論文

クラスごとの性能不均衡を考慮した深層学習モデルの最適化とデータ拡張戦略の検討

岩本 隆志 [1)]

キーワード：Efficiennet、CNN、CIFAR-10、CIFAR-100、アンサンブル学習、不均衡データ

1 はじめに

2025 年には、情報技術（IT）の分野において重要な課題が浮上してくる。この年までに、ブラックボックス化したシステムへの対応がますます困難になると予想されている。この問題の背後には、IT技術者の大量定年退職などが影響しており、システムの改定、改修、運用保守業務への対応が難しくなる可能性がある。この問題は非常に深刻で、対応を怠ると2025 年以降、経済に大きな損失をもたらす可能性がある。具体的には、年間で最大 12 兆円にも達する経済損失が発生する可能性があり、これは一般に「2025 年の崖」と呼ばれている。さらに、現在進行形で、人工知能（AI）の技術が急速に進歩し、私たちの社会に多くの変化をもたらしている。例えば、買い物をする際にも、自動レジが普及しており、AI技術の影響が日常生活に浸透している。AI技術は大まかに分けると機械学習とディープラーニングに区分できる。その中でも、特に社会に大きな変革をもたらしているのがディープラーニングである。ディープラーニングは、多層のニューラルネットワークを使用して、複雑なタスクを学習し、高度な認識能力を持つAIモデルを構築する技術である。本研究は、畳み込みニューラルネットワーク（CNN）と転移学習を組み合わせたアプローチに焦点を当て、画像認識に関連するタスクにおける特徴と課題を明確にし、AI技術の進化に向けた理解を深めることを目的とする。

1.1 CNN

CNN（Convolutional Neural Network）は、その名前からもわかるように、畳み込みニューラルネットワークとしても知られており、主に画像認識の分野で重要な役割を果たしている。このネットワークは、多くの深い層で構成されており、畳み込み層［注 1］、プーリング層［注 2］、ドロップアウト層［注 3］など、独自の機能を組み合わせている。CNN の最

1) 山陽学園大学総合人間学部ビジネス心理学科

も顕著な特徴は、一般物体認識と呼ばれる画像認識分野での卓越した性能である。この技術は、複雑な画像データセットから物体や特徴を識別し、高い精度でタスクを実行できることで知られており、自動運転車、医療診断、セキュリティ監視など、さまざまな領域で利用されている。さらに、CNN は転移学習やファインチューニングといった手法と組み合わせて使用することで、ネットワークの学習を効率化することに成功している。これは、事前に訓練されたモデルを新しいタスクに適用し、少ないデータで高い性能を達成する方法である。この革新的なアプローチにより、多くの実世界の問題に対して効果的なソリューションを提供する可能性が広がった。CNN は畳み込み層、プーリング層、ドロップアウト層などの特徴的な要素を持つ深層学習ネットワークであり、主に画像認識分野で優れた性能を発揮する。これらの要因から、CNN は機械学習とコンピュータビジョンの分野で重要な技術として広く注目されている［1-2］。

1.2 CNN の数式表現

U:入力特徴マップの枚数、M:入力特徴マップ高さ,N:入力特徴マップ幅、C′:フィルタの枚数、I:フィルタ高さ、J:フィルタ幅、M′：出力特徴マップ高さ、N′：出力特徴マップ幅とすると、順伝播におけるパディングされたバッチ入力データ$\mathbf{X}_u$に対する畳み込みは、出力データ$\mathbf{Y}_u$の成分$y_{u,c',m',n'}$は

$$y_{u,c',m',n'} = \sum_{c=0}^{c-1}\sum_{i=0}^{I-1}\sum_{j=0}^{J-1} x_{u,c,i+Sm',j+Sn'} w_{c',c,i,j} + b_{c'} \tag{1}$$

という関係で定式化される。次に、バイアス$b_{c'}$を除いた部分は線型結合となり行列積となる。総和はc、i、j に対して行っており、$\mathbf{X}_u$は 2 次元配列となる。次に逆伝播について考える。損失関数Lを$w_{c',c,i,j}$、$b_{c'}$について偏微分すると、

$$\frac{\partial L}{\partial w_{c',c,i,j}} = \sum_{u=0}^{U-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} \frac{\partial L}{\partial y_{u,c',m',n'}} \frac{\partial y_{u,c',m',n'}}{w_{c',c,i,j}} = \sum_{u=0}^{U-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} \frac{\partial L}{\partial y_{u,c',m',n'}} x_{u,c,i+Sm',j+Sn}$$

$$\frac{\partial L}{\partial b_{c'}} = \sum_{u=0}^{U-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} \frac{\partial L}{\partial y_{u,c',m',n'}} \frac{\partial y_{u,c',m',n'}}{b_{c'}} = \sum_{u=0}^{U-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} \frac{\partial L}{\partial y_{u,c',m',n'}} \tag{2}$$

となる。ここで、$\partial L / \partial y_{u,c',m',n'}$の配列と$UM'N' \times C'$は同次元の配列である。

次に、損失関数Lを$\partial x_{u,c,m,n}$を用いて偏微分すると、

$$\frac{\partial L}{\partial x_{u,c,m,n}} = \sum_{c'=0}^{C'-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} \frac{\partial L}{\partial y_{u,c',m',n'}} \frac{\partial y_{u,c',m',n'}}{\partial x_{u,c,m,n}} \tag{3}$$

となる。上式において、$\partial y_{u,c',m',n'} / \partial x_{u,c,m,n}$をさらに変形すると、

$$\frac{\partial L}{\partial x_{u,c,m,n}} = \sum_{c'=0}^{C'-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} \frac{\partial L}{\partial y_{u,c',m',n'}} \left(\sum_{i=0}^{I-1}\sum_{j=0}^{J-1} w_{c',c,i,j} \right)$$

$$= \sum_{i=0}^{I-1}\sum_{j=0}^{J-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1}\left(\sum_{c'=0}^{C'-1}\frac{\partial L}{\partial y_{u,c',m',n'}}w_{c',c,i,j}\right)$$

$$= \sum_{i=0}^{I-1}\sum_{j=0}^{J-1}\sum_{m'=0}^{M'-1}\sum_{n'=0}^{N'-1} dout_{u,c,i,j,m',n'}, \text{ where } m = i + Sm' \text{ and } n = j + Sn' \tag{4}$$

となる［3］。

1.3 転移学習の数式表現

$\mathcal{X}$を特徴空間、$\mathcal{D} = \{\mathcal{X}, P(X)\}$とすると、周辺確率分布$P(X)$は$X = \{x_1, \dots, x_n\} \in \mathcal{X}$を満たす。次に、教師あり手法に関する定式化を進める。S はソースドメイン、T はターゲットドメインである。U を元の高次元データを低次元表現にマッピングするための直交行列とすると、

$$argmin_{A,U} \sum_{t \in T,S} \sum_{i=1}^{n_t} L(y_{t_i}, \langle a_t, U^T x_{t_i} \rangle) + \gamma |A|_{2,1}^2 \tag{5}$$

$$\text{s.t. } U \in \mathbf{O}^d$$

と定式化できる。さらに、教師なし手法に関する定式化を進めと、

$$\min_{a,b} \sum_i |x_{S_i} - \sum_{i=0}^{I-1} a_{S_i}^j b_j|_2^2 + \beta |a_{S_i}|_1 \tag{6}$$

$$\text{s.t. } |b_j|_2 \leq 1, \forall j \in 1, \dots, s$$

となる。さらに、上式を最適化問題として捉え、基底ベクトルb_jが学習される。次に、

$$a_{T_i}^* = \arg \min_{a_{T_i}} \min |x_{T_i} - \sum_j a_{T_i}^j b_j|_2^2 + \beta |a_{T_i}|_1 \tag{7}$$

の目的関数を最小化することにより定式化できる［4-5］。

1.4 EfficientNet

EfficientNet は 2019 年に提案された深層学習モデルで、モデルのサイズと性能のバランスを取りながら、効率的な設計を行うアプローチである。このモデルは、異なるスケールのモデルを提供するために、Convolutional Neural Network（CNN）のスケーリング原則を導入している。その特徴を表 1 に示す[6] 。

1.5 CIFAR-10

CIFAR-10（Canadian Institute for Advanced Research 10）は、コンピュータビジョンや機械学習の分野で広く使用されるデータセットの 1 つある。CIFAR-10 データセットは、10 の異なるクラスに属するカラー画像から構成されており、主に画像認識タスクのベンチマークとして使用される。図 1 に CIFAR-10 を、表 2 に詳細を、図 3 CIFAR-10 の不均衡データを示した散布図を示す。

1.6 CIFAR-100

CIFAR-100（Canadian Institute for Advanced Research 100）は、CIFAR-10 データセットの拡張版であり、コンピュータビジョンと機械学習の分野で広く使用されるデータセットの 1 つ

である。CIFAR-100 データセットは、より多くのクラスとより高度な認識タスクに焦点を当てたものである。図 2 に CIFAR-100 を、表 3 に CIFAR-100 に関する詳細を、図 4 CIFAR-100 よりランダムに選出したクラスを示す。

表 1 EfficientNet の特徴　[6]を元に筆者作成

	説明
複合スケーリング（Compound Scaling）	チャンネル数、レイヤー数、解像度をバランスよくスケーリングし、高性能化を実現。
効率的なブロック（Efficient Blocks）	Efficient Block を使用し、特徴抽出の効率化を図りつつ、計算効率を向上させる。
深さ方向分離畳み込（Depth-wise Separable Convolution）	ネットワーク全体の計算量を削減し、効率的なモデル構築に使用される畳み込み方式を使用。
バージョン（B0 から B7）	B0 は小さなモデルで、B7 は大きなモデル。

表 2 CIFAR-10 に関する詳細 [7-8]を元に筆者作成

	詳細
クラス	・10 クラス（Airplane、Automobile、Bird、Cat、Deer、Dog、Frog、Horse、Ship、Truck）。
画像	・カラー画像で構成され、解像度は 32x32 ピクセル。
トレーニングとテストセット	・トレーニングには約 5 万枚、テストには約 1 万枚の画像。
用途	・CNN 等の機械学習アルゴリズムやディープラーニングモデルの性能を評価し、比較するためのベンチマークとして使用される。

表 3 CIFAR-100 に関する詳細 [9]を元に筆者作成

	詳細
クラス	・100 の異なるクラスが含まれており、更にこれらのクラスは、より詳細なカテゴリに分けられている。 ・CIFAR-10 のクラス「犬」が CIFAR-100 では「家庭用の犬」や「野生の犬」等、より具体的なクラスに分割されており、認識の難易度が高くなっている。
画像	・CIFAR-10 と同様にカラー画像で、解像度は 32x32 ピクセル
用途	・より高度な物体認識タスクや分類タスクを評価するために使用される。 ・画像認識の研究において、モデルの汎化能力や複雑なカテゴリの認識能力をテストするのに役立つ。

図 1 CIFAR-10 データセット [8]

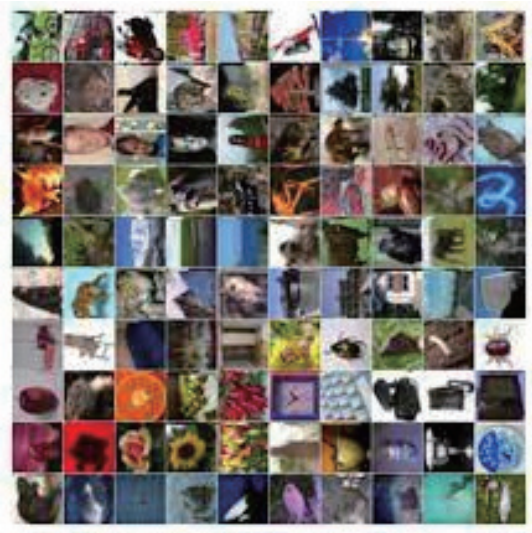

図 2 CIFAR-100 データセット [9]

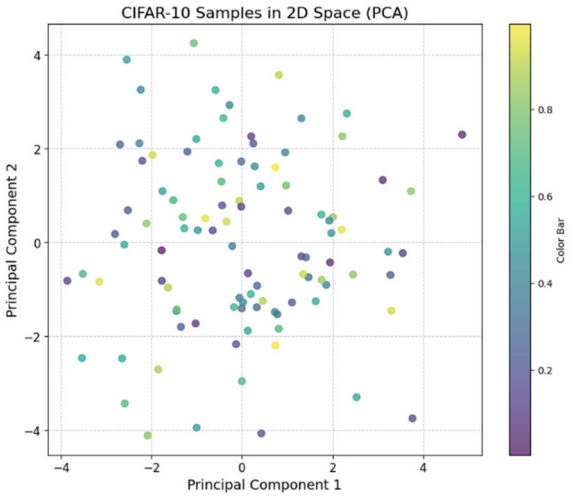

図 3 CIFAR-10 の不均衡データを示した散布図

図 4 CIFAR-100 よりランダムに選出したクラス

2 実験

CIFAR-10 データセットと CIFAR-100 データセットを用いて、画像識別実験 1 から 3 を、Coogle Colaboratory TPU で実施した。

2.1 実験 1

CIFAR-10 データセットを使用し、EfficientNetB5 を用いて、畳み込み CNN を用いて、画像識別を実施した。表 4 に使用したプログラムの解説、表 5 に設定したハイパーパラメータを、図 5 に Epoch と Accuracy の変化を、図 6 に Epoch と Loss の変化をそれぞれ示す。

2.2 実験 2

CIFAR-100 データセットを使用し、EfficientNetB0 を用いて、畳み込み CNN を用いて、画像識別を実施した。表 6 に使用したプログラムの解説、表 7 に設定したハイパーパラメータ、図 7 に Epoch と Accuracy の変化、図 8 表に Epoch と Loss の変化、表 8 に class 単位の precision・ recall・ f1-score・support、図 9 に混同行列のヒートマップをそれぞれ示す。

表 4 実験 1 で使用したプログラムの解説

	詳細
ライブラリとデータの準備	TensorFlow、matplotlib、numpy をインポート。
	CIFAR-10 データセットを読み込み、訓練データ (x_train, y_train) とテストデータ (x_test, y_test) に分割。
画像データの前処理と拡張関数	画像データに対するデータ拡張（Data Augmentation）［注 4］のための関数 image_augmentation を定義。
モデルのトレーニング	TransferTrainer［注 5］クラスを使用して、モデルのトレーニングを実施。
	トレーニングのパラメータの設定（バッチサイズ、エポック数、オプティマイザ、データ拡張）。
	EfficientNetB5 をビルドし、トレーニングの準備を実施。
トレーニングと結果の表示	train 関数を呼び出し、モデルを指定された設定でトレーニングを実施。トレーニング結果である Validation Accuracy［注 6］や Loss を trainer.history［注 7］に記録。
	学習率の変化やトレーニングの進捗は、showHistory 関数を使用してグラフ表示。
TPU 利用	TPU が利用可能な場合、使用してモデルのトレーニングを実施。TPU を利用しない場合、通常の設定でトレーニングを実施。

表 5 実験 2 ハイパーパラメーターの設定

hyperparameter	Value	hyperparameter	Value	hyperparameter	Value
width_shift_range	0	batch_size	128	freeze_ratio	0
height_shift_range	0	burnin_lr	0.0001	burnin_epochs	3
rotation_range	0	cutout_num	1	warmup_epochs	10
horizontal_flip	FALSE	cutout_size	0.25	flat_epochs	5
vertical_flip	FALSE	random_color	1	cooldown_epochs	20
Cval	0	label_smoothing	0	min_lr	0.0001
cutout_size	None	Resolution	1	max_lr	0.03
cutout_num	1	model_name	'EfficientNetB5'	dropout	0.5
random_color_p	0				

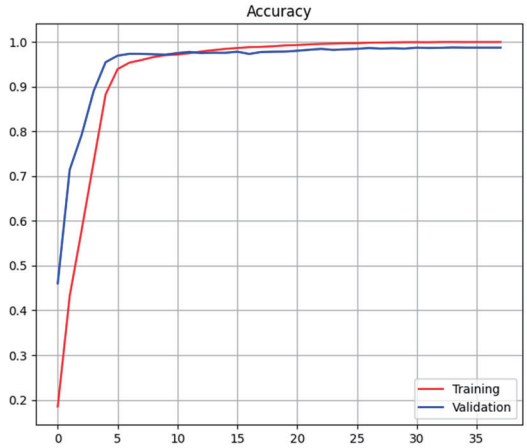

図 5　実験 1 の学習曲線

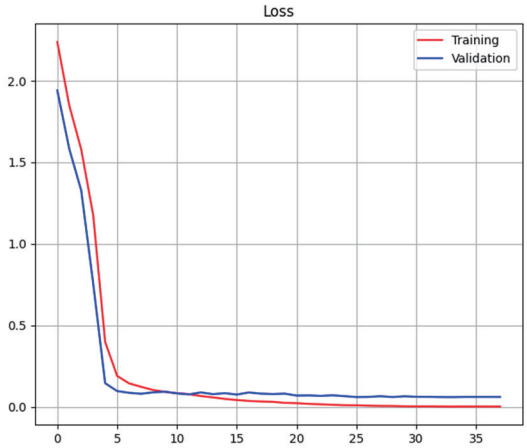

図 6 Epoch と Loss の関係

表 6　実験 2 で使用したプログラムの解説

	詳細
1　ライブラリとデータの準備	1.1 TensorFlow、keras、efficientnet をインポート。
2 CIFAR-100　データセットの読み込み	2.1　CIFAR-100 データセットをトレーニング用とテスト用に分割し、それぞれ (x_train, y_train) と (x_test, y_test) に格納して読み込む。
3　データの前処理	3.1　EfficientNet モデルに合わせて、preprocess_input 関数を使用して画像データに対して前処理を実施。
4 EfficientNet　モデルの構築	4.1　モデルの入力層として、Input(shape=(32, 32, 3)) を指定し、それを input_tensor［注 8］に代入。
	4.2　EfficientNetB0 モデルを呼び出して、事前学習済みの特徴抽出器として使用。
	4.3　プーリング層を無効にし、出力の形状を指定。
	4.4　ベースモデルの出力をグローバル平均プーリング層に渡し、それに続けて全結合層を追加。
5　モデルのコンパイル	5.1　model.compile［注 9］メソッドを使用して、モデルのコンパイル。損失関数と最適化アルゴリズム、評価指標（精度）を指定。
6　モデルのサマリー表示	6.1　model.summary()［注 10］を呼び出して、モデルの概要情報を表示。
7　モデルのトレーニング	7.1　model.fit［注 11］メソッドを使用して、モデルをトレーニングする。トレーニングデータ (x_train, y_train) を使用し、バッチサイズとエポック数を指定。
	7.2　トレーニングの進行状況と損失、精度などの情報が history 変数に格納。

表 7 実験 2 ハイパーパラメーターの設定

Hyperparameter	Value
Lr	0.001
batch_size	32
Epochs	10

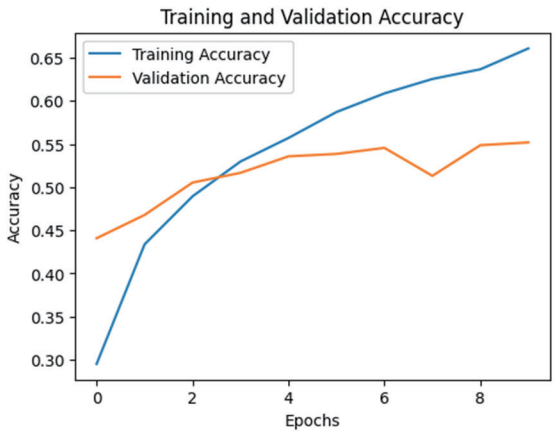

図 7 実験 2 の学習曲線

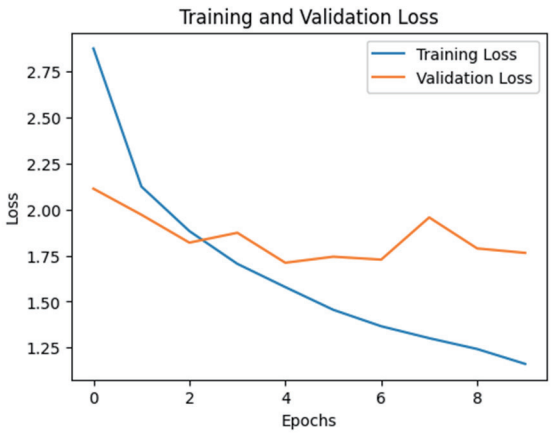

図 8 Epoch と Loss の関係

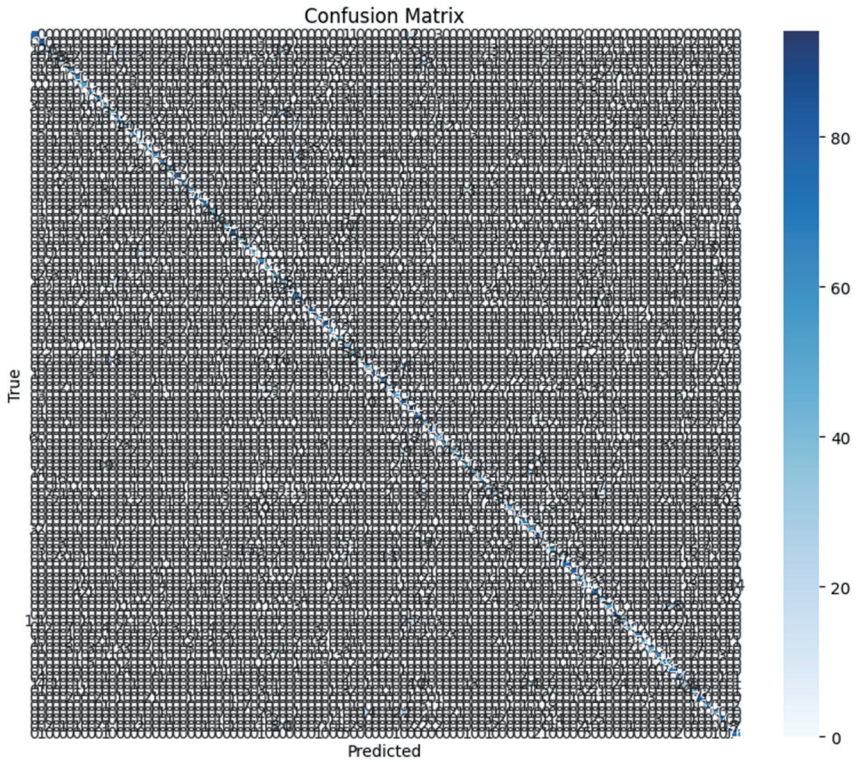

図 9 実験 2 の処理結果　混同行列のヒートマップ

表 8　実験 2 の処理結果（class 単位の precision・ recall・ f1-score・support）

Class	precision	recall	f1-score	support	Class	precision	recall	f1-score	support
0	0.72	0.77	0.74	100	50	0.31	0.27	0.29	100
1	0.58	0.81	0.68	100	51	0.62	0.55	0.59	100
2	0.45	0.26	0.33	100	52	0.51	0.81	0.63	100
3	0.47	0.18	0.26	100	53	0.51	0.92	0.65	100
4	0.32	0.18	0.23	100	54	0.58	0.75	0.66	100
5	0.62	0.61	0.61	100	55	0.2	0.43	0.27	100
6	0.63	0.67	0.65	100	56	0.67	0.86	0.75	100
7	0.73	0.52	0.61	100	57	0.7	0.54	0.61	100
8	0.74	0.69	0.72	100	58	0.77	0.6	0.67	100
9	0.71	0.68	0.69	100	59	0.68	0.42	0.52	100
10	0.44	0.48	0.46	100	60	0.84	0.64	0.73	100
11	0.33	0.37	0.35	100	61	0.65	0.6	0.62	100
12	0.69	0.59	0.63	100	62	0.73	0.45	0.56	100
13	0.57	0.4	0.47	100	63	0.6	0.49	0.54	100
14	0.59	0.59	0.59	100	64	0.45	0.25	0.32	100
15	0.5	0.46	0.48	100	65	0.3	0.28	0.29	100
16	0.63	0.59	0.61	100	66	0.7	0.3	0.42	100
～	～	～	～	～	～	～	～	～	～
～	～	～	～	～	～	～	～	～	～
40	0.6	0.55	0.58	100	90	0.52	0.75	0.62	100
41	0.63	0.82	0.71	100	91	0.53	0.74	0.62	100
42	0.67	0.6	0.63	100	92	0.66	0.29	0.4	100
43	0.51	0.58	0.54	100	93	0.29	0.45	0.36	100
44	0.25	0.37	0.3	100	94	0.84	0.79	0.81	100
45	0.4	0.38	0.39	100	95	0.61	0.64	0.62	100
46	0.44	0.24	0.31	100	96	0.48	0.47	0.48	100
47	0.6	0.53	0.56	100	97	0.48	0.64	0.55	100
48	0.83	0.8	0.82	100	98	0.33	0.18	0.23	100
49	0.83	0.63	0.72	100	99	0.57	0.76	0.65	100
				Accuracy				0.55	10000
				macro average		0.57	0.55	0.55	10000
				weighted average		0.57	0.55	0.55	10000

2.3 実験 3 で使用したプログラムの解説

CIFAR-100 データセットを使用し、EfficientNetB0 をアンサンブル学習させ、畳み込み CNN を用いて、画像識別を実施した。表 9 に使用したプログラムの解説、表 10 に設定したハイパーパラメータ、図 10 に Epoch と Accuracy の変化、図 11 に Epoch と Loss の変化、図 12 に混同行列のヒートマップ、表 11 に class 単位の precision・recall・f1-score・support をそれぞれ示す。

表 9 実験 3 で使用したプログラムの解説

	詳細
1　ライブラリとデータの読み込み	1.1 TensorFlow、Keras、EfficientNet をインポート。
	1.2 CIFAR-100 データセットを読み込み、訓練データ (x_train, y_train) とテストデータ (x_test, y_test) に分割。
2　データの前処理	2.1 EfficientNet モデルの要件に合わせて、ラベルを one-hot エンコーディング形式［注 12］に変換。
3　アンサンブル学習のためのモデルの作成	3.1 num_models［注 13］数 EfficientNetB0 ベースのモデルを作成。今回は 5 モデルを作成。
	3.2 各モデルは、EfficientNetB0 の事前学習済み重みを利用し、最後に 100 クラスの分類を行う Dense 層［注 14］で終了。
	3.3 各モデルは異なる初期化の重みを持ち、それぞれが独自の訓練を受ける。
4　各モデルの訓練	4.1 batch_size と epoch を設定し、各モデルを訓練データで訓練。
	4.2 各モデルの訓練の進行状況（accuracy と loss）は histories リストに保存。
5　テストデータでの予測とアンサンブル	5.1 各モデルに対してテストデータで予測を行い、結果を y_preds リストに保存。
	5.2 y_preds リスト内の各モデルの予測を平均し、最終的なアンサンブル予測 y_pred_avg を計算。
6　評価と可視化	6.1 混同行列を計算し、ヒートマップとして可視化。
	6.2 クラスごとの評価指標（precision［注 15］、recall［注 16］、f1-score［注 17］）を計算し、classification_report を表示。
	6.3 各モデルの訓練中の accuracy と loss を可視化。

表 10 実験 3 ハイパーパラメータ―の設定

Hyperparameter	Value
Lr	0.001
batch_size	32
Epochs	10
num_models	5

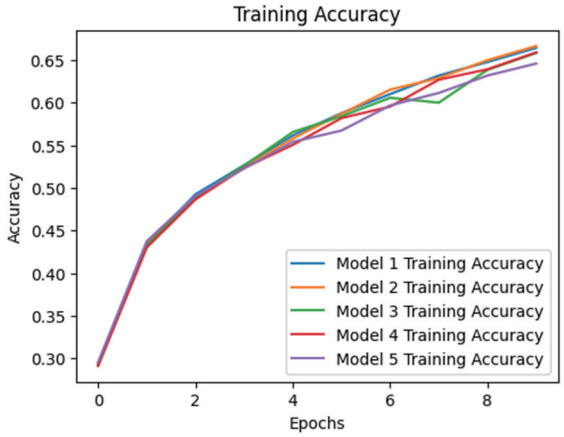

図 10　実験 3 の学習曲線

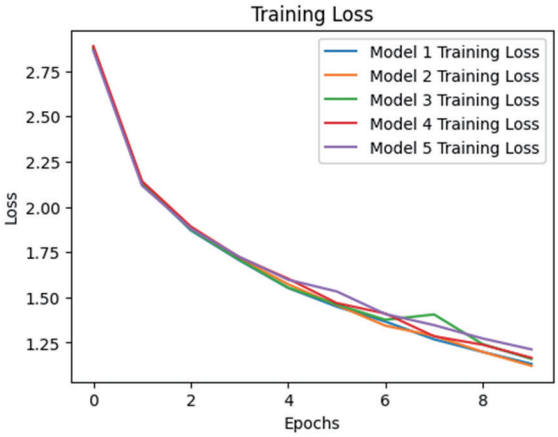

図 11　実験 3 Epoch と Loss の関係

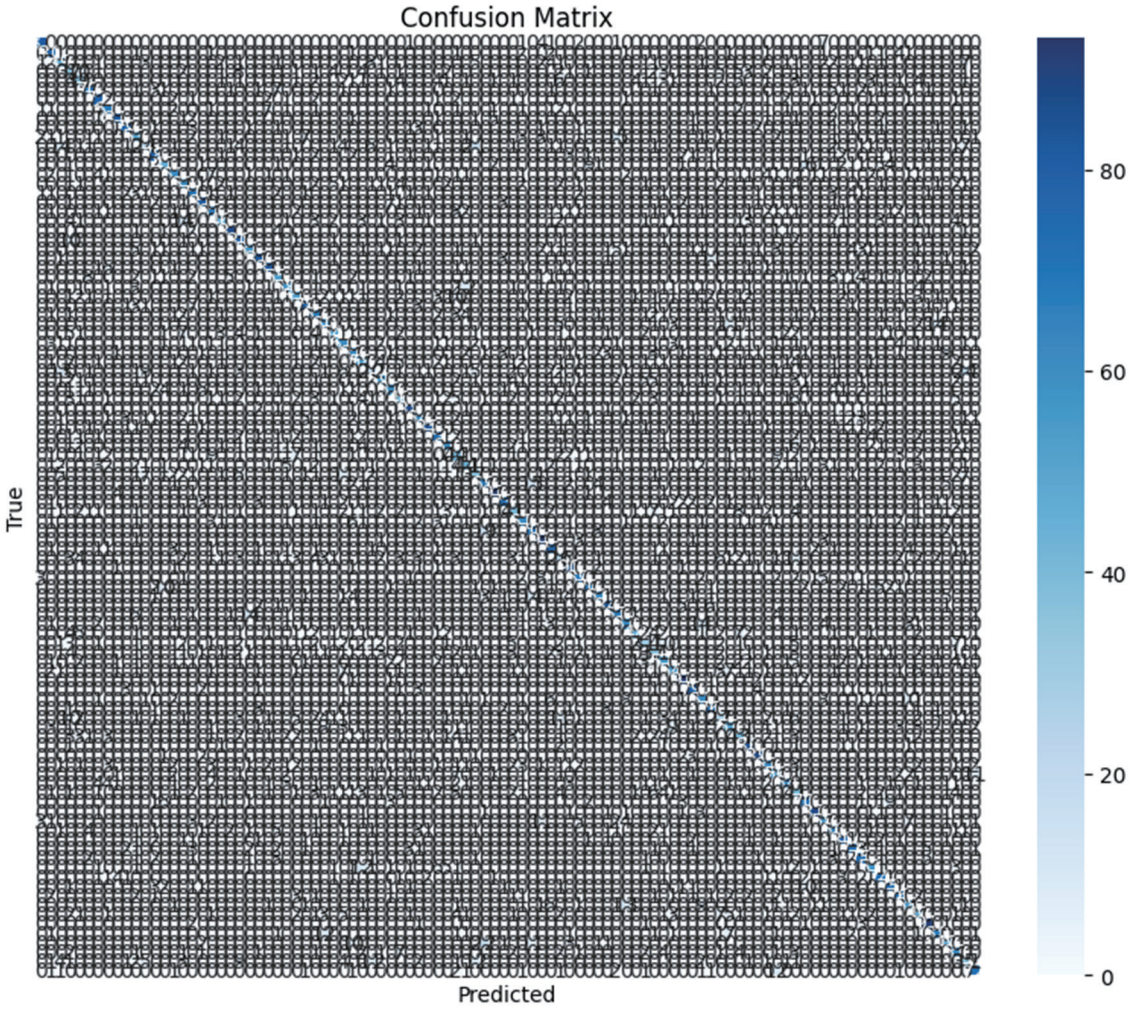

図 12　実験 3 の処理結果　混同行列のヒートマップ

表 11 実験 3 の処理結果（class 単位の precision・recall・f1-score・support）

class	precision	Recall	f1-score	support	Class	precision	recall	f1-score	support
0	0.88	0.81	0.84	100	50	0.48	0.41	0.44	100
1	0.7	0.8	0.75	100	51	0.6	0.62	0.61	100
2	0.49	0.49	0.49	100	52	0.63	0.7	0.66	100
3	0.32	0.37	0.35	100	53	0.71	0.91	0.79	100
4	0.46	0.39	0.42	100	54	0.65	0.77	0.7	100
5	0.76	0.7	0.73	100	55	0.26	0.26	0.26	100
6	0.69	0.77	0.73	100	56	0.7	0.9	0.79	100
7	0.68	0.67	0.68	100	57	0.7	0.69	0.7	100
8	0.84	0.81	0.83	100	58	0.77	0.7	0.73	100
9	0.84	0.79	0.81	100	59	0.75	0.61	0.67	100
10	0.51	0.53	0.52	100	60	0.76	0.84	0.8	100
11	0.44	0.24	0.31	100	61	0.68	0.67	0.67	100
12	0.73	0.78	0.75	100	62	0.65	0.63	0.64	100
13	0.6	0.47	0.53	100	63	0.76	0.48	0.59	100
14	0.67	0.61	0.64	100	64	0.48	0.31	0.38	100
15	0.53	0.62	0.57	100	65	0.38	0.37	0.38	100
16	0.63	0.71	0.67	100	66	0.5	0.57	0.54	100
～	～	～	～	～	～	～	～	～	～
～	～	～	～	～	～	～	～	～	～
38	0.44	0.46	0.45	100	88	0.7	0.57	0.63	100
39	0.85	0.88	0.87	100	89	0.64	0.74	0.69	100
40	0.68	0.64	0.66	100	90	0.78	0.75	0.77	100
41	0.89	0.85	0.87	100	91	0.68	0.76	0.72	100
42	0.66	0.73	0.69	100	92	0.62	0.58	0.6	100
43	0.71	0.63	0.67	100	93	0.47	0.49	0.48	100
44	0.35	0.44	0.39	100	94	0.83	0.91	0.87	100
45	0.53	0.44	0.48	100	95	0.68	0.68	0.68	100
46	0.4	0.37	0.39	100	96	0.6	0.49	0.54	100
47	0.58	0.69	0.63	100	97	0.56	0.63	0.59	100
48	0.88	0.93	0.9	100	98	0.36	0.35	0.36	100
49	0.82	0.73	0.77	100	99	0.72	0.72	0.72	100
				Accuracy				0.55	10000
				macro average		0.57	0.55	0.55	10000
				weighted average		0.57	0.55	0.55	10000

3 考察

3.1 実験 1 の考察

バーンイン（Burn-in）［注 18］のフェーズ（エポック 1-3）にて、クラス分類の訓練が行われ、バリデーション精度（val_acc）は 0.4601 から 0.7924 に向上した。次に、Fine-tuning フェーズ（エポック 4-35）では、クラス分類の訓練が行われ、バリデーション精度は徐々に向上した。最終的に、バリデーション精度は最高で 98.72%に達した。学習率は Warmup フェーズ［注 19］で増加し、Flat フェーズで一定の高い値を維持し、Cooldown フェーズで減少した。これにより、学習が効率的に進行したと考えられる。また、ドロップアウト率が 0.5 と設定されており、過学習を抑制するための正則化が機能している。さらに、ランダムなデータ拡張として水平反転、カラー変換、Cutout が実施され、モデルの汎化性能を向上させるのに役立っている。

3.2 実験 2 の考察

クラス 1 で Precision が 0.58、Recall が 0.81 で、F1-score が 0.68 となった。これは、クラス 1 の正確さはやや低いが、クラス 1 のインスタンスをほぼすべて識別できることを示している。また、クラス 4 では Precision が 0.32、Recall が 0.18、F1-score が 0.23 と、性能が低い結果となった。全体的な Accuracy は 0.55 で、比較的低い水準であり、モデルはまだ改善の余地がある。

3.3 実験 3 の考察

Precision に関しては、クラス 0 は 0.88 であり、高精度で分類しているが、クラス 11 が 0.44 と低い Precision で分類している。Recall は、クラス 0 が最高で 0.81、最低はクラス 11 で 0.24 となった。F1-score は、クラス 0 が 0.84 で、クラス 11 が 0.31 で、最も低い値となった。全体の Accuracy は約 0.64 で、中程度の性能が出たと言える。今回の実験では、アンサンブル学習を用いる事により、実験 2 よりよりも Accuracy が 0.09 向上した。このようにアンサンブル学習を用いることにより、モデルの予測の信頼性が高まることが分かった。

4 まとめ

本研究での、結果を表 12 に纏める。

表 12　実験 1 から 3 のまとめ

実験	まとめ
実験 1	・Warmup フェーズでは、バリデーション精度が向上。 ・Fine-tuning フェーズでは、に高レベルに達した。 ・学習率スケジュールが効果的に設定され、過学習対策が機能した。 ・ランダムなデータ拡張も実施され、モデルの汎化性能が向上。
実験 2	・クラスごとの Precision、Recall、F1-score が異なり、クラス 1 では高い Recall を達成したが、クラス 4 の性能は低く、モデルの改善の余地がある。
実験 3	・クラスごとの Precision、Recall、F1-score の差異が観察され、アンサンブル学習を使用した実験により、全体的な Accuracy は実験 2 より 0.09 向上したが、クラス 11 の識別精度が低く改善が必要。

全体的には、クラスごとの性能の不均衡を考慮し、モデルの最適化とデータ拡張の戦略を検討する必要がある。

（注記）

1.入力画像に対してフィルター（カーネルとも呼ばれる）を適用し、畳み込み演算を実行する層。

2.畳み込み層の出力から情報を圧縮し、計算効率を向上させる層。

3.モデルの過学習を防ぐために使用される層。

4.限られたトレーニングデータでモデルの汎化性能を向上させるために使用される。

5. 転移学習（Transfer Learning）に関連した機械学習やディープラーニングのコードやライブラリ内で使用されるクラス。

6. 検証データに対するモデルの正確さを示す割合。

7.各エポックに、トレーニングデータに対する正確度や精度が記録される。

8.ディープラーニングモデルの入力データを表現するためのテンソル。

9. 機械学習モデルをトレーニングする前に、モデルのトレーニングプロセスを設定し、モデルをトレーニング可能な状態に準備するためのメソッド。

10. ディープラーニングモデルの概要を表示するためのメソッド。

11. 機械学習モデルのトレーニングプロセスを開始するためのメソッド。

12. カテゴリカルデータを数値データに変換するための手法。

13. アンサンブル内のbase modelの数。

14. 密な層、全結合層。

15. 正と予測されたサンプルのうち、実際に正であるサンプルの割合。

16. 実際にPositiveであるサンプルのうち、モデルが正しくPositiveと予測できたサンプルの割合。

17. PrecisionとRecallの調和平均。

18. 特定のプロセスやシステムが安定状態に達するまでの初期の期間または過程。

19. モデルの性能を向上させるために、一定のデータを使用してモデルを予めトレーニングまたは運用状態に持っていくプロセス。

（引用文献）

[1] IBM Corporation :「What are convolutional neural networks?」、https://www.ibm.com/topics/convolutional-neural-networks（2023/9/8閲覧）。

[2] stanford university :「Convolutional Neural Network」、http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/（2023/9/8閲覧）。

[3] 東京大学 大学院情報理工学系研究科 中山 英樹 :「深層畳み込みニューラルネットワークによる画像特徴抽出と転移学習」、http://www.nlab.ci.i.u-tokyo.ac.jp/pdf/CNN_survey.pdf（2023/1/7 閲覧）。

[4] Ledge inc. :「転移学習とは｜、メリット・デメリット・ファインチューニングの意味」https://ledge.ai/transfer-learning/（2023/1/7 閲覧）。

[5] 株式会社ベネッセコーポレーション :「転移学習とは？ディープラーニングで期待の「転移学習」はどうやる？」https://udemy.benesse.co.jp/data-science/deep-learning/transfer-learning.html（2023/1/7 閲覧）。

[6] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, https://arxiv.org/abs/1905.11946（2023/8/30 閲覧）。
[7] アイティメディア株式会社：「CIFAR-10：物体カラー写真（乗り物や動物など）の画像データセット」https://atmarkit.itmedia.co.jp/ait/articles/2006/10/news021.html（2023/9/8 閲覧）。
[8] Alex Krizhevsky,toronto.edu: The CIFAR-10 datase, https://www.cs.toronto.edu/~kriz/cifar.html（2023/9/8 閲覧）。
[9] Alex Krizhevsky,toronto.edu: The CIFAR-100 datase, https://www.cs.toronto.edu/~kriz/cifar.html（2023/9/8 閲覧）。