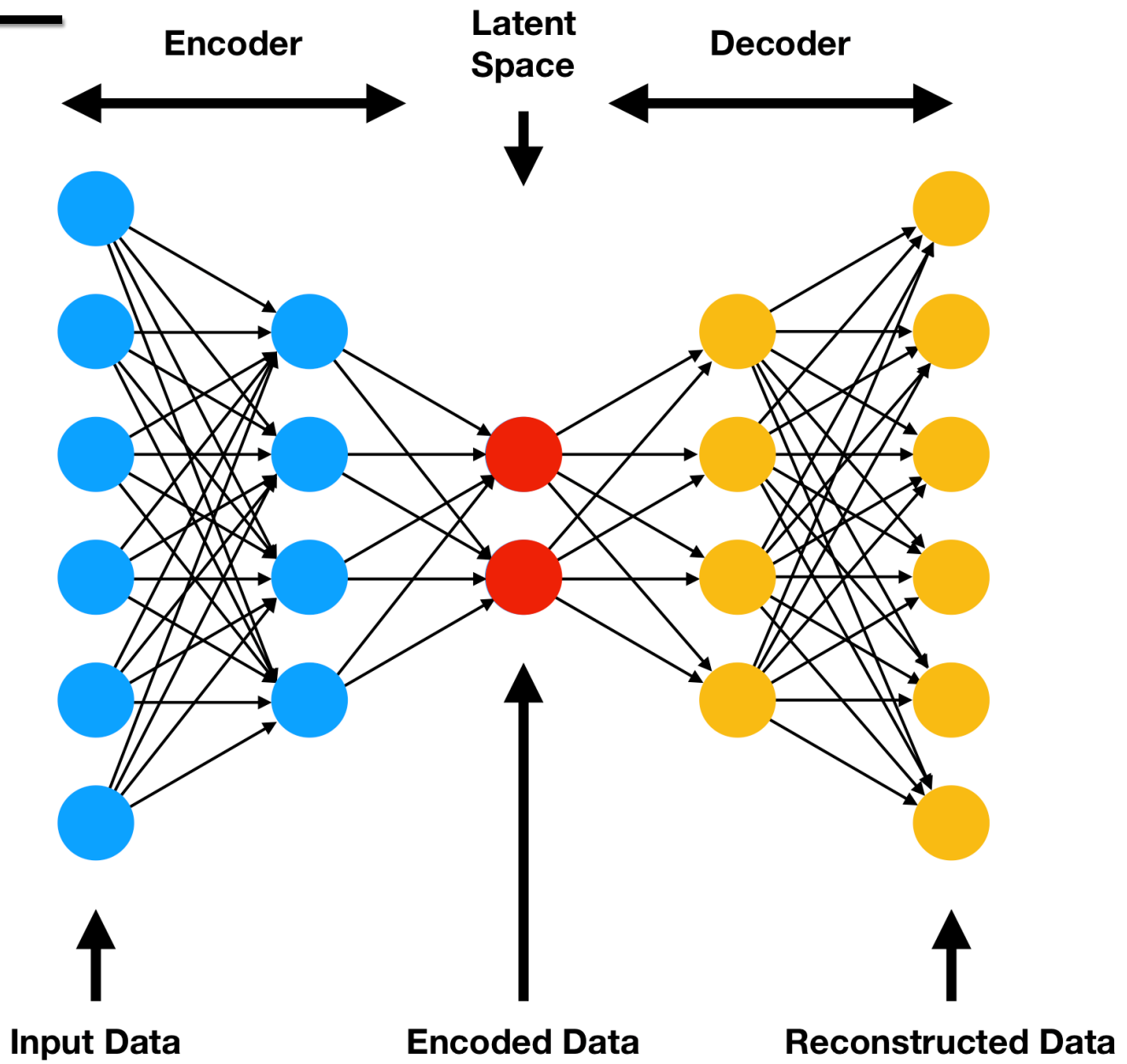
講演内容

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
    1. オートエンコーダ
    2. GAN
7. 今後の予定

オートエンコーダー

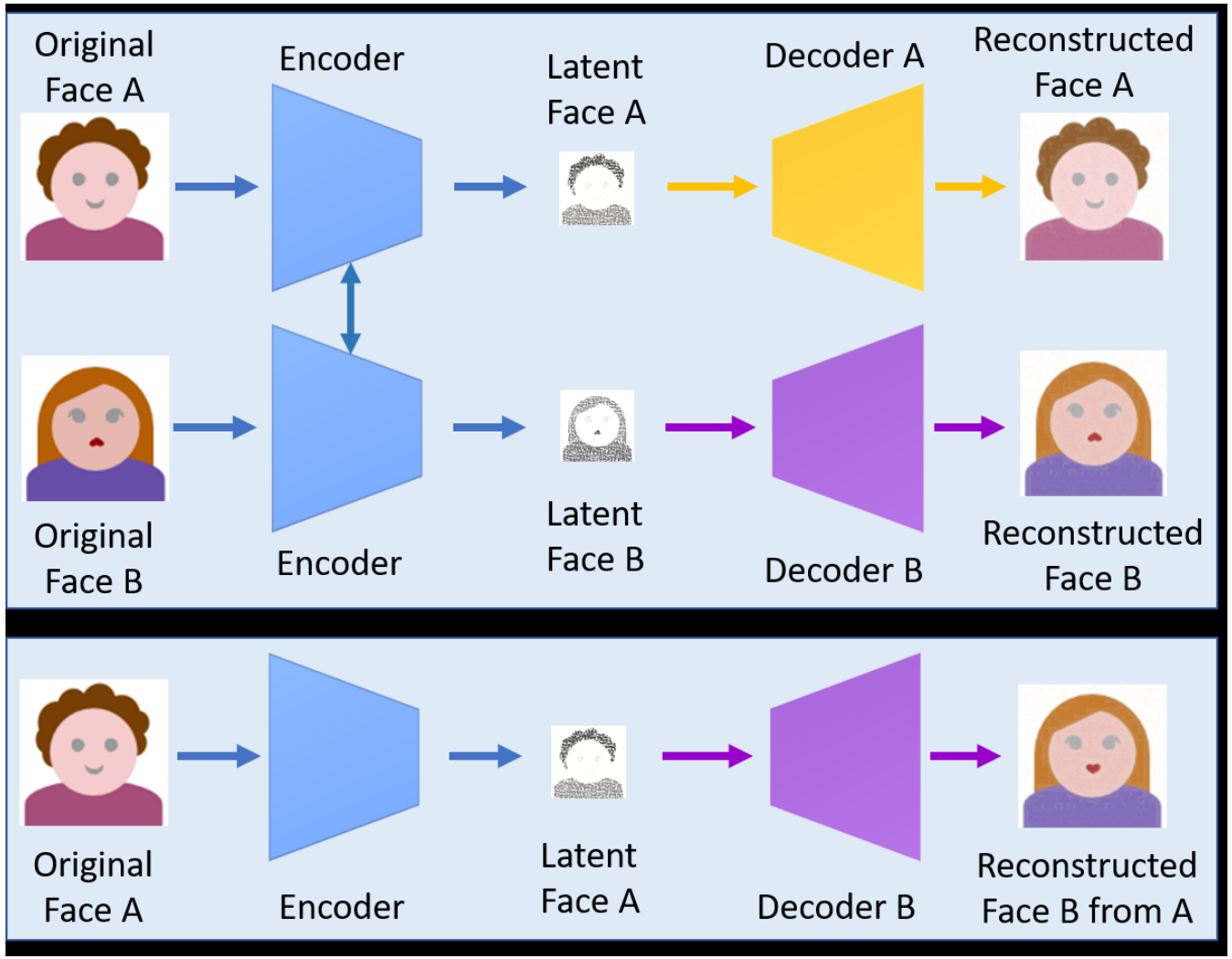
- 画像を一度低次元に
  エンコードした後、
  再度復元(デコード)し、
  入力と出力が一致する
  ようにトレーニングする

- ドメインに特化した
  高次の空間の内容を
  少ない次元で表現できる



共有潜在空間 オートエンコーダー

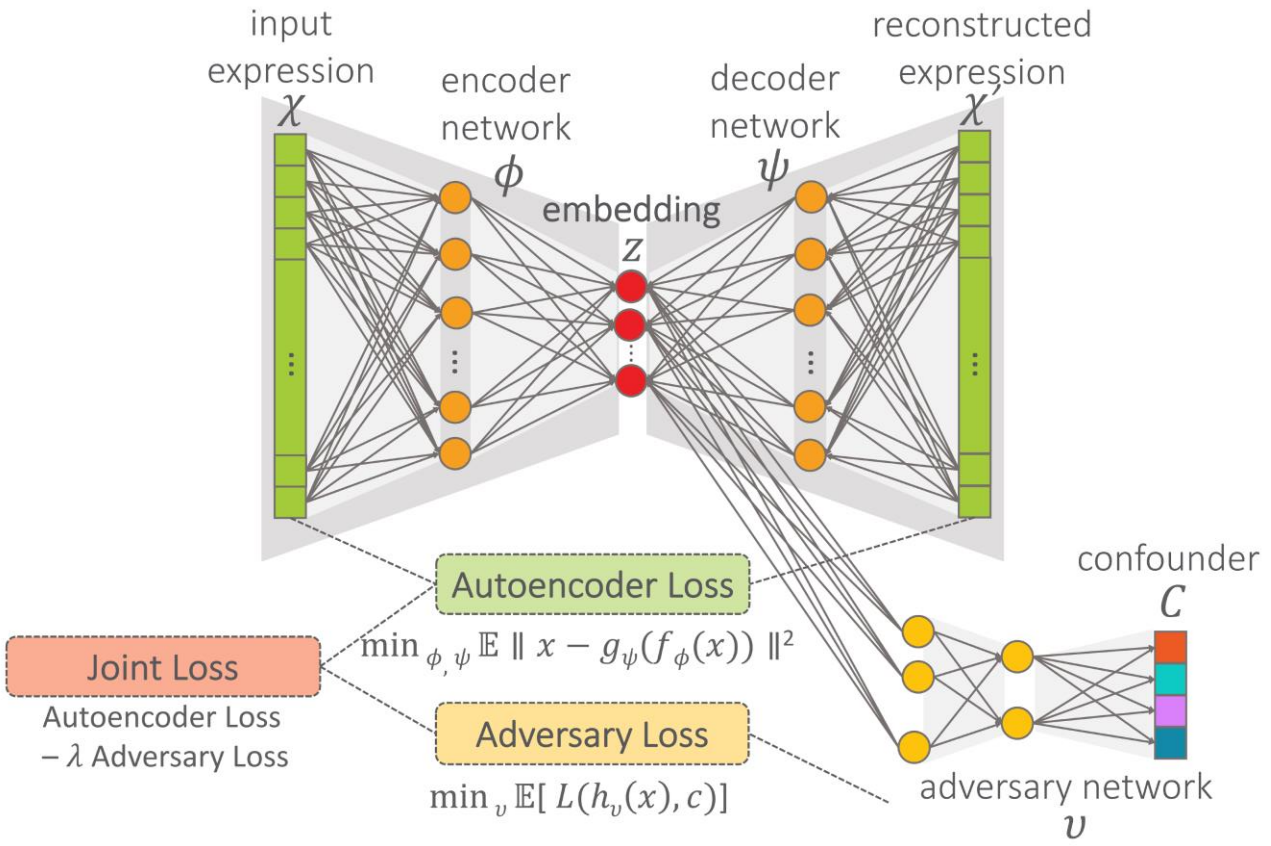
- エンコーダは複数の人間で共有
- デコーダは共有しない
- 別の人のデコーダを使えば、ある人の顔を別の人の顔に変えることができる
- ディープフェイクで使用されており、製品レベルの画質に近づいている
- フォトリアリスティックな入力が必要



Thanh Thi Nguyen, Cuong M. Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen and Saeid Nahavandi. "**Deep Learning for Deepfakes Creation and Detection**“, 2019

敵対的オートエンコーダ

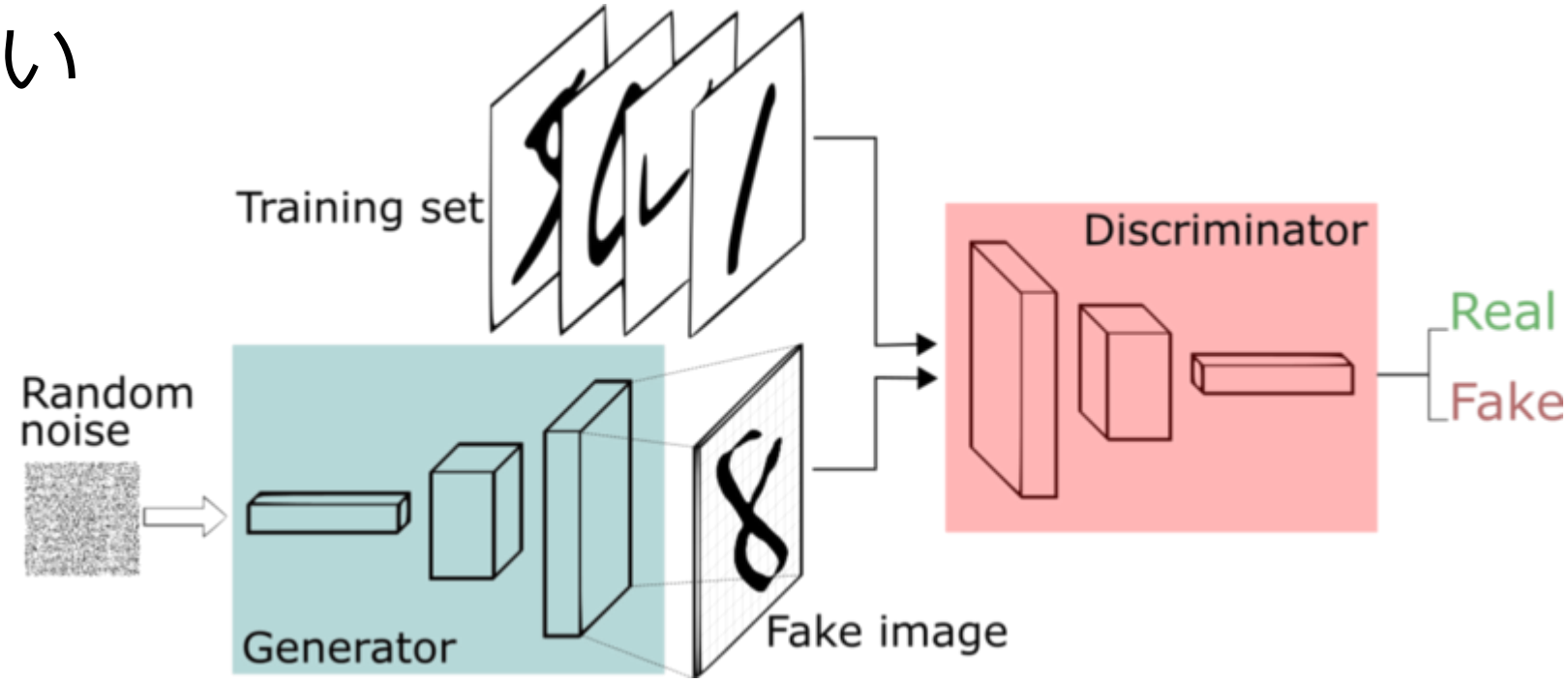
- オートエンコーダの潜在空間を
  もっと使いやすい形にしたい
- 別のニューラルネットワーク
  「敵対的な識別器」を
  損失関数に用いる
- 潜在空間の分布を制御できる
- 本研究では、
  教師データ生成に用いる予定



Ayse B. Dincer et al., "**Adversarial Deconfounding Autoencoder for Learning Robust Gene Expression Embeddings**", 2020

GAN（敵対的生成ネットワーク）

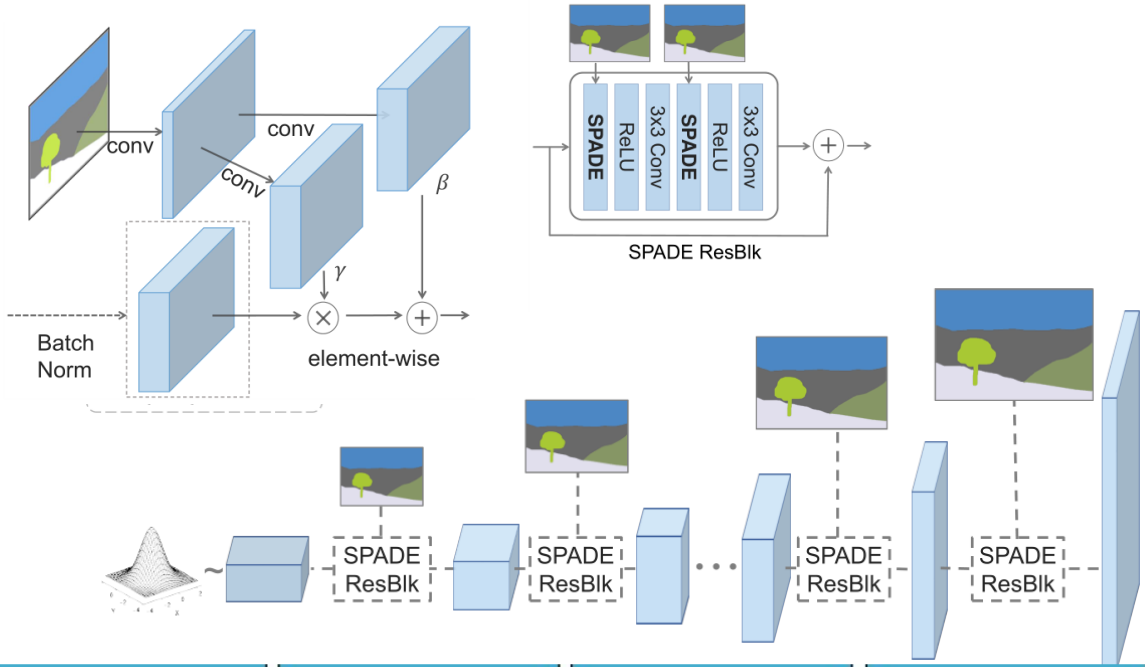
- 敵対的オートエンコーダと似ているが、
  潜在空間ではなく生成器の出力に識別器を適用する
- 生成器への入力はランダムな分布
- 出力の制御が難しい

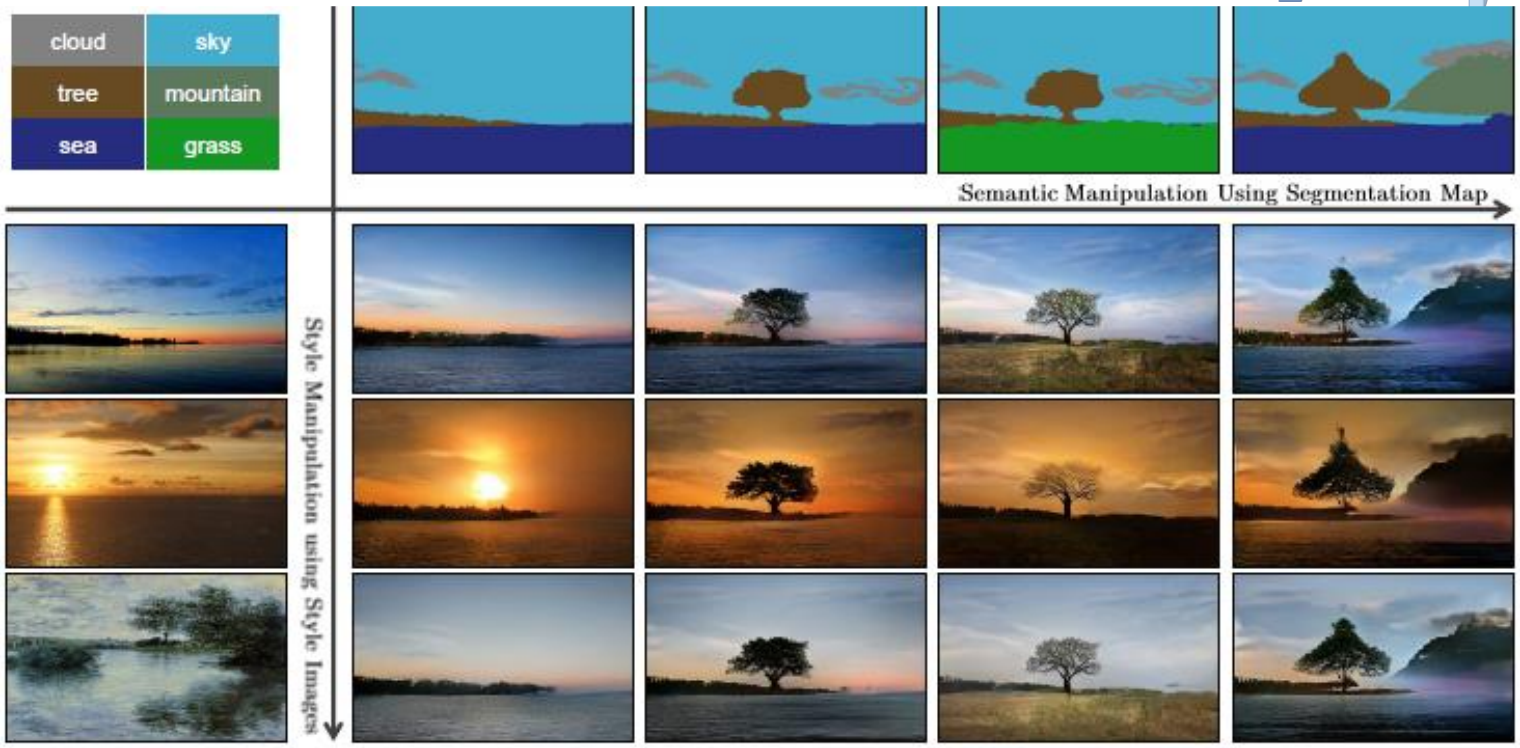


https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

条件付きGAN

- 出力を制御できるGAN
  - 生成器の各階層における Batch Normalizationの後に 整数マスク画像を入力
  - 条件付けすることで、 望ましい出力が得られる
- Batch Normalization により入力の構造が 壊れてしまう問題を 解決した





Taesung Park et al., “**Semantic Image Synthesis with Spatially-Adaptive Normalization**“, 2019

講演内容

Silicon Studio

1. 前置き
2. 背景
3. 既存の神経回路網の非効率さについて
4. リアルタイム描画での条件による効率化の機会
5. 非効率さを解決するアーキテクチャの提案
6. 重要な論文のアーキテクチャの紹介
7. 今後の予定
    1. まとめ
    2. 今後の予定

7-1. まとめ

- リアルタイムAI描画に至る道
  - 現状の神経回路網によるAIはリアルタイム描画に不向き
    - 実行速度が遅い
    - 顔の形や方向、ライティングなどを直接指定できない
  - 神経回路網の構成とGPUの非効率を考察
    - データ依存性、チャンネル数の問題
  - リアルタイム描画に特化したAIを提案
    - メッシュやライティング情報を構造化した入力フォーマットに変換
    - 特定の描画に特化した神経回路網を多数学習し、ハッシュによって使い分ける
    - 神経回路網はハードの制約に収まるようにする
  - 人工的に生成した学習データで検証
    - 実写映像からの学習データ生成が今後の課題

7-2. 今後の予定

Silicon Studio

- 複数光源対応
- GPGPU最短経路探索を用いた3Dカメラ点群データのUV生成
- 敵対的オートエンコーダによるUV座標
- 速度向上
- 条件付きGANによるカスタムメイドのトレーニングデータ

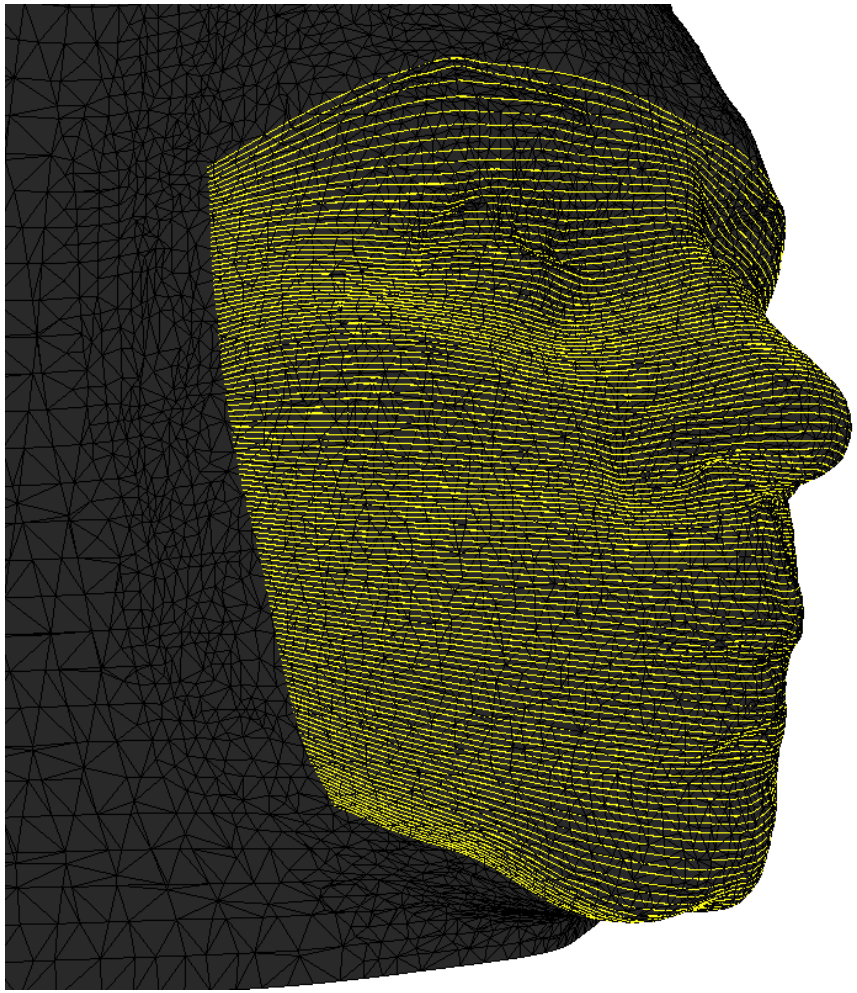
GPGPU最短経路探索を用いた3Dカメラ点群データのUV生成

Silicon Studio

- トレーニングデータ
  生成プログラムによって、
  顔の正しいUVマッピングが
  提供されたデータで学習を行ったが、
  可能ならば実写映像を使いたい

- 3Dカメラならば、顔の実写映像と
  同時に3D形状データも取得できる

- 3D点群データに対し、
  GPGPU最短経路探索を用いて
  UV生成する実装を行っている



敵対的オートエンコーダによるUV座標

- 2D映像のみからUV生成できれば、
  教師データがより集めやすくなる
- 顔の潜在的な変数空間を生成し、それから顔を再作成する
  敵対的なオートエンコーダを想像してほしい
- 識別器によって、トレーニングデータの顔から定義された
  正しいUVマッピングにするように、
  潜在的な空間を強制することができるはず
- これはあくまでトレーニングデータを生成するための
  ものであり、リアルタイムに実行できる必要はない

End

ご清聴ありがとうございました

質問

Special Thanks

- Avatar Generatorチームの皆さん
- Bosphorus 3 Dチーム

Savran, N. Alyüz, H. Dibeklioğlu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun, “**Bosphorus Database for 3D Face Analysis**,” The First COST 2101 Workshop on Biometrics and Identity Management (BIOID 2008), Roskilde University, Denmark, 7-9 May 2008

Contact

Silicon Studio

- Fredrik Herzberg
  fredrik.hertzberg@siliconstudio.co.jp