2025 年度　アジャイルワーク 報告書

24G1089　武本 龍

2025 年 12 月 8 日

1　はじめに

近年、ディープラーニングをはじめとする AI 技術は急速に発展しており、その学習や推論には大規模な計算資源や電力を必要とすることが一般的となっている。しかし、組込み機器や IoT デバイスのような小型環境では、CPU 性能やメモリ容量、電力供給といったリソースが大幅に制限されるため、従来の手法をそのまま適用することは困難である。そこで本研究では、既存の Arduino Uno R4 WiFi 環境において可能な限り高性能なニューラルネットワーク推論を実現することを目的とし、学習データの軽量化やモデルの圧縮を含む最適化手法を検討し、特に、学習データを圧縮した上で推論に必要な情報を保持できるか、また限られた計算能力の中で最大限の推論精度を引き出せるかを検証し、その実験手法および得られた知見について報告する。

2　実験の概要

図 1に，本実験の全体構成を示す．

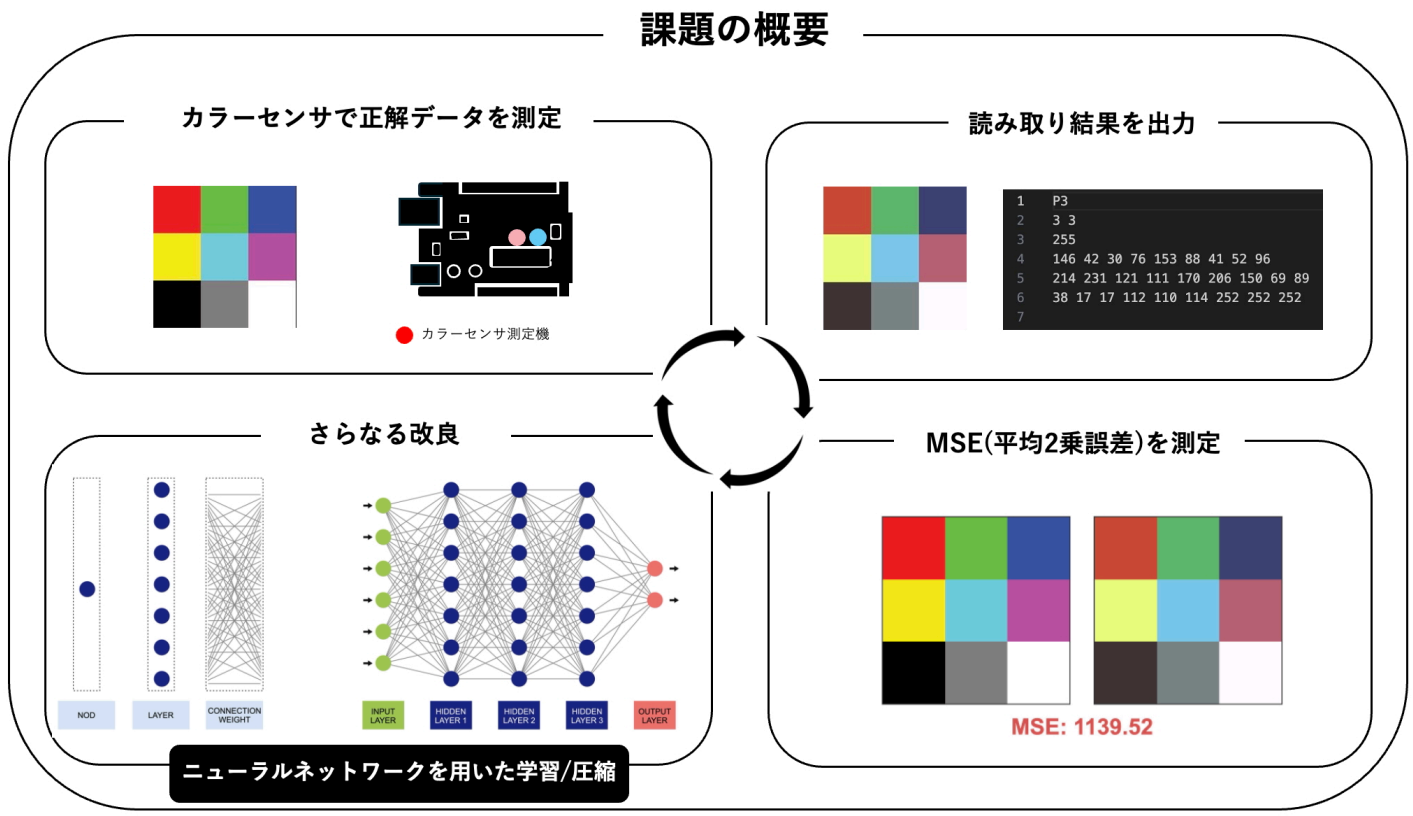

図 1: 実験の全体構成

本実験では，Arduino Uno R4 WiFi を用いてカラーセンサーを構築し，授業内で配布された 3×3 のカラーチャートを測定する．測定後，ニューラルネットワークを用いた補正処理により，平均二乗誤差（MSE）の低減を目指す．前章で述べた 2 つの検証項目に対応し，本実験では以下の観点から評価を行う．

第一の検証として，ニューラルネットワークのパラメータ圧縮によるメモリ効率の改善を検討する．具体的には，隠れ層のデータ圧縮手法を導入し，推論に必要な情報を保持しながらメモリ使用量をどの程度削減できるかを評価する．

第二の検証として，限られた計算資源の中での推論性能の最大化を検討する．測定したカラーチャートと基準カラーチャート間の MSE を評価指標とし，ニューラルネットワークによる学習を活用して推論精度の向上を図る．また，predict 関数の処理時間を計測し，実行速度についても評価する．

2.1 平均二乗誤差（MSE）

平均二乗誤差（MSE）は，2 枚の画像の対応するピクセル位置における輝度差の 2 乗の平均値として定義され，以下の式で表される．

$$\mathrm{MSE} = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(I(i,j) - K(i,j) \right)^2 \tag{1}$$

ここで，I と K は比較対象となる 2 枚の画像，$m \times n$ は画像サイズを表す．

本実験では RGB 画像を扱うため，各ピクセルについて R，G，B チャネルごとの差の 2 乗を加算し，チャネル数で除した値を MSE として算出する．具体的な計算式を以下に示す：

$$\mathrm{MSE} = \frac{(r_{\mathrm{diff}})^2 + (g_{\mathrm{diff}})^2 + (b_{\mathrm{diff}})^2}{3} \tag{2}$$

3　実験理論

本章では，ニューラルネットワークを用いた読み込み品質の改善手法について述べる．

3.1　ニューラルネットワークによる品質改善

サンプル画像と測定画像との誤差を最小化するため，本研究ではニューラルネットワークを用いた学習手法を導入する．特に，低リソース環境である Arduino 上での推論実行を前提とし，モデルの圧縮および疎化手法を併用しながら性能の最適化を図る．

3.1.1　ネットワーク構造

本実験で採用したモデルは，全結合型（Fully Connected）の 4 層ニューラルネットワークである．入力層は RGB の 3 次元，2 つの隠れ層はそれぞれ 40 次元（40 個のニューロンが全結合），出力層は再び RGB の 3 次元とした．各ニューロンにおける推論は以下の式で表される．

$$y = x_1 w_1 + x_2 w_2 + \cdots + x_{n-1} w_{n-1} + b \tag{3}$$

ここで，w_i は重み，b はバイアスを表す．算出された値は ReLU 活性化関数を通過し，次層へ伝搬される．

モデルは PyTorch により定義・訓練し，L1 正則化を導入して重みの疎化を促進した．訓練後の重みおよびバイアスは C++ 配列としてエクスポートし，Arduino 上で実行可能な形式に変換した．さらに，本研究では隠れ層の次元拡張性を検証するため，40 次元に加えて 80 次元，120 次元のモデルについても実験を行った．

3.2　隠れ層の圧縮手法

3.2.1　圧縮の必要性

Arduino Uno R4 WiFi のようなマイクロコントローラでは，利用可能なメモリ（SRAM 約 32 KB，Flash 256 KB）が極めて限定的である．隠れ層の次元を増加させると，重みおよびバイアスの格納や推論演算に必要なメモリが不足する可能性が高い．

例えば，隠れ層次元を 40 から 70 以上に拡張するとパラメータ総数は急増する．

- 40 次元：約 1,923 パラメータ
- 70 次元：約 3,000 パラメータ以上

float32 形式では 10 KB を超える場合，Arduino 上での動作が困難となる．したがって，隠れ層の次元を拡張しながら高い推論性能を維持するには，重みデータの圧縮が不可欠である．

本研究では，この制約を克服するため，まず CSR 形式による疎行列圧縮を適用し，さらなる圧縮が必要な場合には量子化を追加適用する方針とする．以下では，各手法の原理と特徴について述べる．

3.2.2　プルーニング（Pruning）

プルーニングとは，閾値以下の重みを 0 とみなし接続を削除することで，モデルを疎行列化する手法である．Han ら [2] は，重要接続の学習，閾値による削減，再訓練を繰り返すことで，大規模モデルを 10 倍以上圧縮できることを示した．

本研究では，L1 正則化により重みを 0 に近づけた後，閾値 0.01 でプルーニングを実施する．この処理により多くの重みが 0 となり，後述する疎行列表現による効率的な格納が可能となる．

■メリット

- パラメータ削減：非ゼロ率を大幅に低下可能（40 次元モデルで 1,923 から 200 以下も実現可能）
- 計算量削減：不要な演算が減少し推論が高速化
- 精度維持：軽微な削減であれば再訓練により精度回復が可能

■デメリット

- 再訓練なしでは精度低下が生じやすい
- インデックス保存など疎行列特有のオーバーヘッドが存在
- プルーニング量の決定に追加の探索が必要

3.2.3 CSR（Compressed Sparse Row）形式

CSR 形式は，非ゼロ要素を行単位でまとめて保存する疎行列表現であり，値配列（data），列インデックス（indices），行ポインタ（indptr）で構成される．プルーニングにより疎化された重み行列を効率的に格納でき，行方向のアクセスが高速なため Arduino 上での推論に適している．

本研究では，CSR 形式を第一の圧縮手法として採用する．L1 正則化およびプルーニング後の重み行列を CSR 形式に変換することで，SRAM 32 KB の制約下でも動作可能なモデルへ圧縮する．

■メリット

- 行方向のアクセスが高速で推論が効率的
- インデックス管理が最適化されメモリ効率が高い
- 固定長配列で Arduino 実装に適する
- 非ゼロ率に応じて 3〜10 倍のメモリ削減が可能

■デメリット

- 変換時にソートが必要
- 列方向アクセスは非効率

なお，疎行列表現としては COO（Coordinate）形式も存在し，実装が容易という利点があるが，行列ベクトル積の計算効率が CSR 形式より劣るため，本研究では CSR 形式を採用した．

3.2.4 量子化（Quantization）

量子化とは，ニューラルネットワークの重みや活性化値を 32 ビット浮動小数点から 8 ビット整数へ変換し，メモリ削減と計算高速化を図る手法である．Jacob ら [1] が提案したアフィン変換

$$r = S(q - Z) \tag{4}$$

を用いることで，推論時の整数演算が可能となる．ここで，S はスケール，Z はゼロポイントを表す．

本研究では，CSR 圧縮のみでは十分なメモリ削減が得られない場合の追加手法として量子化を位置づける．CSR 形式で格納された重みに対して float32 から int8 への変換を適用することで，さらに約 4 倍のメモリ削減が期待できる．

■メリット

- メモリ削減：32 ビットから 8 ビットへの変換により約 4 倍削減
- 計算高速化：整数演算により浮動小数点演算より低レイテンシ
- 精度維持：量子化感知訓練（QAT）を用いることで MSE をほぼ維持可能

■デメリット

- モデル規模が小さい場合，チャネルごとの差により誤差が生じやすい
- スケール・ゼロポイントの計算や int32 蓄積処理など実装がやや複雑
- 訓練時に fake quantization ノードの挿入が必要

3.2.5 圧縮手法の比較と本研究での適用方針

表 1に，各圧縮手法の特徴を示す．

表 1: 重み圧縮手法の比較

手法	メモリ削減率	計算効率	実装難易度	Arduino 適合性
CSR 形式	3–10 倍	高	中	高
量子化	約 4 倍	高	中	高
CSR+ 量子化	12–40 倍	高	高	高

本研究では，CSR 形式による疎行列圧縮を第一の手法として適用し，推論品質への影響を評価する．CSR 圧縮により MSE が許容範囲内に収まる場合はそのまま採用し，さらなるメモリ削減が必要な場合や隠れ層次元の拡張を行う場合には，CSR 圧縮モデルに対して量子化を追加適用する．この段階的なアプローチにより，推論精度とメモリ効率のバランスを最適化することを目指す．

4　システムの構成

本章では，実験に用いたシステムのプログラム構成および配線について述べる．

4.1　プログラムの構成

本システムのプログラムを以下に示す．

```
% TODO： コードを記載
```

Listing 1: メインプログラム

```
% TODO： コードを記載
```

Listing 2: 補助関数

4.2　ハードウェア構成

本節では，実験で使用した配線および回路構成について述べる．表 2に Arduino のピン割り当て，表 3に使用機材一覧，図 2に回路図を示す．

表 2: Arduino のピン割り当て

ピン	機能	説明
D2	タクトスイッチ（赤）	最大値・最小値の切り替え
D3	タクトスイッチ（青）	RGB データの読み取りトリガー
A0	照度センサー	フォトトランジスタからのアナログ入力

表 3: 使用機材一覧

機材名	型番	個数
炭素皮膜抵抗 330Ω	-	3
炭素皮膜抵抗 3.3$k\Omega$	-	1
炭素皮膜抵抗 10$k\Omega$	-	2
RGB フルカラー LED	OSTA5131A	1
照度センサー（フォトトランジスタ）	NJL7302L-F3	1
タクトスイッチ (赤・青)	1273HIM-160G-G	2
ジャンパーワイヤ	BBJ-65	13
マイコンボード	Arduino UNO R4 WiFi	1
ブレッドボード	-	1

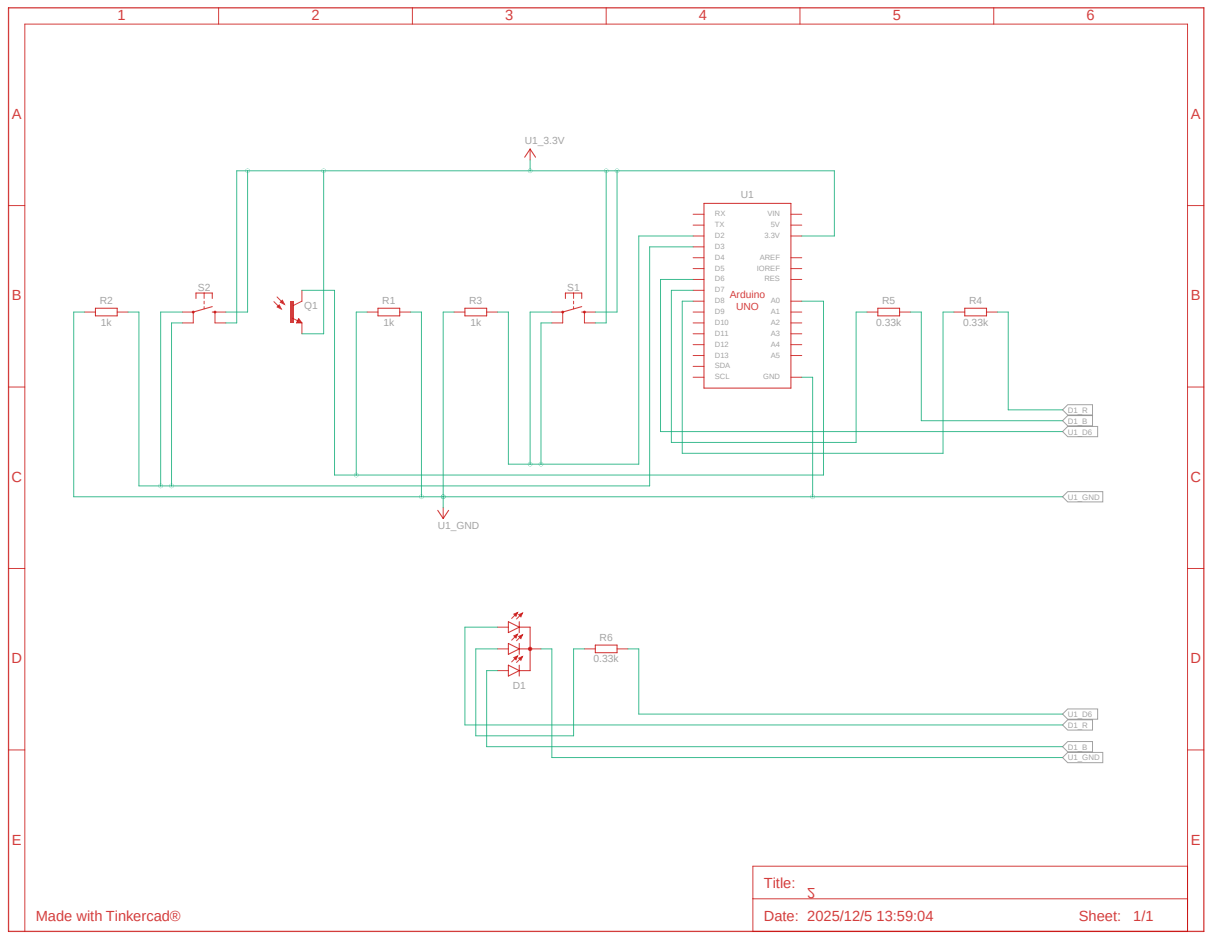
1
2
3
4
5
6
A
B
C
D
E
U1_3.3V
U1
Arduino
UNO
RX
TX
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
SDA
SCL
VIN
5V
3.3V
AREF
IOREF
RES
A0
A1
A2
A3
A4
A5
GND
R2
1k
S2
Q1
R1
1k
R3
1k
S1
R5
0.33k
R4
0.33k
D1_G
D1_B
U1_D6
U1_GND
U1_GND
R6
0.33k
D1
U1_D6
D1_R
D1_B
U1_GND
Made with Tinkercad®
Title:
S
Date: 2025/12/5 13:59:04
Sheet: 1/1

図 2: 回路図

5　実験方法

以下に，実験方法を示す.

シリアルモニタ経由で画像サイズを入力（配布したカラーチャートだと「3」⊠ボタンを押すごとに、RGB データを読み取る⊠紙をスライドさせて、すべての画素を読み取る（配布したカラーチャートだと、計 9 回）⊠シリアルモニタに PPM 形式で画像データをテキスト出力

読み込み品質の評価は，以下の手順で実施する.

1. **2 つのボタンプログラムによるカラーチャートの読み取り**
   事前に黒と白のサンプルを測定し，最小値・最大値を基準として読み込み精度をキャリブレーションする．これにより，センサの感度を調整し，安定した RGB 値を取得する.
2. **PPM 画像出力**
   前回授業で配布された 3×3 のカラーチャートをセンサで読み取り，取得した RGB データを PPM 形式の画像ファイルとして出力する．この画像は，後続の品質評価に用いる.
3. **平均二乗誤差（MSE）の計算**
   出力した PPM 画像と基準画像の品質を，専用の MSE 測定ソフトウェアで評価する.

6　実験方法

本節では，重み圧縮手法として CSR（Compressed Sparse Row）形式を第一優先とし，次に CSR 圧縮モデルに対する量子化を適用してさらなる圧縮を検討する．実験は，隠れ層次元 40 から開始し，「非圧縮」「CSR 圧縮」「CSR+ 量子化」の MSE を比較して品質影響を評価する．全体フローは，PyTorch による訓練・圧縮生成，Arduino Uno R4 WiFi へのデプロイ，センサーデータによる推論・MSE 計算である．目標は，MSE 相対誤差を 5 モデル訓練: PyTorch で ColorNet（入力 3→ 隠れ $d_1 \to d_2 \to$ 出力 3，$d_1 = d_2 = 40$ 初期）を訓練（epochs=50,000, Adam lr=0.01, $\lambda_{L1} = 10^{-6}$）．L1 正則化で非ゼロ率 50 %以下を目指す．圧縮生成: プルーニング（閾値 0.01）後，CSR 変換（SciPy）し，model$_{p}arameters.h$ エクスポート．$Arduino$ デプロイ：スケッチで整数 $-$ $onlyforward$ 実装（$PROGMEM$ 保存）．RGB 入力→ 推論 →PPM 出力．評価基準: MSE/PSNR（PC ソフト）とレイテンシ（millis()）を測定．相対誤差<5 %，PSNR>25dB で次元拡張（40→60→100→200）．

結果は表**??**にまとめ，圧縮率・誤差・時間のトレードオフを分析する．

6.1　CSR 形式圧縮の実験方法

CSR を優先する理由は，L1 正則化による疎化を活かした高圧縮率（3-10 倍）と推論高速化（行指向 matvec）である．以下の手順で実施．

1. **モデル訓練（Python）：** PyTorch で ColorNet を訓練．L1 正則化で重みを疎化（非ゼロ率<50 %）．ReLU 活性化使用．
2. **プルーニングと CSR 生成:** 閾値 0.01 で重みを 0 化（‘torch.abs(w) $<$ 0.01‘）．SciPy の ‘sparse.csr$_{m}atrix$‘で CSR 変換（$data, indices, indptr$ を $NumPy$ 出力）．C 配列として ‘$model_{p}arameters.h$‘生成（例：‘$constfloatcsr_{w}1_{d}ata[NNZ] = ...;$‘）．
2. **Arduino 実装:** CSR matvec 関数を実装（‘for j=indptr[i]; j<indptr[i+1]; y[i] += data[j] * x[indices[j]];‘）．センサー RGB 入力 →CSR forward（ReLU 統合）→ 出力 RGB→PPM 出力．PROGMEM で Flash 保存．
3. **評価:** PPM と基準画像の MSE/PSNR 計算．非圧縮 vs CSR の相対誤差を算出（目標: $<3\%$）．40 次元で確認後，次元増加・再訓練．

6.2 CSR+ 量子化による追加圧縮の実験方法

CSR 後，8-bit 量子化（uint8 重み/活性化，int32 バイアス）を適用し，メモリをさらに 4 倍削減．論文 [1] のスキーム（$r = S(q - Z)$）を基に，CSR の疎性を保ち 200 次元対応を目指す．量子化感知訓練（QAT）で精度を維持．

1. **量子化感知訓練（Python）：** 訓練グラフに fake quantization 挿入（$\hat{r} = ((r; a, b)/S) \cdot S + Z$, $S = (b - a)/255$，範囲 $[a, b]$ を EMA 学習）．初期 50k ステップで量子化無効化，ReLU6 で範囲安定化（epochs=50,000）．
2. **量子化実行（Python）:** CSR データ（float32）を uint8 へ変換（$q = ((r - Z)/S)$, scale=$max(|r|)/127$）．バイアスは $S_w S_a$ スケールで int32．multiplier $M = S_w S_a / S_o$ を fixed-point オフライン計算．indptr/indices を int16 圧縮，‘model$_parameters.h$‘更新．
2. **Arduino 実装:** matvec 内で uint8 乗算 +int32 蓄積 + スケーリング（gemmlowp 風）．ReLU を int8 clamp(0,127)．EloquentTinyML ライブラリで int8 対応．
3. **評価:** CSR vs CSR+ 量子化の MSE 比較（目標: 追加誤差<2%）．レイテンシ短縮率測定．100 次元で PSNR>25dB 確認後，200 次元挑戦．

これらの実験により，CSR 基盤上で量子化を積層し，MSE 低減・時間短縮を両立．プルーニングとの相乗効果で，非ゼロ率低減と整数演算効率を検証する．

7 実験方法

本節では，重み圧縮手法として CSR（Compressed Sparse Row）形式を第一優先とし，次に CSR 圧縮モデルに対する量子化を適用してさらなる圧縮を検討する．実験は，隠れ層次元 40 から開始し，「非圧縮」「CSR 圧縮」「CSR+ 量子化」の MSE を比較して品質影響を評価する．全体フローは，PyTorch による訓練・圧縮生成，Arduino Uno R4 WiFi へのデプロイ，センサーデータによる推論・MSE 計算である．

目標は，MSE 相対誤差を 5 %以内に抑えつつ，次元を 100-200 へ拡張可能とする．これにより，L1 正則化による疎化を活かし，メモリ制約下での画像復元品質向上と読み取り時間短縮を実現する．

7.1 全体実験フロー

データ準備: 授業配布の 3×3 カラーチャート RGB 値を訓練データ（9 サンプル，ノイズ付加）とし，基準画像を評価用とする．センサー（例: AS7341）で実測データを収集．モデル訓練: PyTorch で ColorNet（入力 3→ 隠れ $d_1 \to d_2 \to$ 出力 3，$d_1 = d_2 = 40$ 初期）を訓練（epochs=50,000, Adam lr=0.01, $\lambda_{L1} = 10^{-6}$）．L1 正則化で非ゼロ率 50 %以下を目指す．圧縮生成: プルーニング（閾値 0.01）後，CSR 変換（SciPy）し，model$_parameters.h$ エクスポート．$Arduino$ デプロイ : スケッチで整数 $-$ $onlyforward$ 実装（$PROGMEM$ 保存）．RGB 入力→ 推論 →PPM 出力．評価基準: MSE/PSNR（PC ソフト）とレイテンシ（millis()）を測定．相対誤差<5 %，PSNR>25dB で次元拡張（40→60→100→200）．

結果は表**??**にまとめ，圧縮率・誤差・時間のトレードオフを分析する．

本節では、

参考文献

[1] B. Jacob et al., “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,” CVPR, 2018.

[2] Song Han et al., “Learning both Weights and Connections for Efficient Neural Networks,” NeurIPS, 2015.