東京理科大学

創域理工学部情報計算科学科

2024年度　卒業研究論文

深層学習におけるドロップアウトデザインの適用

学籍番号 6321120

横溝 尚也

2025年2月9日

目 次

1　はじめに

深層学習は急速に発展を遂げ, あらゆる分野で需要が拡大している. しかし, この技術が現在のような発展を遂げるまでには, 多くの障壁が存在した. 脳の神経回路を模倣したニューラルネットワークは 1960 年代に広がりを見せたが, 局所最適化や勾配消失問題などの課題により，他の分析手法に比べて実用性が乏しかった. そのなかで深層学習を世に広めたのは Geoffrey Everest Hinton である. Hinton は誤差逆伝播法を普及させ, 深層学習の基礎を築き, 2024 年にはノーベル物理学賞を受賞している. 深層学習の正則化技術としては，2012 年に Hinton ら [3, 5] によって提案されたドロップアウト法があり，現在でも広く使用されている． 正則化技術とは, 機械学習における過学習を抑制するために罰則を与えることでモデルの自由度を下げる手法であり, ドロップアウト法は，ニューラルネットワークのノードを無作為に不活性にする正則化手法である.

近年, 深層学習分野においてモデルのパラメータ数が急激に増加している. これは, より複雑な関係を持つ事象を予測し, より表現力の高い AI を構築するためである. 一方で，モデル規模の拡大に伴い，過学習の適切な認識と正則化技術の重要性もますます高まっている.

本研究ではドロップアウト法に代わる新たな正則化手法として, ドロップアウトデザインを導入する. ドロップアウトデザインは, 組み合わせ数学の一分野であるブロックデザインを応用した手法である. ブロックデザインは, 実験計画法, 暗号理論, ソフトウェアテストなど幅広い分野で応用されており, その中でもドロップアウトデザイン [1] は深層学習の多層ニューラルネットワークにおいて, ドロップアウトを行うノードを無作為に選択するのではなく, 規則的に選択するためのブロックデザインである.

深層学習にドロップアウトデザインを適用する研究は既に行われているが, 既存のドロップアウト法と同程度の正則化効果であり, いまだ優れている点は確認されていない [8]. 本研究の目的は, ドロップアウトデザインを用いることで, 事前に定められたノードを訓練し, ドロップアウト法に比べてテストデータに対する精度や, 精度の収束速度が優れているかどうかを検証することである. 現在の正則化技法よりもドロップアウトデザインを適用することでより速く精度が収束するのであれば, 深層学習のコスト削減や, より複雑なモデルの利用における課題が軽減されることが期待される.

本論文では, 深層学習の基礎知識を紹介した後, ドロップアウトデザインの詳細な解説および大規模なパラメータを持つデザインの構成手法について述べる. また, 大規模ネットワークにドロップアアウトデザインを適用する方法を解説する. 従来のドロップアウト法とドロップアウトデザインの汎化性能の違いや, 未知データに対する予測精度の収束速度などの違いについても言及し, ドロップアウトデザインの有効性を考察する.

2　深層学習とドロップアウト法

2.1　ニューラルネットワークによるモデル訓練

深層学習で利用される多層ニューラルネットワークは入力層と出力層, 多層の中間層から構成され, 隣り合う層のノード間はエッジで結合されており, 各エッジは重みと呼ばれるパラメータ w を持っている. この重みはノードからノードへと情報が伝達される強さを表しており, 深層学習の目的は訓練データを用いてこの重みを最適な値へ更新し, テストデータに対する予測精度を上げることである. 入力層で入力された値は, 各ノードにおいて値を計算しながら入力層から出力層にかけて伝播していく. 第 i 層目から第 $i+1$ 層目にかけてノードは以下のように計算される.

$$\mathbf{y_{i+1}} = f(\mathbf{W_i x_i} + \mathbf{b_i}) \tag{1}$$

$\mathbf{W}$ は層の重み, $\mathbf{b}$ はバイアス, f は活性化関数である.

訓練データに対して上記の計算をすることで出力層のノードの値を計算し, 最後に損失関数 $E(\mathbf{x})$ を最小化するように重みを更新する. 損失関数は平均2乗誤差や交差エントロピー誤差など, 扱う予測問題に合わせて適切な損失関数を選択する. この一連の訓練を効率的に学習させるため, データセットを**バッチ**と呼ばれるあらかじめ決められたデータサイズに分割し, 学習を行う. また, データセットすべてを1回訓練させる学習単位を**エポック**と呼び, このバッチ数やエポック数をハイパーパラメータとして最適な値に設定することでより良いモデル構築を目指す.

2.2　過学習に対するアプローチ

1エポックの訓練を繰り返すことで訓練に使用したデータに対する予測精度は次第に高くなり, より訓練データに適合したモデルが構築されていく. 前述のとおり, 関心のあるものは未知データに対する予測精度である. 一般的に, 訓練データを用いた学習を過度に行うことで未知データに対する汎化性能が低下し, 結果としてモデルの性能が悪化する現象が起こる. これを**過学習**という. 深層学習における最良なモデル構築をするために過学習を適切に抑え, 汎化性能を向上させることが重要である.

過学習を防止する方法は数多く提案されている. データセットを訓練データとバリデーションデータに分割し, バリデーションデータによって汎化性能を計測しながら訓練を続けることで過学習が始まるタイミングで学習を停止させる **Early Stopping** や, モデルをより簡易化させる**正則化**が存在する. 正則化には L1 正則化や L2 正則化といった様々な手法があるが, その中でもニューラルネットワークに対して強力に作用するドロップアウトがある.

2.3　ドロップアウト法

ドロップアウト法 [3, 5] とは, 深層ニューラルネットワークにおいて過学習を抑制し汎化性能を向上させるための正則化手法である. 学習時にノードを無作為に不活性化し, 残りの活性化されたノードのみで学習を行う手法である. この手法はハイパーパラメータとしてノードを不活性にする確率を決定し, 全ノードに対して活性化させるかをベルヌーイ試行によって選択する. 予測モデルを構築するにあたって, 複数モデルで予測を行い, 結果を融合させるアンサンブル学習は, 一般的に汎化性能を向上させることがわかっている. このドロップアウト法はバッチごとに活性化されるノードが異なり, 部分ネットワークの訓練を行うため, 疑似的に一つのネットワークで複数の小規模ネットワークのアンサンブル学習を行っているとも解釈できる [5]. また, あるノードから流れる情報が後続のノードに多くの影響を及ぼしているとき, そのノードが不活性化されることで他のノードによる新たな学習が進み, 小数の経路にとらわれない表現力の高いモデルを形成することができる.

ドロップアウト法は各バッチごとにドロップアウトを行うためのマスク行列を生成し, マスク処理を行うことで部分ネットワークの構築を行う. ドロップアウト率 p をハイパーパラメータで指定し, 確率 p でノードを活性化, $1-p$ でノードを不活性にする. よって第 i 行目のマスク行列 $\mathbf{M_i}$ は以下のようになる.

$$\mathbf{M_i} = (m_{i,j}), \quad m_{i,j} \sim Bernoulli(1-p)$$

生成されたマスク行列を用いて式 (1) の値をマスク処理することでドロップアウトを行う. i 層目のノードの値をもとに $i+1$ 層目のノードの値を計算方法は以下となる.

$$\mathbf{y_{i+1}} = f(\mathbf{W_i x_i} + \mathbf{b_i}) \cdot \mathbf{M_i}$$

例題 2.1.

図 1 のようなニューラルネットワークに対して以下のマスク行列

$$\mathbf{M_2} = \{1, 0, 1\}, \ \mathbf{M_3} = \{1, 0, 1, 0\}$$

を用いて中間層にドロップアウトを行うとする. このときマスク処理を行った後の部分ネットワークは図 2 のようになる.

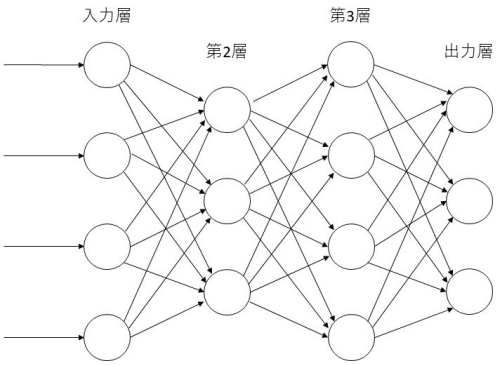

図 1: ドロップアウト前のネットワーク構造

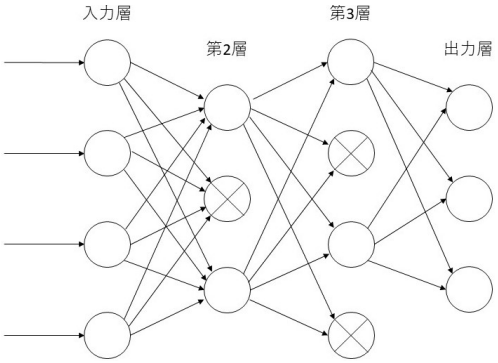

図 2: ドロップアウト後のネットワーク構造

未知データに対して予測を行う際は, ドロップアウトを行わずネットワーク全体を使用する. 訓練時は未知データの予測を行う時に比べてネットワークの規模が縮小されているため, 訓練された重みもドロップアウト率に依存してスケールが異なっている. そのため訓練時に使用したドロップアウト率 p を用いて, 未知データへの予測の際は p 倍することでスケールを変更する [5].

3　ドロップアウトデザイン

3.1　Dropout Design (DD)

既存のドロップアウト法は無作為に活性化ノードを選出するアルゴリズムに対して, ドロップアウトデザインとは, 組み合わせ最適化の観点から活性化させるノードの選出に規則性を持たせ, 均一に活性化ノードを選ぶ組み合わせ構造のことである.

ドロップアウトデザインは一つのスーパーブロックと呼ばれる集合族によって構成されており, そのスーパーブロックは各バッチごとに活性化するノードを含んだブロックを要素として持つ. 対応するブロックを参照することであるバッチの訓練時に活性化するノードを選択し, 部分ネットワークの構成をすることで計画的な深層学習を行うことができる. またこのドロップアウトデザインには各要素の集合に属する回数や, ブロックの濃度が一定であるという正則性を持つブロックデザインである. 具体的なドロップアウトデザインの定義は次に与えるとおりである.

定義 3.1 (**Dropout Design (DD)**, [1] **Definition 4.3**)**.**
$V_1, V_2, \ldots, V_n$ をそれぞれ要素数 $v_1, v_2, \ldots, v_n$ の異なる点集合とし, ブロック集合 $\mathcal{B}$ を

$$\mathcal{B} = \{\{C_1|C_2|, \ldots, |C_n\}|\ C_i \subset V_i,\ C_i \neq \phi,\ 1 \leq i \leq n\}$$

とする. 各 C_i を**サブブロック**と呼ぶ. $\mathcal{B}$ のうち, 連続する t 個の点集合におけるサブブロックの集合を

$$\mathcal{B}|_{V_i, V_{i+1}, \ldots, V_{i+t+1}} = \{\{C_i|C_{i+1}|, \ldots, |C_n\}|\ C_j \subset V_j\},\ i = 1, 2, \ldots, n-t+1$$

とする. このとき, 次の条件を満たす $(V_1, V_2, \ldots, V_n; \mathcal{B})$ を $\boldsymbol{(d_1, \ldots, d_t)}$**型 Dropout Design** (**DD**) という.
(条件)
t 個の連続する点集合 $V_{i+1}, V_{i+2}, \ldots, V_{i+t} (0 \leq i \leq n-t)$ において, 各 $V_{i+j},\ j = 1, 2, \ldots, t$ の中からそれぞれ任意に g_j 個 $(0 \leq g_j \leq d_j)$ ずつ取り出した点集合 (ただし $g_1 + g_2 +, \ldots, +g_t \geq 1$) を同時に含む $\mathcal{B}|_{V_{i+1}, V_{i+2}, \ldots, V_{i+g}}$ の中のブロック数は $\lambda^{(i+1)}_{g_1, \ldots, g_t} (\geq 1)$ 個存在する. 特に $g_i = d_i$ のとき $\lambda^{(i+1)}$ と書く.

説明を簡潔にするため, 名称とその記号を定義する.

記号	用語
V	点集合
$\mathcal{B}$	スーパーブロック
B_t	ブロック
C_t	サブブロック
$b(= \|\mathcal{B}\|)$	ブロック数, スーパーブロックサイズ
$n(= \|B_t\|)$	t 番目のサブブロック数, ブロックサイズ
$k_i(= \|C_i\|)$	i 番目のサブブロックサイズ
λ	会合数

例題 3.2.
3 つの点集合をそれぞれ $V_1 = \{0, 1, 2, 3\},\ V_2 = \{a, b, c\},\ V_3 = \{\alpha, \beta, \gamma, \delta\}$ とし, スーパーブロックを以下のように定める.

$$\begin{aligned}\mathcal{B} = \{&\{0,1|a,b|\alpha,\beta\},\ \{0,1|b,c|\alpha,\beta\},\ \{0,1|a,c|\alpha,\beta\},\ \{0,2|a,b|\alpha,\gamma\},\ \{0,2|b,c|\alpha,\gamma\},\ \{0,2|a,c|\alpha,\gamma\},\\ &\{0,3|a,b|\alpha,\delta\},\ \ \{0,3|b,c|\alpha,\delta\},\ \{0,3|a,c|\alpha,\delta\},\ \{1,2|a,b|\beta,\gamma\},\ \{1,2|b,c|\beta,\gamma\},\ \{1,2|a,c|\beta,\gamma\},\\ &\{1,3|a,b|\beta,\delta\},\ \ \{1,3|b,c|\beta,\delta\},\ \{1,3|a,c|\beta,\delta\},\ \ \{2,3|a,b|\gamma,\delta\},\ \{2,3|b,c|\gamma,\delta\},\ \ \{2,3|a,c|\gamma,\delta\}\}\end{aligned}$$

このとき, このデザインは $(1,2)$ 型 Dropout Design をなす. デザインのパラメータは以下となる.

点集合 V_i の要素数 v_i $(1 \leq i \leq 3)$	(4,3,4)
サブブロックサイズ $\lvert C_i \rvert$ $(1 \leq i \leq 3)$	(2,2,2)
ブロック数 b	18
会合数 $(\lambda_{1,2}^{(1)}, \lambda_{1,2}^{(2)})$	(3,2)

このデザインは (2,1) 型 Dropout Deisgn にもなっており, そのときの会合数は $\lambda_{(2,1)}^{(1)} = 2,\ \lambda_{(1,2)}^{(2)} = 3$ である.

3.2 Uniform Dropout Design (UDD)

ドロップアウトデザインの定義では会合数や点集合の要素数, サブブロックサイズが一定である必要はない. これらを一定にする制約を加え, デザインの構成を簡単にすることができる.

定義 3.3 (**Uniform Dropout Design (UDD)**, [1] **Definition 4.8**)**.**
$(d_1, \ldots, d_t)$ 型 DD において, n 個の点集合 V_i の要素数がすべて一定であり, かつサブブロックサイズも k で一定であるとき, $\lambda^{(1)} = \cdots = \lambda^{(n-t+1)} = \lambda$ となる. このような DD を **uniform** であるといい, $\boldsymbol{(v, k, \lambda; n)}$**-UDD** と表記する.

以降, uniform なドロップアウトデザインを考え, 点集合の要素数 v, サブブロックサイズ k, 会合数 λ は一定であるとし, この UDD を前提で話を進めていく.

定理 3.4 ([1] **Theorem4.10**)**.**
デザイン $\mathcal{D}$ を $(d_1, d_2, \ldots, d_t)$ 型 $(v, k, \lambda; t)$-UDD とする. $\mathcal{D}$ が $(d_2, d_3, \ldots, d_t, d_1)$ 型, $(d_3, d_4, \ldots, d_t, d_1, d_2)$ 型 $\ldots$ かつ $(d_t, d_1, \ldots, d_{t-2}, d_{t-1})$ 型 であるとき, $(d_i, \ldots, d_{i-1})$ 型の (v, k, λ)-UDD が存在する. ただし, $n \geq t,\ 1 \leq i \leq t$ とする.

定理 3.4 を満たす UDD を**巡回 UDD** といい, サブブロック数を増やしてデザインの形状を自由に変えることができる

例題 3.5. $(4, 2, 1; 3)$-巡回 UDD
点集合を $V_1 = \{0, 1, 2, 3\}, V_2 = \{a, b, c, d\}, V_3 = \{\alpha, \beta, \gamma, \delta\}$ とする. 以下のブロック集合は $(1, 2)$ 型かつ $(2, 1)$ 型 $(4, 2, 1; 3)$-UDD をなす.

$$\begin{aligned}\mathcal{B} = \{&\{0,1|a,b|\alpha,\beta\}, \{0,1|b,c|\alpha,\beta\},\ \{0,2|b,d|\alpha,\gamma\},\ \{0,2|a,c|\alpha,\gamma\},\\ &\{0,3|b,c|\alpha,\delta\},\ \{0,3|a,d|\alpha,\gamma\},\ \{1,2|b,d|\beta,\gamma\}, \{1,2|a,d|\beta,\gamma\},\\ &\{1,3|b,d|\beta,\delta\},\ \{1,3|a,c|\beta,\delta\},\ \{2,3|a,b|\gamma,\delta\},\ \{2,3|c,d|\gamma,\delta\}\}\end{aligned}$$

4　ドロップアウトデザインの構成

ドロップアウトデザインの構成方法について解説する. パラメータの小さなデザインであれば手作業で構成することも可能だが, 深層学習に適用するためにはより大きなパラメータを持つデザインを構成する必要がある. そこでドロップアウトデザインの構成方法は直交配列を用いたアプローチが存在するが, 今回は幾何的な側面からデザインを構成する方法を紹介する.

4.1　有限幾何の基礎知識

ドロップアウトデザインを構成するために必要なアフィン幾何についての紹介を行う. ここでは 2 次元に限定したアフィン平面から始まり, 高次元に拡張した高次元幾何の概念の解説とアフィン幾何における構成可能な部分空間の数の計算方法について述べる.

4.1.1　アフィン平面

$\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ を有限個の点集合, $\mathcal{L} = \{l_1, l_2, \ldots, l_b\}$ を有限個の直線集合とする. また, $\mathcal{I}$ を $\mathcal{P}$ と $\mathcal{L}$ の結合関係とし, p_i と l_j が結合関係にあるとは, 点 p_i が直線 l_j 上にあることを意味する. この $\mathcal{P}$, $\mathcal{L}$, $\mathcal{I}$ の組 $(\mathcal{P},\ \mathcal{L},\ \mathcal{I})$ を結合構造という.

ここでアフィン平面を定義する.

定義 4.1. アフィン平面, [9]

次の公理を満たす結合構造 $\mathcal{A} = (\mathcal{P},\ \mathcal{L},\ \mathcal{I})$ である.

(i) 任意の 2 点を通る直線は必ず 1 つ存在する.

(ii) 任意の直線 $l \in \mathcal{L}$ と l 上にはない任意の点 $p \in \mathcal{P}$ を与えたとき, p を通り l と交わらない直線 h がちょうど 1 つ存在する.

(iii) 1 直線上にない 3 点が存在する.

定理 4.2.

$\mathcal{A} = (\mathcal{P},\ \mathcal{L},\ \mathcal{I})$ が有限アフィン平面ならば以下を満たす自然数 q が存在する.

(i) 1 点を通る直線の数は $q + 1$ である.

(ii) 直線上の点は q である.

(iii) $\mathcal{A}$ の点の総数は q^2, 直線の総数は $q^2 + q$ である.

この自然数 q を**位数**という. また, 位数 q, 次元 d のアフィン幾何を $\mathbf{AG}(\boldsymbol{d}, \boldsymbol{q})$ と表す. 特にアフィン平面は AG(2, q) である. q が素数の場合, アフィン幾何上の点は mod q 上で計算される. アフィン平面のすべての直線は $ax + by = c$ で表すことができ, (a, b, c) には mod q 上の数字が入る. またすべての点はこの直線上に存在する.

例題 4.3. AG(2, 3) の構成

$\mathcal{P} = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$, $\mathcal{L}$ は表 1 のような点集合の集まりで構成された直線集合とする. 表 1 より 1 点を通る直線の数は 4 本, 直線上の点は 3 点, $|\mathcal{P}| = 9$, $|\mathcal{L}| = 12$ であり, これは定理 4.2 の主張と一致する.

(a,b,c)	$ax+by=c$	(x,y)
$(1,0,0)$	$x=0$	$(0,0),(0,1),(0,2)$
$(1,0,1)$	$x=1$	$(1,0),(1,1),(1,2)$
$(1,0,2)$	$x=2$	$(2,0),(2,1),(2,2)$
$(0,1,0)$	$y=0$	$(0,0),(1,0),(2,0)$
$(0,1,1)$	$y=1$	$(0,1),(1,1),(2,2)$
$(0,1,2)$	$y=2$	$(0,2),(1,2),(2,2)$
$(1,1,1)$	$x+y=0$	$(0,0),(1,2),(2,1)$
$(1,1,2)$	$x+y=1$	$(0,1),(1,0),(2,2)$
$(1,2,0)$	$x+y=2$	$(0,2),(2,0),(1,1)$
$(1,2,1)$	$x+2y=0$	$(0,0),(1,1),(2,2)$
$(1,2,1)$	$x+2y=1$	$(0,2),(1,0),(2,1)$
$(1,2,2)$	$x+2y=2$	$(2,0),(0,1),(1,2)$

表 1: AG(2,3) の構造

次にアフィン平面を有限体によって構成する方法を解説する. V を $\mathrm{GF}(q)$ 上の 2 次元線形空間とする. すなわち, $V = \{(a,b);\ a,b \in \mathrm{GF}(q)\}$ である. $\mathrm{GF}(q)$ とは位数 q の有限体である. ただし, 位数 q が素数べきであるとき, かつそのときに限り有限体が存在するため [9], q は素数べきとする. ここで, V の 1 次元線形部分空間を考えると, 任意の $\mathbf{x} \in V$ $(\mathbf{x} \neq 0)$ に対して

$$S\mathbf{x} = \{\alpha\mathbf{x};\quad \alpha \in \mathrm{GF}(q)\}$$

と表せる. $\mathbf{a} \in V$, $\mathbf{a} \notin S\mathbf{x}$ である各 $\mathbf{a}$ に対して,

$$S\mathbf{x} + \mathbf{a} = \{\mathbf{z} + \mathbf{a};\ \mathbf{z} \in S\mathbf{x}\}$$

を $S\mathbf{x}$ の**コセット**という. コセットを用いることですべての点を $S\mathbf{x}$ とそのコセットで分割することができる.

1 次元線形部分空間の数を数えると,

$$\begin{aligned}(1\text{ 次元線形部分空間の数}) &= \frac{(\text{全ベクトル数}) - (\text{原点})}{(1\text{ 次元部分空間の中のベクトル数})} \\ &= \frac{q^2-1}{q-1} \\ &= q+1\end{aligned}$$

と計算できる. これは原点と原点以外の 1 点を結ぶベクトルが 1 次元部分空間を張ることができ, そのベクトルの向きが等しい $q-1$ 個の重複しているベクトル数で除することで求めることができる. V の要素を点, 1 次元部分空間とそのコセットを直線ととらえることで位数 q のアフィン平面となる.

このように有限体を利用したベクトル空間に対して, 部分空間とそのコセットによってベクトルを分割することでアフィン平面を構成することができる. また後述の通り, 高次元アフィン幾何に対しても多次元ベクトル空間 V の部分空間とそのコセットを考えることで構成することができる.

4.1.2　アフィン幾何

アフィン平面では 2 次元に限定して話を進めてきた. ここからは多次元に拡張しても同様に考えることができることを解説する. GF(q) 上の m 次元ベクトル空間 V を考える. 互いに独立な k 個のベクトル $\mathbf{v_1}, \mathbf{v_2}, , , \mathbf{v_k}$ を用いて

$$M = \{\lambda_1\mathbf{v_1} + \lambda_2\mathbf{v_2} + \cdots + \lambda_k\mathbf{v_k};\ \lambda_1, \lambda_2, \ldots, \lambda_k \in \mathrm{GF}(q)\}$$
$$M + \mathbf{a} = \{\mathbf{v} + \mathbf{a};\ \mathbf{v} \in M,\ \mathbf{a} \notin M\}$$

は, V における k 次元部分空間 M とそのコセット $M + \mathbf{a}$ と表すことができ, アフィン平面のときと同様にすべてのベクトルを M とそのコセットで分割できている. アフィン平面のときは 1 次元部分空間を考えていたが, 高次元幾何の場合部分, 空間の次元数も考える必要がある. そこで以下を定義する.

定義 4.4. *k*-flat
k 次元部分空間やそのコセットのこと. 例えば 0-flat は点, 1-flat は直線, 2-flat は平面を意味する.

定理 4.5.
AG(m, q) の異なる k-flat の数は

$$\frac{(q^m-1)(q^{m-1}-1)\cdots(q^{m-k+1}-1)}{(q^k-1)(q^{k-1}-1)\cdots(q-1)} \cdot q^{m-k}$$

である.

(証明). m 次元のうち k 次元の独立なベクトルを選ぶ組合せを考える. まず, 初めの 1 つ目に選ぶベクトルは $\mathbf{0}$ 以外のベクトルを選ぶことで 1 次元部分空間を張ることができるため $q^m - 1$ 通り. 次に 2 つ目に選ぶベクトルは 1 つ目と独立なベクトルを選べばいいので, $q^m - q$ 通り. これを k 個目まで繰り返すと,

$$(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})$$

通りである. 同じ k 次元部分空間を張る重複を省くために以下の式で割ればよい.

$$(q^k-1)(q^k-q)\cdots(q^k-q^{k-1})$$

よって, 異なる k 次元部分空間の数は

$$\begin{aligned}\begin{bmatrix} m \\ k \end{bmatrix}_q &= \frac{(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})}{(q^k-1)(q^k-q)\cdots(q^k-q^{k-1})} \\ &= \frac{(q^m-1)(q^{m-1}-1)\cdots(q^{m-k+1}-1)}{(q^k-1)(q^{k-1}-1)\cdots(q-1)}\end{aligned}$$

となる. 1 つの部分空間に対してコセットはその部分空間も合わせて q^{m-k} 個あるので, k-flat の数は $\begin{bmatrix} m \\ k \end{bmatrix}_q \cdot q^{m-k}$ 個である　□

$\begin{bmatrix} m \\ k \end{bmatrix}_q$ は**ガウス係数**と呼ばれ, 高次元幾何における部分空間の組み合わせ数を求めるときに使用されることが多い.

4.2　大規模デザインの構成

ドロップアウトデザインを構成する際に 4.1 節で解説した有限幾何の概念を直接利用できる. 有限体によってアフィン空間や射影空間を構成することで "どの点がどの直線上に位置するか" という結合関係には対称性や正則性があり, これは組み合わせ数学における "点集合のどの点がどのブロックに配置されるか" という問題に言い換えることができる. またこれは, 点と直線の結合関係だけでなく, 高次元空間における t-flat と超平面の交点の集合も同様のことが言える. つまり, アフィン幾何や射影幾何を有限体によって構成することで点と直線の結合構造をブロックデザインとみなすことができ, 位数や次元数を増やすことで大規模ブロックデザインを構成することが可能になる. 例えば, 例 4.3 で紹介した AG(2, 3) は $(9, 3, 1)$-BIBD という組み合わせ構造を構成することが知られている [6].

すでに有限幾何を用いたドロップアウトデザインの構成方法はいくつも提案されており [1], そのうちのいくつかを紹介し, その定理をもとにドロップアウトデザインの構成を考える.

補題 4.6.

AG(d, q) における超平面と k-flat の積集合は $(k-1)$-flat か, k-flat を完全に含むかのどちらかである.

補題 4.6 を利用して AG(d, q) を考えることで以下のような定理が成り立つ.

定理 4.7 ([1] **Theorem5.13**)**.**

d を 3 以上の整数とし, q を素数べきとする. このとき, $(1, 2)$ 型かつ $(2, 1)$ 型 (v, k, λ)-巡回 UDD が存在する. ただし,

$$v = q^t,\ k = q^{t-1},\ \lambda = \frac{q^{d-2} - q^{d-t-1}}{q-1},\ n = q^{d-t}$$

であり, t は $2 \leq t \leq d-1$ なる整数である.

(証明).

T_1 を AG(d, q) における t-flat とし, この T_1 に対する平行類を $C_t(T_1) = \{T_1, \cdots, T_{q^{d-t}}\}$ とする. また, $\mathcal{B}$ を $\mathcal{C}_t(T)$ のどの t-flat も含まない AG(d, q) のすべての超平面の集合とする. すなわち, すべての超平面の集合 $\mathcal{H}$ に対して, $H \in \mathcal{H}$ かつ $T_i \notin H,\ i = 1, \ldots, q^{d-t}$ を満たす H に対し, $\mathcal{B}$ の各ブロックは, $(H \cap T_1 | H \cap T_2 | \cdots | H \cap T_{q^{d-t}})$ として与えられる. このとき, $\mathcal{D} = (T_1, T_2, \ldots, T_{q^{d-t}};\ \mathcal{B})$ は (1,2) かつ (2,1) 型 Dropout Design を構成することを示す.

T_i は t-flat より点集合の濃度は $v = |T_i| = q^t$, n は平行類の濃度となるため $n = q^{d-t}$ は明らかである. サブブロックは各 T_i と $H \in \mathcal{B}$ の交点集合であるので, そのサイズ k は二つの交点の数を数えればよい. 補題 4.6 より, 超平面 H と t-flat T_i の積集合は $(t-1)$-flat か T_i 自身である. H はどの T_i も含まないため交点は $(t-1)$-flat の点の数より, $k = q^{t-1}$ となる.

次にブロック数 $|\mathcal{B}|$ を考える. 定理 4.5 より, AG(d, q) における超平面の数は $\begin{bmatrix} d \\ d-1 \end{bmatrix}_q \cdot q$ であり, ある T_i を含む超平面の数は $d-1$ 次元のうち t 次元は平行類で使用している次元を選び, 残りの $d-t$ 次元から $d-t-1$ 次元を選ぶ通りとなるため, $\begin{bmatrix} d-t \\ d-t-1 \end{bmatrix}_q \cdot q$ 通りである. 超平面のコセットはその超

平面を含めて q 通りであることに注意したい. これより,

$$\begin{aligned}
\mathcal{B} &= (\text{平行類 } C_t \text{内のどの } T_i\ (1 \le i \le q^{d-t}) \text{ も含むことのない超平面の数}) \\
&= (\text{超平面の数}) - (\text{ある } T_i \text{を含む超平面の数}) \\
&= \begin{bmatrix} d \\ d-1 \end{bmatrix}_q \cdot\ q\ - \begin{bmatrix} d-t \\ d-t-1 \end{bmatrix}_q \cdot\ q \\
&= \begin{bmatrix} d \\ 1 \end{bmatrix}_q \cdot\ q\ - \begin{bmatrix} d-t \\ 1 \end{bmatrix}_q \cdot\ q \\
&= q\left(\frac{q^d-1}{q-1} - \frac{q^{d-t}-1}{q-1}\right) \\
&= \frac{q(q^d-q^{d-t})}{q-1}
\end{aligned}$$

と計算される.

最後に会合数 λ を考える. $P_1, P_2 \in T_i,\ Q \in T_j,\ (1 \le i \ne j \le q^{d-t})$ である 3 点 $P_1,\ P_2,\ Q$ が同時に含まれる超平面 $H \in \mathcal{B}$ の枚数を考える. P_1, P_2 を含む超平面の数は $d-1$ 次元のうち 2 次元は 2 点により固定され, 残りの $d-2$ 次元から $d-3$ 次元を選べる数を求める. そこから 3 点 P_1, P_2, Q を含みつつ, T_i を含む超平面の数を引くことで会合数を求めるが, T_i と点 Q を含むため $t+1$ 次元は固定し, 残りの $d-t-1$ 次元から $d-t-2$ 次元を選ぶことで 超平面の数を求める. ただし, 今回は実際の点を含む超平面を考えているため, コセット数を乗する必要はないことに注意したい. 計算式は以下となる.

$$\begin{aligned}
\lambda &= (P_1, P_2 \text{を含む超平面の数}) - (P_1, P_2, Q \text{ を含み, } T_i \text{を完全に含む超平面の数}) \\
&= \begin{bmatrix} d-2 \\ d-3 \end{bmatrix}_q - \begin{bmatrix} d-t-1 \\ d-t-2 \end{bmatrix}_q \\
&= \begin{bmatrix} d-2 \\ 1 \end{bmatrix}_q - \begin{bmatrix} d-t-1 \\ 1 \end{bmatrix}_q \\
&= \frac{q^{d-2}-1}{q-1} - \frac{q^{d-t-1}-1}{q-1} \\
&= \frac{q^{d-2}-q^{d-t-1}}{q-1}
\end{aligned}$$

□

実際に定理 4.7 をもとにドロップアウトデザインを構成することを考える. (d, t, q) を $(3, 2, 2)$ とし, AG(3, 2) をもとに $(4, 2, 1; 2)$-UDD を作成する. まずは AG(3,2) 全体の点集合と GF(2^3) の原始元によるべき乗表現を行う. 原始元とは, 有限体の全ての非零元をべき乗で生成できる要素のことであり, 最小多項式とは, 原始元を根に持つ最小次数の既約多項式である. 原始元 α の最小多項式を $\alpha^3 + \alpha + 1 = 0$ としておくと, 以下のように表せる.

べき乗表現	$\mathbf{GF}(\mathbf{2^3})$ 上の多項式	多項式の係数
α^{∞}	0	$(0,0,0)$
α^1	α	$(0,1,0)$
α^2	α^2	$(1,0,0)$
α^3	$\alpha+1$	$(0,1,1)$
α^4	$\alpha^2+\alpha$	$(1,1,0)$
α^5	$\alpha^2+\alpha+1$	$(1,1,1)$
α^6	α^2+1	$(1,0,1)$
α^7	1	$(0,0,1)$

表 2: AG(3, 2) における点集合

このように原始元のべき乗を考えることでアフィン幾何における任意の点を表現することができる. ここからは説明の簡潔化のために表 2 の点を上から $0,1,2,3,4,5,6,7$ と命名する (∞ を 0 としたときのべき乗を意味する). 次に AG(3, 2) における 2-flat, すなわち, ある平面を $T_1=\{0,1,3,7\}$ とすると, この平行類 $C_t(T_1)$ は $C_t(T_1)=\{T_1,T_2\},\ (T_1=\{0,1,3,7\},\ T_2=\{2,4,5,6\})$ である. AG(3, 2) は 8 点で構成されており, そのうちある一つの平面は 4 点で構成されるため, コセットは 2 つである. この T_1,T_2 がドロップアウトデザインにおける点集合となる.

ドロップアウトデザインのスーパーブロックは平行類 C_t を含まない超平面と平行類との交点集合である. 今回は 3 次元アフィン幾何を考えているため超平面は 2 次元平面となる. $T_1,\ T_2$ を含まない超平面は以下の 12 通りである.

$$\{0,2,6,7\},\{1,3,4,5\},\{0,1,2,4\},\{3,5,6,7\}$$
$$\{0,2,3,5\},\{1,4,6,7\},\{0,4,5,7\},\{1,2,3,6\}$$
$$\{2,3,4,7\},\{0,1,5,6\},\{0,3,4,6\},\{1,2,5,7\}$$

T_1,T_2 を含まない超平面の数は 3 次元の中から 2 次元部分空間を選ぶ組み合わせ数から T_1,T_2 の 2 平面を引くことにより, $\begin{bmatrix}3\\2\end{bmatrix}_2\cdot 2-2=12$ であることが確認できる. この 12 通りの超平面と T_1,T_2 の交点がブロックを構成する. 例えば, 超平面 $\{0,2,6,7\}$ と平行類 $C_t(T_1)$ との交点は $\{0,7|2,6\}$ となる. 最終的に定理 4.7 をもとに, (d,t,q) を $(3,2,2)$ としたときの AG(3, 2) によって構成される $V_1,\ V_2,\ \mathcal{B}$ は以下となる.

$$\begin{aligned}V_1&=\{0,1,3,7\},\quad V_2=\{2,4,5,6\},\\ \mathcal{B}&=\{\{0,7|2,6\},\ \{1,3|4,5\},\ \{0,1|2,4\},\ \{3,7|5,6\},\\ &\qquad \{0,3|2,5\},\ \{1,7|4,6\},\ \{0,7|4,5\},\ \{1,3|2,6\},\\ &\qquad \{3,7|2,4\},\ \{0,1|5,6\},\ \{0,3|4,6\},\ \{1,7|2,5\}\}\end{aligned}$$

このとき, $(V_1,V_2;\ \mathcal{B})$ は (1,2) 型かつ (2,1) 型 UDD をなす. スーパーブロックを見てわかる通り, 任意の $a,b,c\ (a\in C_1,\ b,c\in C_2,\ b\neq c)$ は 1 組ずつあることがわかる. これは会合数 λ が 1 であることを意味する.

アフィン空間			UDD			
d	q	t	v	k	λ	n
6	2	4	16	8	14	4
8	2	7	128	64	63	2
7	3	6	729	243	121	3
10	3	8	6561	2187	3279	9
10	7	9	40353607	5764801	960800	7

表 3: 定理 4.7 より構成されるドロップアウトデザイン

定理 4.7 より, d, t, q を定め, ドロップアウトデザインの構成可能なパラメータは以下のようなものが考えられる.

このようにパラメータの大きな大規模デザインの構成も幾何学的な知見から構成可能であり, より大規模な深層ニューラルネットワークに対してもドロップアウトデザインを適用することができる. ドロップアウトデザインの構成方法の一部を紹介したが, ほかにも高次元射影幾何を利用することで多層かつ, 多くのノードを持つデザインを構成することができる.

5　深層学習への適用

生成されたドロップアウトデザインを深層学習のニューラルネットワークに適用する方法を説明する. 既存のドロップアウト法では, ハイパパラメータ p によってニューラルネットワークに対するマスク行列を生成し, マスク処理を行うことで不活性にし, ノードの学習を行わないようにしていた. ドロップアウトデザインを適用する際も同様に, 指定したパラメータをもとに生成されたドロップアウトデザインのブロックを参照することでマスク行列を生成し, 同じ処理を行うことが目的である.

ここでドロップアウトデザインとニューラルネットワークの適用方法について整理する.

Dropou Design の役割	**ニューラルネットワークの役割**
スーパーブロック $\mathcal{B}$	1 エポック (全バッチ) の活性化ノード集合
ブロック $B_i \subset \mathcal{B}$ $(1 \leq i \leq n)$	i 番目のバッチに対する活性化ノード集合
ブロック数 (スーパーブロックサイズ) b	バッチ数
サブブロック $C_j \subset B_i$ $(1 \leq i \leq b,\ 1 \leq j \leq n)$	i バッチ目 j 層目の活性化ノード集合
サブブロックサイズ $k = \lvert C_i \rvert$ $(1 \leq i \leq n)$	i 層目の活性化ノード数
点集合 V_i $(0 \leq i \leq v_i)$	第 i 層目のノード集合
点集合の要素数 v_i $(0 \leq i \leq n)$	第 i 層目のノード数
$(d_1, d_2, \cdots, d_t)$ 型 Dropout Design における会合数 λ	i 層目から $i+t$ 層目にかけて任意にそれぞれ $d_1, d_2, \cdots, d_t$ 個ずつ選んだノードの組が同時に活性化されるバッチ数
点集合に対するサブブロックの濃度の割合 $\frac{\lvert C_i \rvert}{\lvert V_i \rvert}$ $(1 \leq i \leq n)$	i 層目のノードの活性化率 (= 1− ドロップアウト率)

上記のようにドロップアウトデザインの各ブロックを深層学習のバッチにおける活性化ノードの集合とすることでマスク行列を生成することができる. 例 3.2 によるドロップアウトデザインをニューラルネットワークに適用し, ドロップアウトを実行する手順を解説する.

まず, デザインのパラメータからニューラルネットワークの構造が決定される. 各層のノード数や各層ごとのドロップアウト率, 各バッチで処理するデータ数であるバッチサイズといったハイパーパラメータは, ドロップアウトデザインを使用すると自動的に決定される. 例 3.2 で紹介したドロップアウトデザインを使用したときのニューラルネットワークのハイパパラメータは以下となる.

ハイパパラメータ	**1 層目**	**2 層目**	**3 層目**
ノード数	4	3	4
活性化ノード数	2	2	2
ドロップアウト率	0.5	0.67	0.5

各層のノード数はドロップアウトデザインの点集合の濃度に対応し, 活性化ノード数は各サブブロックのノード数に対応する. また, ドロップアウト率は点集合ノードのうち, サブブロックに含まれるノードの割合により計算される.

次に各バッチごとに対応するブロックをもとにマスク行列を生成する. 各サブブロックが各層のノードに対応しているため, 層ごとにマスク行列 (ベクトル) を準備する. スーパーブロックのはじめの要素 (ブロック) により 1 バッチ目のドロップアウトを行うとき, 1,2,3 層目のノードに対するマスク行列は $\mathbf{M_1} = (1, 1, 0, 0)$, $\mathbf{M_2} = (1, 1, 0)$, $\mathbf{M_3} = (1, 1, 0)$ となる. ドロップアウトマスクは各バッチごとにドロップアウトデザインの対応するブロックを参照することでマスク処理を行う.

例 3.2 のデザインを深層学習に適用したときのニューラルネットワーク構成と, 1, 2 ブロック目を用いてドロップアウトを行い, 第 1, 2 バッチ目の学習ネットワークの図は以下のとおりである.

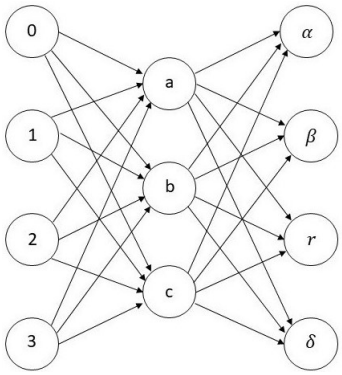

図 3: 例 3.2 のニューラルネットワーク構造

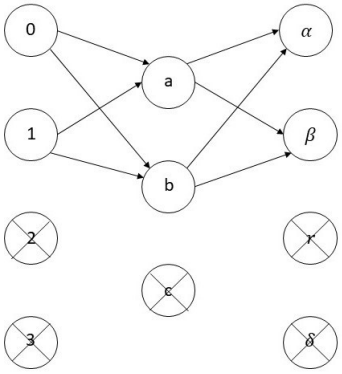

図 4: 1 バッチ目におけるドロップアウト後

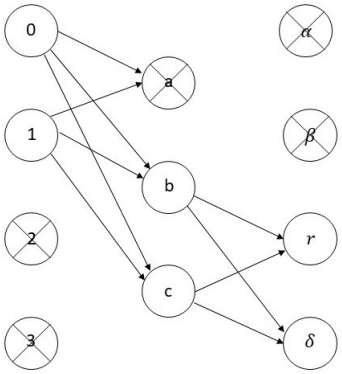

図 5: 2 バッチ目におけるドロップアウト後

5.1 実験設定

本実験では, 深層学習によるドロップアウトデザインの有効性を検証した. 実験題材として cifar10[4] データセットに対する画像分類を扱い, 畳み込みニューラルネットワーク (CNN) をベースにテストデータに対する汎化性能の違いを考察する. cifar10 とは飛行機, 自動車, 鳥などの 10 種類に対して各クラス 6000 枚からなる画像データセットである. 各画像は 32×32 ピクセルのカラー画像であり, 1 クラス当たり訓練データとテストデータがそれぞれ 5000, 1000 枚の計 50000,10000 枚で構成されている.

5.1.1 ネットワーク構造

CNN をベースとし, 畳み込み層は共通の構造とした. 全結合層において, ドロップアウトデザインのパラメータによってネットワーク構造を決定し, 以下の 4 つのモデルについてそのネットワーク構造に合わせて実験を行った.

(i) 正則化なし

(ii) ドロップアウト法

(iii) ドロップアウトデザイン（シフトあり）

(iv) ドロップアウトデザイン（シフトなし）

ドロップアウトデザインのシフトありとは, 各エポックごとに各バッチで適用するブロックの順序を変更し, 画像の学習時に使用するネットワークを変更することを意味する. シフトをしないモデルでは各画像を用いて学習される際の活性化ノードを固定することになる.

層数, 各層のノード数やドロップアウト率などのネットワーク構造は表 3 で構成されたドロップアウトデザインのパラメータをもとにを決定した. 以下が実験に使用したデザインパラメータとニューラルネットワークの構造である.

実験	ドロップアウトデザイン				ニューラルネットワーク				
	サブブロック数	点集合サイズ	サブブロックサイズ	スーパーブロックサイズ	層数	1 層あたりのノード数	1 層あたりの活性化ノード数	ドロップアウト率	バッチ数
1	2	128	64	508	2	128	64	0.5	508
2	3	729	243	3276	3	729	243	0.67	3276

5.1.2 学習におけるハイパーパラメータ

各モデルの学習は, 以下の設定で行った.

- 最適化アルゴリズム: 確率的勾配降下法 (SGD)
- エポック数: 100
- 活性化関数: ReLU (出力層は softmax)
- 損失関数: クロスエントロピー

5.1.3 実行環境

以下のライブラリや計算資源によって実行を行った.

- 深層学習フレームワーク: Keras 2.4.3, tensorflow 2.4.0
- GPU: NVIDIA A100 (80GB)
- CPU: AMD EPYC 7453 28-Core Processor
- OS: Ubuntu 20.04.4 LTS

上記の設定で, 各ニューラルネットワークに対して 4 つのモデルを学習させた. 学習を 10 回繰り返し, 精度 (正解率) や損失の平均と分散を算出した. 各モデルの性能評価は, テストデータを用いて行った.

5.2 実験結果

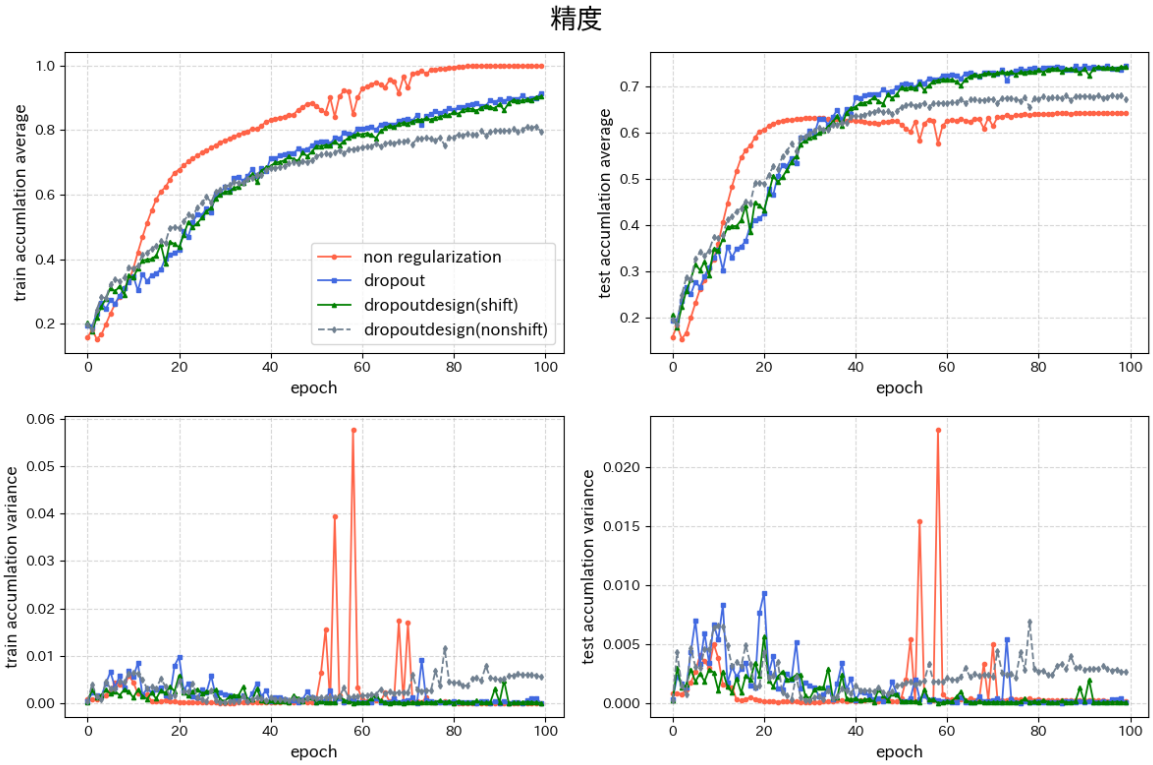

図 6: 2 層,1 層あたり 128 ノード, ドロップアウト率 0.5 の実験結果

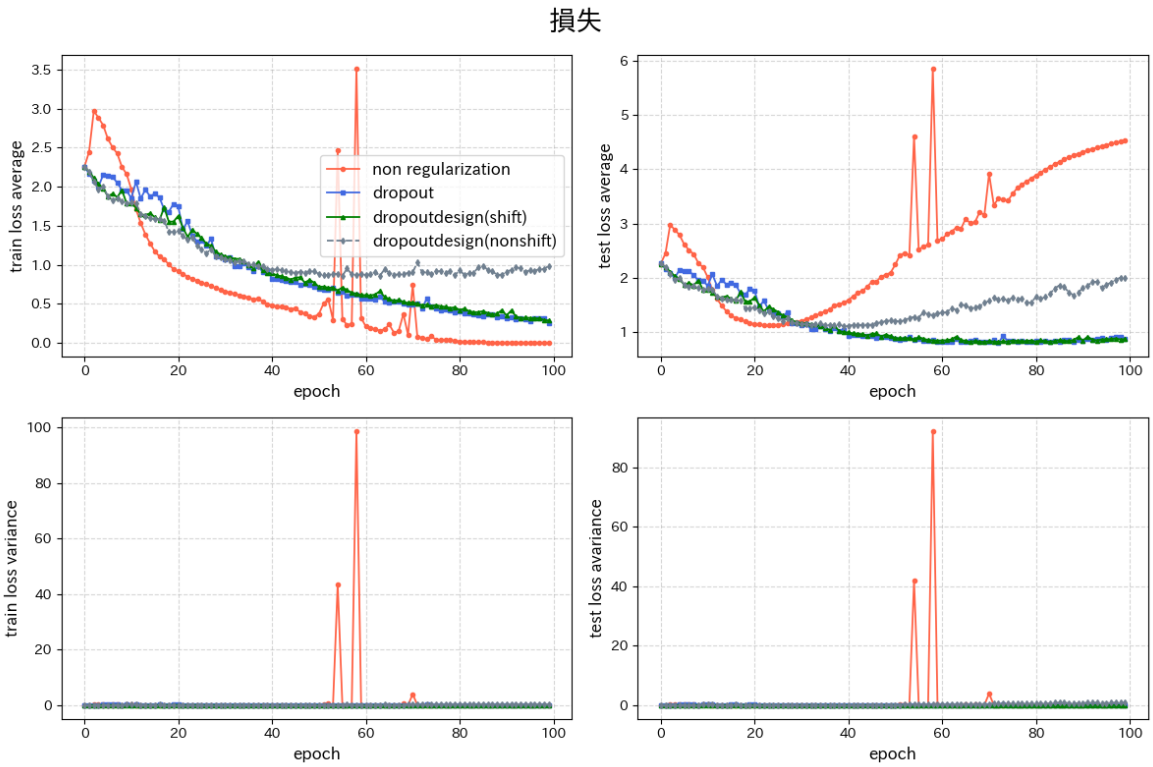

図 7: 2 層,1 層あたり 128 ノード, ドロップアウト率 0.5 の実験結果

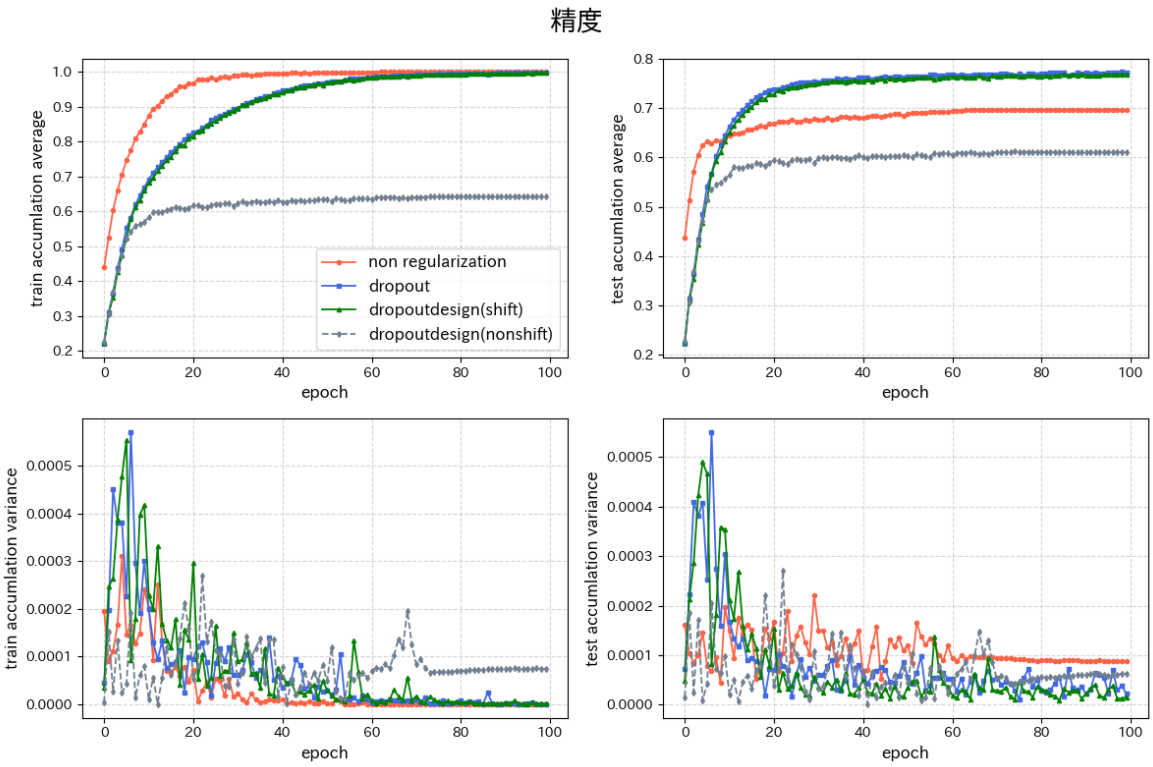

図 8: 3 層,1 層あたり 729 ノード, ドロップアウト率 0.67 の実験結果

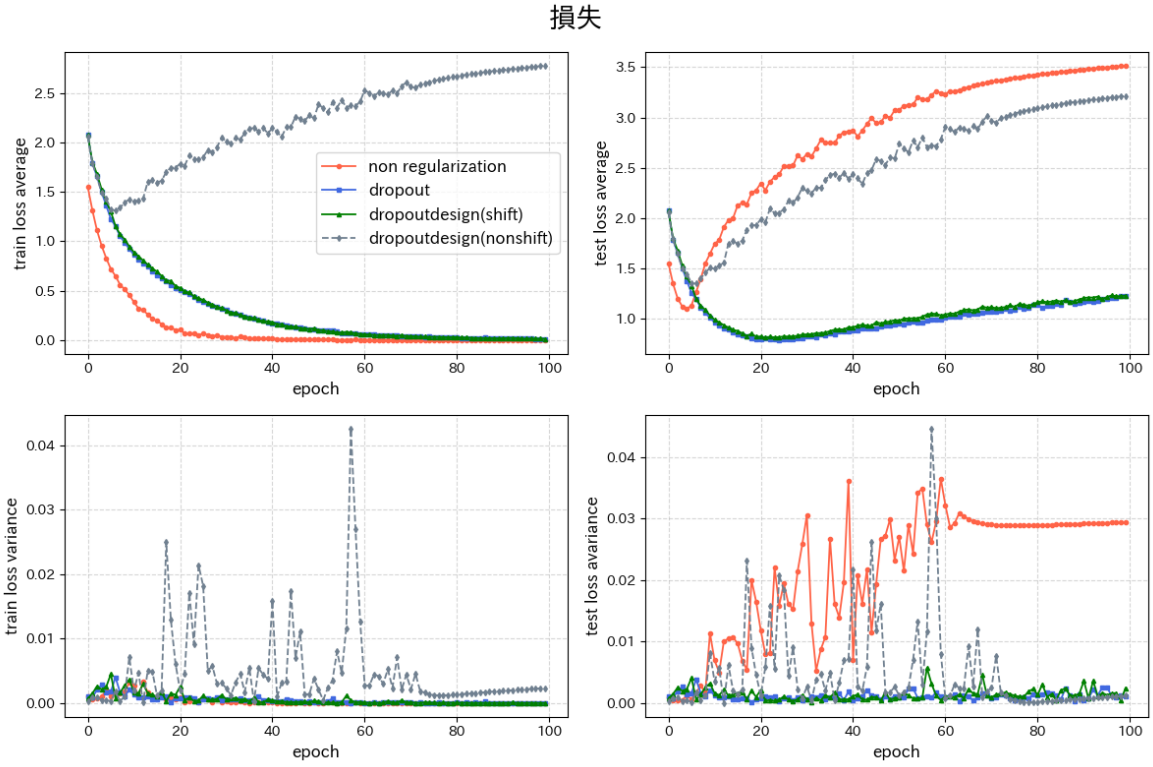

図 9: 3 層,1 層あたり 729 ノード, ドロップアウト率 0.67 の実験結果

5.2.1 過学習の発生とその抑止効果

全体を通して, 学習が進むにつれて訓練データへの精度が上がっていくのに対し, テストデータへの精度が改善されなくなっている. また, テストデータへの損失が数エポック目を境に減少から増加に変化していることから典型的な過学習を起こしていることがわかる. その中でも non regularization は他の正則化を施したモデルに比べてより早く過学習を起こし, 最終的なテストデータへの精度も劣っている. つまり, non regularization は 5 エポック目前後で過学習を引き起こしてテストデータへの精度が約 0.7 に対して, dropout と dropout design (shift) は 20 エポック前後まで過学習を抑え, 最終的な精度も約 0.8 であり, 正則化の機能を果たしていると考えられる.

5.2.2 学習の収束速度

学習速度に関しても non regularization と dropout, dropout design (shift) で大きく挙動が異なる. non regularization の方は訓練データに対する損失・精度の分散が正則化法に比べて速く収束しているが, テストデータに対しての損失分散は学習を進めるれば進めるほど増えているため, 過適合によって未知データに最適なモデルになっていないことがわかる. 一方で droout, dropout design (shift) は訓練データ, テストデータどちらも学習を進めていく過程でモデルが安定していることがわかる.

5.2.3 ドロップアウト法とドロップアウトデザインの比較

dropout と dropout design (shift) に注目して結果を比較する. 二つのモデルは実験全体の指標を見てわかる通り, 基本的にはほぼ同程度の正則化効果を見せており, dropout design を使用することによる固有の特徴は見られなかった. テストデータに対する損失分散に注目すると, わずかながら dropout design (shift) よりも dropout の方が小さい値で安定しており, 学習における安定性があると考えられる. しかし, 今回は 10 回の実験における分散を計算しており, 実験回数を増やしたり, ほかのネットワークを使用することで dropout design が dropout よりも安定なモデルとなる可能性は十分にあると考える.

dropout design (nonshift) における実行結果は他の正則化法に比べて大きく違う学習挙動が見られた. これはドロップアウト法のランダム性を仮定しないでノードを不活性にしていることで学習に偏りが生じ, 結果として正しく過学習を抑制できていないと考えられる. また, 今回は nonshift による挙動を明確に再現するためにエポックごとに訓練データをシャッフルしない設定で実験を行った. これは, 各バッチごとに活性化されるノードは一定かつ訓練データとして使用する画像が同じであるため, エポックごとに全く同じ学習をしていることになる. よって, 表現力の高いモデル構築ができていないと考える. 実際, 最終的なテストデータへの精度は正則化をしないよりも低い結果であり, 分散に関しては他モデルに比べて大きな値で増加している. ここからわかることは, ドロップアウトにおいて不活性にするノード選出に偏りを持たせて実行することはモデルの不安定性の原因になる. モデル最適化アルゴリズムである確率的勾配降下法 (SGD) では, ランダムなデータによって重みに対する勾配を求め, 重みの最適化を行っている. ランダム性が仮定されない中で重み最適化を行うと, 深層学習における大きな課題である局所解に陥りやすく, グローバル解の探索が行われない可能性がある. そのため, ドロップアウト法において偏りなくノードを不活性にすることも同様に重要であると考える.

6　今後の課題

結論としてドロップアウトデザイン特有の正則化効果は確認できなかった. しかし, ドロップアウトデザインを用いた深層学習を行うことでモデルの収束時点が早まり, それを正確に検知して学習を止めることで, 深層学習を行うコストを削減できる可能性は大いにあると考える. 今回使用したデータセットは比較的小規模であり, 正則化効果を調べる上でより規模の大きいデータセットを使うことでドロップアウト法に比べて正則化の振る舞いが変化する可能性もある.

しかし, 仮にドロップアウトデザインによる正則化を行うことで優れた結果が確認できたとしても, 実用化まではいくつもの課題点がある. 最も大きな課題点はニューラルネットワーク構造を決定する際の操作性を高めることである. 現在の実装法では, ドロップアウトデザインの構成パラメータにあわせてニューラルネットワークの構造が強制的に決定されている. 層数や各層の活性化ノード数, ドロップアウト率などは本来予測したいデータセットの特徴に合わせて最適なニューラルネットワーク構造を選択できるべきである. 特に今回の実験では各層のノード数は一定であると仮定した UDD でのネットワークを構成したが, 各中間層のノード数が一定である場面はむしろ少ない. 画像認識や画像分類では入力層から出力層にかけてノード数を徐々に減らし, 逆に画像生成では増やしていくようにネットワーク構造を決定することでより良いモデル構築ができるとされている. そのため, Uniform でないドロップアウトデザインによる実装をより簡潔に実装できるようにすることや, 中間層に対してパラメータの異なる複数のドロップアウトデザインを適用することで操作性を高めていく必要がある.

ブロックデザインを深層学習に適用する研究は他にも数多く行われている. ノードを不活性にするのではなくノード間のエッジを切断するドロップコネクト法 [7] や, UDD を派生させた circulant dropout design による深層学習への適用 [2] などがある. これらによる優れた正則化が確認できるか検証していきたい.

参考文献

[1] Shoko Chisaki, Ryoh Fuji-Hara, and Nobuko Miyamoto. Combinatorial designs for deep learning. *Journal of Combinatorial Designs*, Vol. 28, No. 9, pp. 633–657, 2020.

[2] Shoko Chisaki, Ryoh Fuji-Hara, and Nobuko Miyamoto. A construction for circulant type dropout designs. *Designs, Codes and Cryptography*, Vol. 89, No. 8, pp. 1839–1852, 2021.

[3] GE Hinton. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[6] W.D. WALLIS. *INTRODUCTION TO COMBITIONAL DEESIGNS*. Chapman and Hall/CRC, 2007.

[7] Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, 2013.

[8] 地嵜頌子. 深層学習における正則化へのドロップアウトデザインの適用. ソフトウェア・シンポジウム 2022 論文集, Vol. 1, , 2022.

[9] 藤原良, 神保雅一. 符号と暗号の数理. 共立出版, 1903.