卒 業 研 究 報 告 題 目

Swin-Transformerによる顔画像の感情認識

Swin-Transformer-based Facial Emotion Recognition

指 導 教 員　　出口 利憲　教授

岐 阜 工 業 高 等 専 門 学 校　電気情報工学科

2021E02 石丸 詩乃

令 和 8 年（ 2 0 2 6 年 ）　　2 月 1 3 日 提 出

Abstract

Facial expression recognition is an important research topic in the field of human–computer interaction, as it enables systems to understand human emotions from facial images. In recent years, convolutional neural networks (CNNs) and Transformer-based models have been widely applied to this task. Among them, the Swin Transformer has attracted attention due to its ability to efficiently capture both local and global features using a window-based self-attention mechanism.

In this study, we investigate the performance of a facial expression recognition model based on Swin Transformer (Tiny) using the RAF-DB public dataset. A pretrained model on ImageNet is fine-tuned by replacing the final classification layer to perform six-class emotion classification. The training process is analyzed using learning curves, and the classification performance is evaluated using accuracy, confusion matrix, and class-wise metrics.

Experimental results show that the proposed model achieves approximately 90% accuracy on the test dataset. The analysis of the confusion matrix and evaluation metrics reveals that the model performs well overall, while differences in performance among emotion classes are observed. These results demonstrate the effectiveness of Swin Transformer for facial expression recognition and highlight remaining challenges related to class imbalance and inter-class similarity.

目 次

第1章　序論

近年、人工知能技術の発展に伴い、画像認識分野において深層学習を用いた手法が広く研究されている。中でも、顔表情認識は人間の感情状態を推定する技術として、ヒューマン・コンピュータ・インタラクション、監視システム、医療支援など様々な分野への応用が期待されている。

従来、顔表情認識には畳み込みニューラルネットワーク（CNN）を用いた手法が多く提案されてきた。CNN は局所的な特徴抽出に優れている一方で、画像全体の長距離依存関係を捉えることが難しいという課題がある。これに対し、Transformer は自己注意機構（Self-Attention）を用いることで、入力全体の関係性を考慮した特徴表現を学習できるモデルとして注目されている。

Vision Transformer（ViT）の登場以降、画像認識分野においても Transformer を基盤とした手法が活発に研究されている。その中でも、Swin Transformer は画像を固定サイズのパッチに分割し、Window 単位で Self-Attention を計算することで、計算量を抑えつつ局所的特徴と大域的特徴を効率的に統合できるモデルである。この特性により、Swin Transformer は物体認識やセグメンテーションのみならず、顔表情認識においても有効であることが報告されている。

しかしながら、Swin Transformer を用いた既存研究においては、学習済みモデルを用いたファインチューニングによる性能評価や、学習曲線や混同行列を用いた学習過程およびクラスごとの挙動を詳細に示した検討は十分とは言えない。

そこで本研究では、Swin Transformer（Tiny）を用いた顔表情認識モデルを構築し、公開データセットである RAF-DB を用いて性能評価を行うことを目的とする。ImageNet で事前学習されたモデルを基盤としてファインチューニングを行い、学習曲線、混同行列および各種評価指標を用いて、モデルの学習挙動とクラスごとの認識性能を明らかにする。

本研究により、Swin Transformer の顔表情認識における有効性を示すとともに、クラス不均衡や表情間の類似性に起因する課題についても考察を行う。

第2章　ニューラルネットワーク

2.1　ニューロン[1)]

深層学習の基本となるのが、人間の脳細胞であるニューロンを数理モデル化した人工ニューロンである。

ニューロンの基本構造をFigure 2.1に示す。ニューロンは細胞核の周辺部に樹状突起と呼ばれる、他のニューロンからの信号に反応する入力素子をもっており、その入力に応じて、軸索に信号が伝達され軸索末端から他のニューロンへ信号が伝達される。これをFigure 2.2に示す人工ニューロンでは、演算ユニットとしてモデル化している。複数の入力信号 x_i に重み w_i を掛け合わせ、それらの総和にバイアス値 b を加えた値を活性化関数 f に入力することで、人工ニューロンの出力値が決定される。

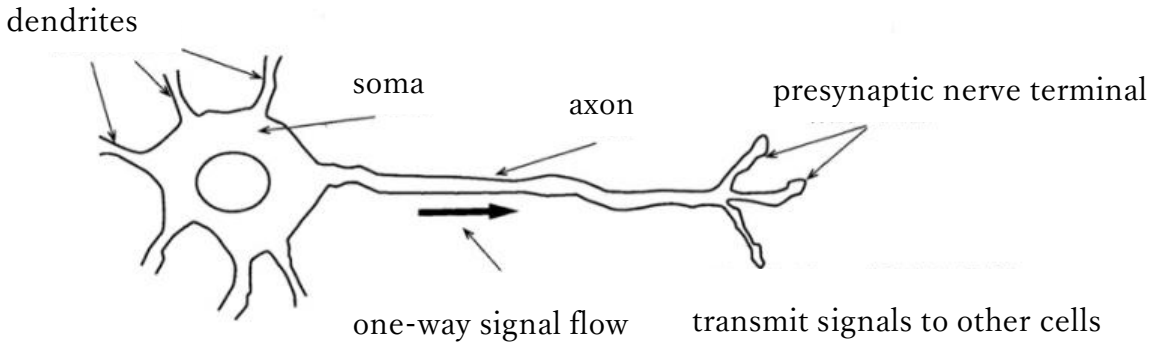


Figure 2.1　Schematic diagram of neuron.[1)]

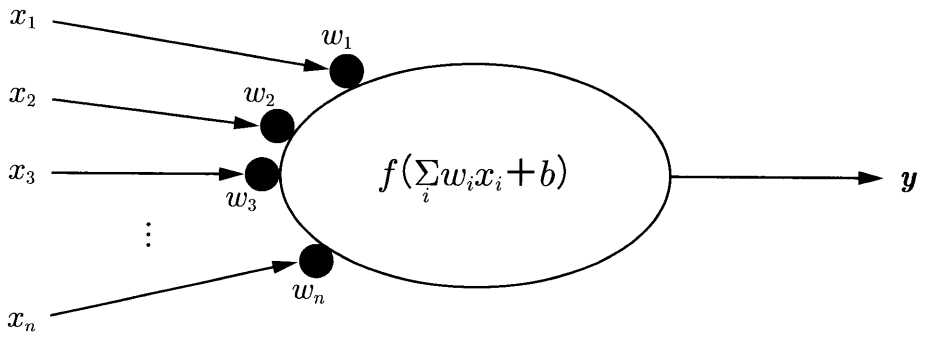


Figure 2.2　Neuron modal.[1)]

2.2　ニューラルネットワーク

ニューラルネットワークは、神経細胞のモデルである人工ニューロンを相互に結合したネットワークであり、人間の神経細胞の回路をモデル化したものである。ニューラルネットワークでは、決められた規則に従って複数の人工ニューロンを結合し、全体として入力信号に対する出力信号を生成する。[2)]

ニューラルネットワークの基本構造として、最も単純な形式は多層パーセプトロンである。Figure 2.3 に基本となる 3 層構造の多層パーセプトロンを示す。複数の人工ニューロンを層状に接続しており、入力層、中間層 (隠れ層)、出力層から構成される。

入力層は、ニューラルネットワークへの入力データを受け取り、中間層へ渡す役割を持つ。中間層は入力層と出力層の間に位置する層であり、ニューラルネットワークの構造によっては複数存在する。中間層では、各ニューロンが入力に重みを掛け合わせ、それらの総和にバイアスを足し合わせたものに活性化関数を適用し、その結果を次の層に伝達する。最終的なニューラルネットワークの出力は、出力層で決定される。

多層パーセプトロンが発展したものが深層学習である。層の種類が増え、深いネットワークとなっているが、基本的な考え方は同じである。層の種類が増えることで、ネットワークの表現力が向上するが、学習は複雑になる。[1)]

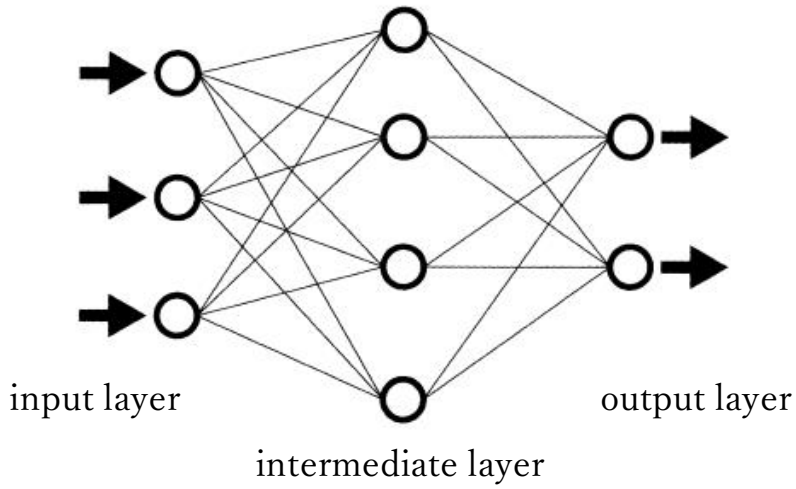


Figure 2.3　Neural Network.[1)]

2.3 勾配降下法 [1)]

学習における誤差最小化のための学習パラメータの更新には、一般的に次の式 (2.1) で表される 1 階微分を用いた学習法である勾配降下法が用いられる。

$$w_i^{(t+1)} = w_i^{(t)} - \eta \frac{\partial E(\boldsymbol{x})}{\partial w_i^{(t)}} \tag{2.1}$$

式 (2.1) における $w_i^{(t)}$ は更新前の学習パラメータ、$w_i^{(t+1)}$ は更新後の学習パラメータを示し、誤差関数 E の偏微分 $\partial E(\boldsymbol{x})/\partial w_i$ で表される勾配に学習率 η をかけた値が学習パラメータの更新に使用される。学習率は経験的に決められ、一般的に 0.001 や 0.0001 が用いられる。勾配の大きさに応じて学習率をかけて、逆方向にパラメータを更新することで誤差を最小化する。

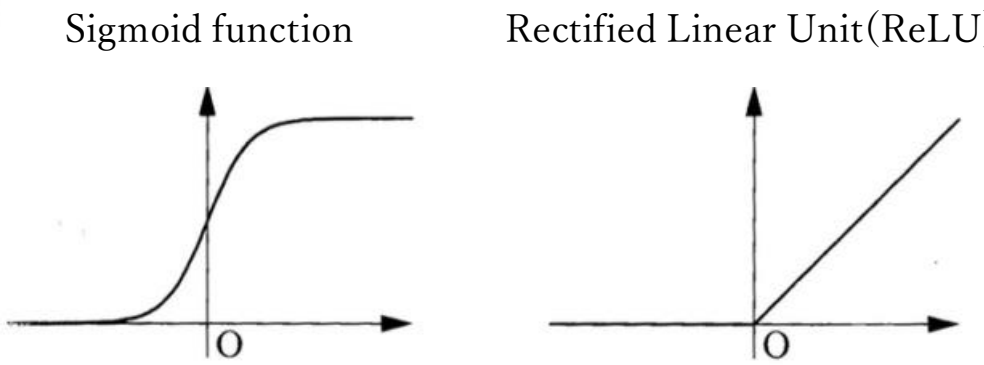
全サンプルの学習の繰り返し回数はエポックと呼ばれ、通常の学習では 10〜100 エポックの繰り返し学習を行うことが一般的である。また、エポックごとに毎回学習データの順番をランダムに変えるのが確率的勾配降下法である。異なる順番で学習することで局所解に捕まりにくくなり、学習が容易になる。

しかし、1 サンプルごとにパラメータを更新するには膨大な計算が必要であり、効率的な計算が難しい。そこで、深層学習では一般的に式 (2.2) に示すように、数十から数百程度の学習サンプルの勾配をまとめて計算し、それらの勾配の平均をパラメータ更新に利用する、ミニバッチ確率的勾配降下法を用いる。

$$w_i^{(t+1)} = w_i^{(t)} - \frac{1}{k}\eta \sum_{j=1}^{k} \frac{\partial E(\boldsymbol{x}_j)}{\partial w_i^{(t)}} \tag{2.2}$$

2.4 活性化関数 [1)]

2.1 節で述べたように、活性化関数とは、各ニューロンにおいて入力の重み付き和に適用される関数である。全結合層は線形変換であるため、多段に重ねるだけではネットワーク全体は線形変換にとどまる。そのため、層の間に非線形活性化関数を導入し、レイヤご



$$f(x) = \frac{1}{1 + \exp(-x)} = \frac{\tanh(x/2) + 1}{2}$$

$$f(x) = \max(x, 0)$$

Figure 2.4　Activation function.[1)]

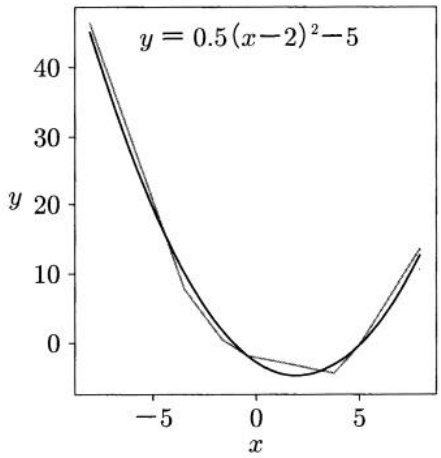
との変換を非線形化する必要がある。

Figure 2.4 に主な活性化関数を示す。二つの関数のうち、右に示す正規化線形ユニット (ReLU) は、現在最も標準的に使われている活性化関数である。左に示すシグモイド関数は、学習時に勾配消失が起こりやすいという欠点を持つが、ReLU は入力が負の場合には 0、正の場合には入力をそのまま出力という単純な構造であるにもかかわらず、シグモイド関数と同様の役割を果たすことができる。

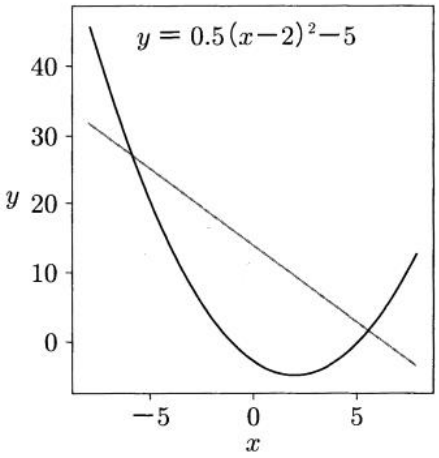
Figure 2.5 に簡単な二次関数を全結合層 3 層で近似した場合の、ReLU を使った場合と、ReLU 含め活性化関数を一切使っていない場合の比較を示す。活性化関数を用いない場合、全結合層を多段に重ねても線形変換にとどまるため、直線による近似しか行えない。一方で、ReLU を用いた場合には、非線形性が導入され、より複雑な関数をうまく近似ができていることが分かる。

2.5　誤差逆伝播法[1)]

誤差関数 E はネットワークの最終的な出力に基づいて定義されるため、各学習パラメータとの関係を単一の式で直接表すことは困難である。一方で、ニューラルネットワークは



(a) With ReLU



(a) Without ReLU

Figure 2.5　Function approximation with and without ReLU.[1)]

各層における変換の積み重ねとして構成されており、層間の関係は数式として明示的に表現できる。そこで、出力層から入力層へと誤差を逆方向に伝播させ、誤差関数 E と任意の学習パラメータとの間で、誤差勾配を計算することが可能である。これを行う方法が誤差伝播法である。

2.6　損失関数[4)]

多クラス分類問題において，入力 x を K 個のクラスのいずれかに分類する。ニューラルネットワークの出力層では、各クラスに対応するスコア (ロジット) u_k を出力する。これらのロジットを確率分布へ年関するために、ソフトマックス関数を用いる。ソフトマックス関数は式 (2.3) で定義される。

$$y_k = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)} \tag{2.3}$$

式 (2.3) で、y_k は入力 x がクラス k に属する確率を表す。ソフトマックス関数により、各出力値は 0 以上となり、さらに全クラスの出力の総和は 1 となるため，出力は確率分布として解釈できる。

多クラス分類における損失関数として、交差エントロピー損失を用いる。交差エントロピーは式 (2.4) で定義される。

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \log y_k \tag{2.4}$$

ここで w はモデルのパラメータ、N は学習データ数を示す。d_{nk} は教師ラベルを表す指標であり、入力 x_n がクラス k に属する場合に 1、それ以外の場合に 0 をとる。

交差エントロピー損失は、正解クラスに高い確立を割り当てるほど小さくなる性質を持つ。この性質により、モデルは正解クラスの確率を最大化するように学習される。

2.7 評価関数 [5)]

分類問題においてモデルの性能を評価するためには、予測結果と正解ラベルとの一致度を定量的に測定する必要がある。本節では，多クラス分類において一般的に用いられる評価指標について述べる。

1. Accuracy(正解率)

Accuracy は、分類モデルが全サンプルに対して正しく予測できた割合を示す指標である。二値分類の場合を、式 (2.5) に定義する。

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

・TP(True Positive): 正しく正例に分類されたサンプル数

・TN(True Negatibe): 正しく負例に分類されたサンプル数

・FP(False Positive): 誤って正例に分類されたサンプル数

・FN(False Negative): 誤って負例に分類されたサンプル数

2. Precision(適合率)

クラス k に対する Precision は、モデルがクラス k と予測したサンプルのうち、正しく

クラス k であった割合である。式 (2.6) に定義する。

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{2.6}$$

・TP_k : クラス k の正しく分類されたサンプル数

・FP_k : クラス k と誤って予測されたサンプル数

3. Recall(再現率)

クラス k に対する Recall は、実際にクラス k に属するサンプルのうち、正しく予測された割合である。式 (2.7) に定義する。

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{2.7}$$

・FN_k : クラス k のサンプルで誤って他クラスに分類された数

4. F1-score

F1-score は Precision と Recall の調和平均である。式 (2.8) に定義する。

$$F1_k = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.8}$$

F1-score は、Precision と Recall のバランスを総合的に評価できる指標であり、特にクラス不均衡が存在する場合に有効である。

5. 混同行列

混同行列は、分類モデルの予測結果と実際のクラスの対応関係を行列形式で表したものである。行を 実際のクラス、列を 予測クラス とすることで、各クラスの分類性能を視覚的に把握できる。

6. 多クラス分類への拡張

二値分類で定義した指標は、多クラス分類に拡張して利用できる。ただし、クラス数 $C > 2$ では TN の概念を直接扱うことは困難である。この場合、混同行列 CM の主対角線上の正解数を分子に、混同行列の全要素の合計を分母に取ることで Accuracy を計算す

る。式 (2.9) に定義する。

$$Accuracy = \frac{\sum_{i=1}^{C} CM_{i,i}}{\sum_{i=1}^{C}\sum_{j=1}^{C} CM_{i,j}} \tag{2.9}$$

・実際のクラス i がクラス j に分類されたサンプル数

・C: クラス数

多クラス分類においても、Precision、Recall、F1-score はクラス k ごとに計算し、必要に応じてマクロ平均やマイクロ平均を取ることで全体の評価指標として利用できる。

2.8 過学習 [5)]

過学習とは、機械学習モデルが訓練データに過度に適合し、未知のデータに対する汎化性能が低下する現象である。過学習が発生すると、訓練データ上では高い精度を示す一方で、テストデータや実際の応用においては正しい予測が困難になる。

過学習の主な原因には、学習データ不足、学習データの偏り、モデル構築の目的の不明確さが挙げられる。学習データが少ない場合、モデルは限られた情報から特徴を学習するため、訓練データ特有のノイズや偶然のパターンまで学習してしまう。データが偏っている場合も、特定条件に適応したモデルとなり、汎化性能が低下する。また、モデルの目的が明確でない場合、必要のないデータや偏ったデータまで学習に用いられることがあり、過学習のリスクが増加する。

過学習の検出には、訓練データ、検証データ、テストデータを分けて学習を進める方法が有効である。訓練データでモデルを学習させ、検証データで精度を確認しながら改善を行い、最後にテストデータで最終評価を行うことで、過学習の発生を確認できる。

過学習の予防には、学習データの増加や多様化が重要である。特にデータ拡張 (augmentation) は、既存のデータに回転や反転、ノイズ付加などを行うことで人工的に学習データを増やし、モデルの汎化性能向上に寄与する。また、正則化や早期終了 (early stopping) などの手法も、モデルの複雑さを制御し過適合を防ぐ効果がある。

第3章　Swin Transformer[3)]

3.1　Transformer

Transfomer は、2017 年に論文 “Attention is All You Need” により提案された、新たなモデルである。タイトル中の Attention(注意機構) とは、目的のタスクを解くのに重要な情報を選択する役割を担う。当時は、機械翻訳のための手法として提案されたモデルであるが、現在では、自然言語処理のさまざまなタスクに Transformer が活用されている。Transformer では、以前から用いられてきた代表的なネットワークである RNN や CNN を用いず、Self-Attention という機構を活用することで、離れた位置にあるトークン同士の関係を捉えつつ計算を並列化できる。

3.2　ViT(Vision Transformer)

コンピュータビジョン分野において、ResNet や EfficientNet をはじめとした CNN ベースのモデルが主流であったが、2020 年に Transformer ベースのモデルである、ViT(Vision Transformer) が登場した。ViT は、Transformer を画像に使うために最適化したものであり、畳み込みの代わりに Self-Attention 機構を用いることで、画像分類タスクで多くの SoTA(State-of-the-Art) を達成した。

3.2.1　パッチ分割

Figure 3.1 に示すように、ViT では入力画像を固定サイズのパッチ (Patch) に分割する。画像を分割するパッチサイズはハイパーパラメータとして設定され、これにより画像全体のパッチ数が決定される。各パッチは H(高さ) × W(幅) × C(チャンネル数) からなる 3 次元テンソルとして表され、これを flatten して 1 次元ベクトルに変換した後、Transformer に入力可能な次元のベクトル表現へ変換することで ViT への入力となる。

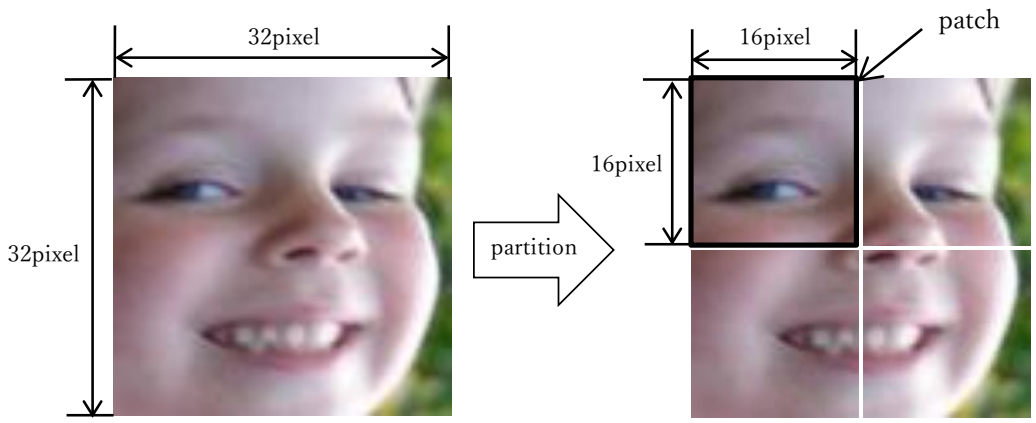


Figure 3.1　Patch Partitioning.

3.2.2　Self-Attention 機構

ViT の Encoder では、入力層で得られたパッチ埋め込みベクトルを入力にとり、クラストークンを含む特徴表現を出力する。この際、パッチ間の関係性を考慮するため、Self-Attention(自己注意) 機構が用いられている。Self-Attention は、ViT において画像全体の情報を統合する上で非常に重要な役割を持つ。

Self-Attention では、各パッチが他のパッチの特徴を参照し、それらとの類似度に応じて情報を取り込む。Figure 3.2 に Self-Attention 機構の全体像を示す。まず、入力られたパッチ埋め込みベクトルから情報を抽出するため、1 層の線形層を用いた埋め込みを行う。具体的には、同一の入力ベクトルに対して 3 つの線形層を用意し、それぞれの線形層で埋め込まれた各ベクトルをそれぞれクエリ $\boldsymbol{q}$(query)、キー $\boldsymbol{k}$(key)、バリュー $\boldsymbol{v}$(value) と呼ぶ。これらは同一の入力から生成されたものであるが、それぞれ異なる線形層を用いて埋め込まれているため、異なる表現となる。次に、クエリとキーの内積を求めることで各パッチ間の類似度を表す Attention Weight が得られる。この重みを用いてバリューの加重和を求めることで、Self-Attention の最終的な出力が計算される。すなわち、Self-Attention の出力は、Attention Weight とバリューの行列積として表現できる。

このように、Self-Attention では 1 つのパッチの特徴を計算する際に、画像内のすべてのパッチの情報を考慮する。そのため、局所的な情報に依存せず、画像全体の大域的な特

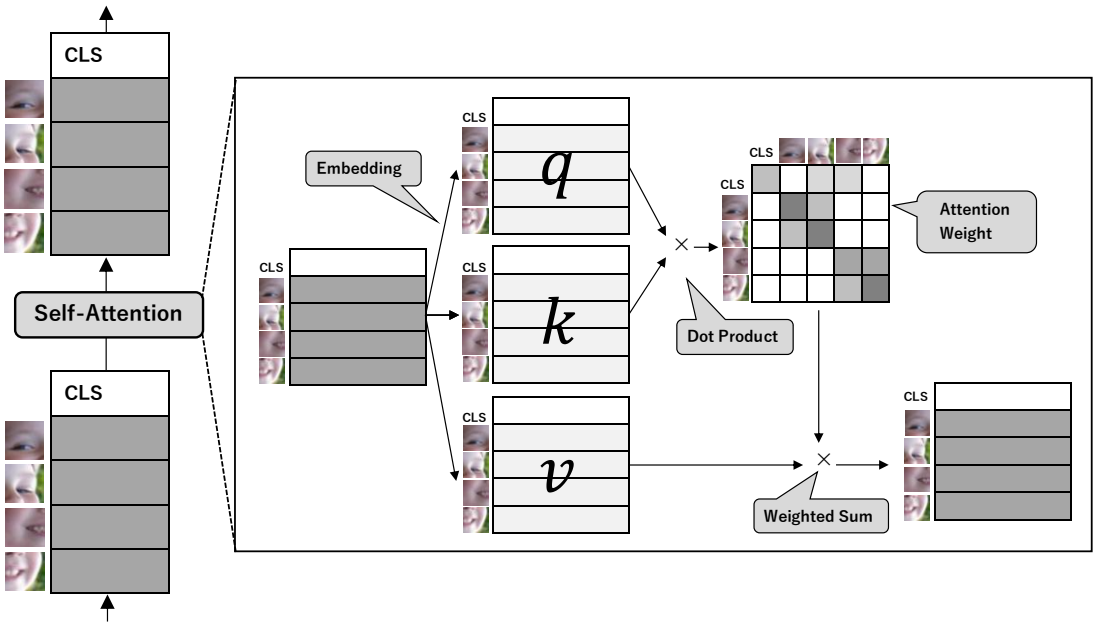


Figure 3.2　Self-Attention.[3)]

徴を学習することができる。

3.3　Swin Transformer

Figure 3.3 に Swin Transformer と ViT における Self-Attention の計算範囲の違いを示す。ViT では画像を特定サイズのパッチに分割し、画像全体に対して Self-Attention を適用するため、多様な物体スケールの変化を十分に捉えにくいという課題がある。そこで、Swin Transformer は、画像認識における多様な物体スケールに対応可能であり、かつ Self-Attention の計算量を削減する手法として提案された。

Swin Transformer では、画像を階層的に処理することで特徴マップを構築する。Figure 3.3(a) に示すように、浅い層では細かく分割されたパッチを用い、層が深くなるにつれてパッチを結合する。その際、Self-Attention の計算はあらかじめ定義された Window(局所窓) 内に限定される。例えば、Figure 3.3(a) の最下層では、16×16 に細かく分割したパッチを入力とし、4×4 に広く分割した Window 内に対してのみ Self-Attention を適用する。このように、Self-Attention を局所的な Window 内に限定しつつ、層を重ねることで参照範囲を段階的に拡大する構造により、計算量の削減と多様な物体スケールへの対応

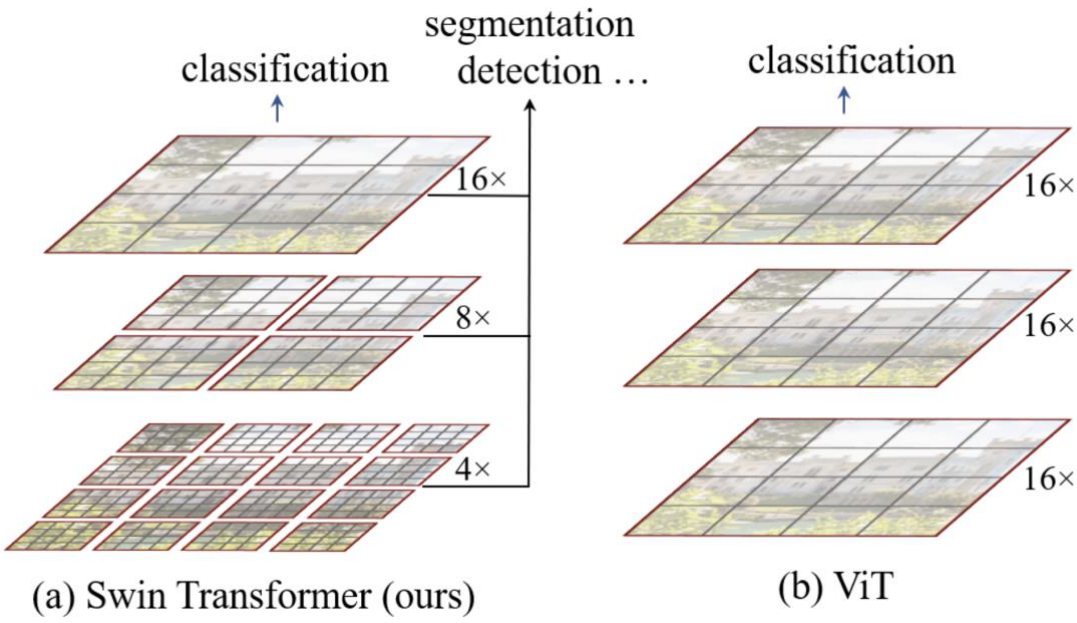


Figure 3.3　Self-Attention Computation Range in ViT and Swin Transformer.[3)]

を両立している。

3.3.1　ネットワーク構造

Tiny モデルを例とした Swin Transformer のネットワーク構造を Figure 3.4 に示す。Swin Transformer は、特徴マップのサイズを段階的に縮小する Stage 構造を持ち、CNN と同様に階層的な特徴抽出を行う。Figure 3.4(a) に示すように、Swin Transformer は 4 つの Stage から構成され、画像特徴量を段階的に抽出する。まず、Patch Partition により入力画像を重複しないパッチに分割する。

Stage1 では、分割した各パッチに Linear Embedding を適用し、パッチ特徴量を生成した後、Swin Transformer Block によりパッチ間の対応関係を学習する。Swin Transformer の内部構造を Figure 3.4(b) に示す。ブロック内では、層ごとに W-MSA(Window based Multi-head Self-Attention) と SW-MSA(Shifted Window based Multi-head Self-Attention) を交互に適用することで、局所的な Self-Attention を維持しつつ、Window 間における情報伝播を可能とする。W-MSA と SW-MSA については 3.3.4 項で述べる。

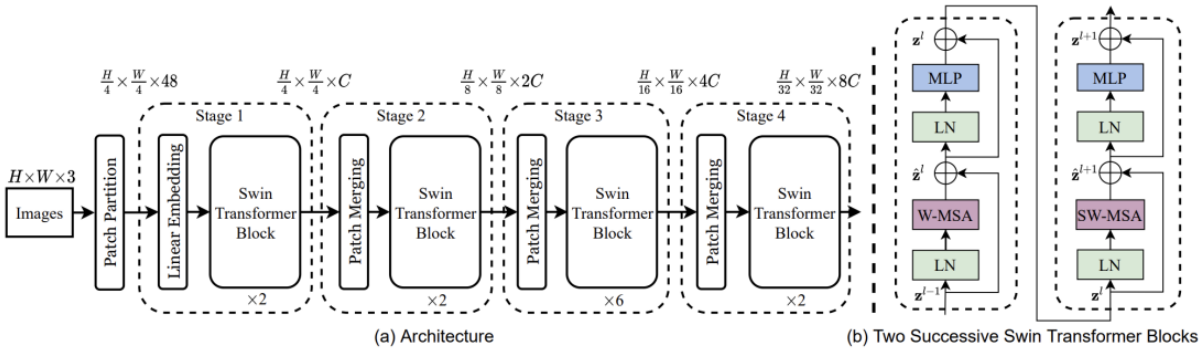


Figure 3.4　SwinTransformer Network Structure.[3)]

Stage2 以降では、Patch Merging を用いて特徴マップの空間サイズを縮小する。具体的には、隣接する 2×2 のパッチ特徴量をチャンネル C 次元方向に連結し、得られた $4C$ 次元の特徴量に対して、線形層を適用することで、チャンネルが $2C$ 次元になるよう線形変換をする。この処理と Swin Transformer Block によるパッチ間の対応関係の学習とを各 Stage で繰り返すことで、より高次の特徴量を獲得する。最終的に特徴マップのサイズを $\frac{H}{32} \times \frac{W}{32}$ ピクセルまで縮小した後に、Global Average Pooling および全結合層を用いてクラス分類を行う。

3.3.2　Patch Partition と Linear Embedding

前述の通り、Swin Transformer では、Patch Partition および Linear Embedding により入力画像をパッチ単位の特徴表現へ変換する。これらの処理は ViT と同様に固定サイズのパッチを用いる。

Patch Partition では、入力される $H \times W \times 3$ の RGB 画像を ViT のように重複しない固定サイズのパッチに分割する。Swin Transformer におけるパッチサイズのデフォルトは 4×4 である。分割された各パッチに対して Linear Embedding を適用し、パッチを特徴ベクトルへ変換する。実装上ではカーネルサイズ 4×4、スライド 4 の畳み込み演算として実現されている。

3.3.3 Patch Merging

Patch Merging は、各 Stage で得た特徴マップの空間サイズを縮小する処理である。入力となる特徴マップは $\frac{H}{4} \times \frac{W}{4} \times C$ の空間サイズを持ち、各グリッドが 1 つのパッチ特徴量に対応している。Patch Merging では各色で表された近傍 2×2 のパッチ特徴量をチャンネル方向に結合することで、空間サイズを半分にした $\frac{H}{8} \times \frac{W}{8} \times 4C$ の特徴マップを得る。最終的に、得た特徴マップのチャンネル次元が $2C$ になるように線形層を適用する。

3.3.4 Swin Transformer Block

Swin Transformer Block では、Self-Attention によってパッチ間の対応関係を捉えるために、Shifted Window と呼ばれる方法が用いられる。Figure 3.5 に Shifted Window の概略図を示す。特徴マップを複数の Window に分割し、各 Window 内でのみ Self-Attention を計算することで、計算量を削減しつつ局所的な特徴の抽出を可能にする。

W-MSA(Window-based Multi-Head Self-Attention) は偶数番目の層で用いられる。Figure 3.5 左に Self-Attention が適用される範囲を示す。Self-Attention の計算では、クエリ、キー、バリューを生成するための線形変換と、クエリとキー間の行列積が必要となる。このとき、$h \times w$ 個のパッチから構成される特徴マップ全体に対する Self-Attention の計算量は式 (3.1) で表される。

$$\Omega(MSA) = 4hwC^2 + 2(hw)^2C \tag{3.1}$$

式 (3.1) において、第 1 項は 4 つの線形層による計算量を表し、第 2 項は行列積にかかる計算量を表す。W-MSA では、$M \times M$(Figure 3.5 の例では 2×2) の Window に特徴マップを分割し、Window 内のパッチに対してのみ Self-Attention を適用する。その結果、Self-Attention 全体の計算量は式 (3.2) に示すように削減される。

$$\Omega(W\text{-}MSA) = 4hwC^2 + 2M^2hwC \tag{3.2}$$

一方、SW-MSA(Shifted Window-based Multi-Head Self-Attention) は奇数番目の層で用いられる。Figure 3.5 右に Self-Attention が適用される範囲を示す。SW-MSA は特徴

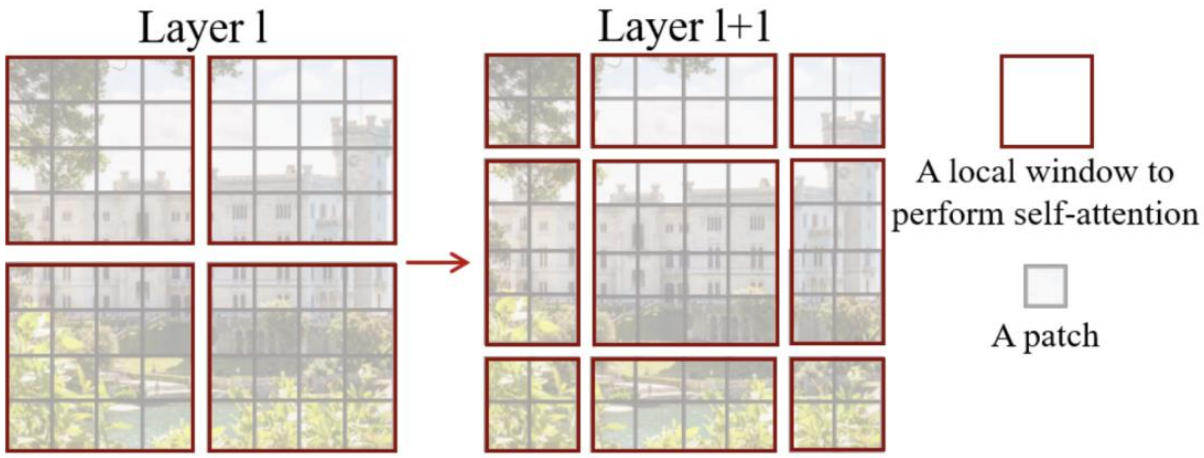


Figure 3.5　Shifted Window.[3)]

マップ上の Window の配置を $\frac{M}{2} \times \frac{M}{2}$ だけ移動させた W-MSA である。W-MSA と交互に適用することで、隣接する異なる Window 間の情報が間接的に統合され、W-MSA のみでは捉えられない隣接領域の関係性を学習可能となる。しかし、Figure 3.5 右のような構造では Window 数が増加し、その結果計算量が増加する問題が生じる。

そこで、Swin Transformer では、特徴マップを分割した Window を右下へ 2×2 移動する cyclic shift を適用する。Figure 3.6 に cyclic shift の概略図を示す。この操作により、Window の再配置を行い、計算効率を維持する。複数の Window が存在する場合には、不要な Window 間の Attention を抑制するためにマスク (mask) を導入し、特定の領域間の Attention Weight を 0 にする。Figure 3.6 の例では、左上の Window にはマスクを適用せず、右下の Window には各領域 (A、B、C、それ以外の領域) 間の Attention が計算されないようにマスクが適用される。各 Window で Self-Attention を計算した後、reverse cyclic shift を適用することで、特徴マップを元の位置に戻す。

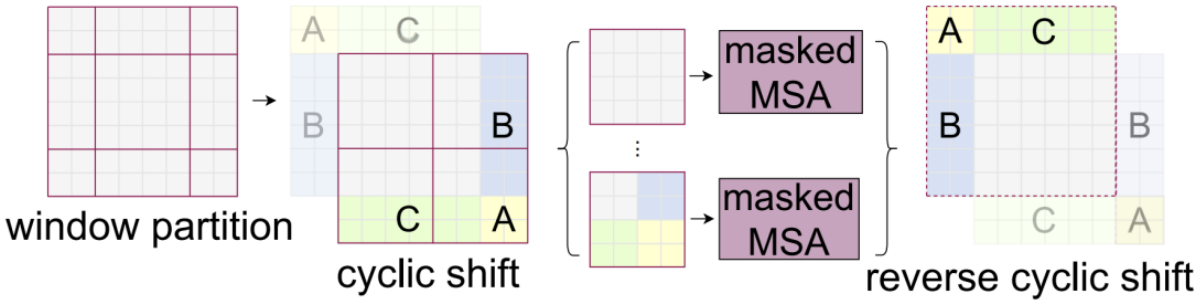


Figure 3.6 Cyclic Shift.[3)]

第4章　実験

4.1　実験目的

本研究では、Swin Transformer を用いた 6 クラス感情認識モデルの性能を、公開データセット (RAF-DB) に適用して評価することを目的とする。特に、学習済みモデルを基盤としてファインチューニングを行い、学習曲線や混同行列を用いたクラスごとの性能評価を実施する。また、最終的な認識精度だけでなく、各クラスごとの性能差を分析することで、少数クラスへのクラス不均衡への対応の適応性を把握する。

Swin Transformer を採用した理由は以下の通りである。:

・パッチ分割による局所特徴の効率的な抽出

・Window 単位での Self-Attention による計算効率の向上

・事前学習済みモデルの利用による少量データセットでの転移学習の効果

これらの特徴により、本研究では少量データでの高精度な感情認識を目指す。さらに、学習過程の可視化を通して、モデルの学習動作とクラスごとの性能を明確に示すことが本研究の主な目的である。

4.2　データセット

本研究では、データセットとして RAF-DB を用いた。Figure 4.1 に RAF-DB サンプル画像を示す。Real-world Affective Faces Database(RAF-DB) は、顔の表情を扱うデータセットである。このバージョンには、単一ラベルの基本表情 7 種類 (Happiness, Sadness, Surprise, Disgust, Anger, Fear, Neutral) でラベル付けされた約 12,000 枚の顔画像が含まれており、ラベル付けは 40 名の独立したラベル付け者によって行われている。このデータベースの画像は、被写体の年齢・性別・人種、頭部の向き、証明条件、部分的な隠蔽など多様な条件を持っている。また、データセットは学習用とテスト用に分かれている。

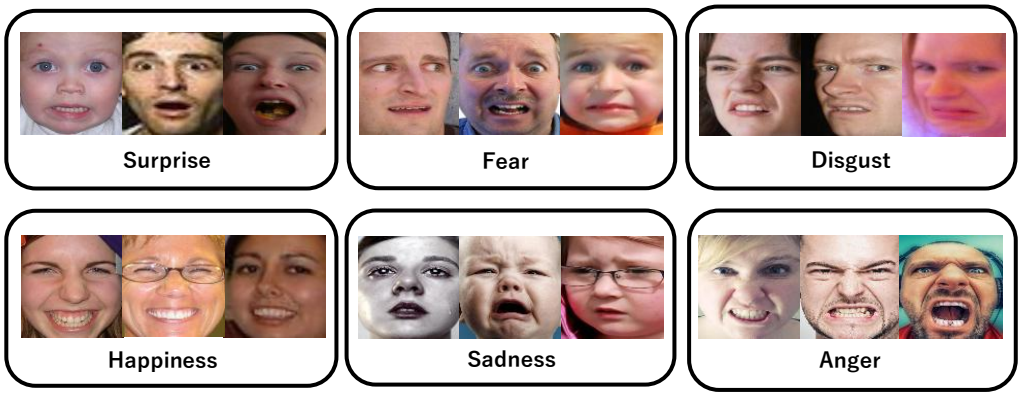


Figure 4.1　RAF-DB Sample.

4.2.1　データ構成

RAF-DB は 7 感情のデータセットを持つが、本研究では、Neutral を除いた基本 6 感情での感情認識モデルの学習および検証を行う。学習用画像には Train セット、検証用には Validation セット、テスト用には Test セットを使用する。ここで、Validation セットは Train セットの一部 (10%) を分割して作成し、学習中の過学習やモデルの汎化性能を評価するのに用いる。

Table 4.1 にクラスごとのサンプル数を示す。Table 4.1 から分かるように、Happiness は多数クラスであり、Fear や Digust は少数クラスである。

Table 4.1　Class-wise Sample Counts in RAF-DB.

Emotion	Total	Train set	Test set
Surprise	1,619	1,290	329
Fear	355	281	74
Disgust	877	717	160
Happiness	5,957	4,772	1,185
Sadness	2,460	1,982	478
Anger	867	705	162

4.2.2　データ前処理

全ての画像は、学習前に以下の前処理を行った。リサイズと正規化については一律に処理を行い、Agumentation はランダムに行った。

1. リサイズ：224 × 224 ピクセル
2. 正規化
3. Agumentation(Train セットのみ)

・ランダム左右反転 (50%)

・+−5 度のランダム回転

・明るさ、コントラスト、彩度のランダム揺らぎ (Color Jitter)

一方、Validation セットおよび Test セットでは、リサイズと正規化のみを適用した。これは、評価時にデータ拡張の影響を排除し、実際の汎化性能を確認するためである。

4.2.3　クラス不均衡対応

Figure 4.2 に示すように、クラスごとのサンプル数に偏りがある。この偏りに対応するため、学習時の損失関数に重み (Class Weight) を適用した。具体的には、PyTorch の CrossEntropyLoss に weight パラメータとして Class Weight を渡し、少数クラスの誤分類

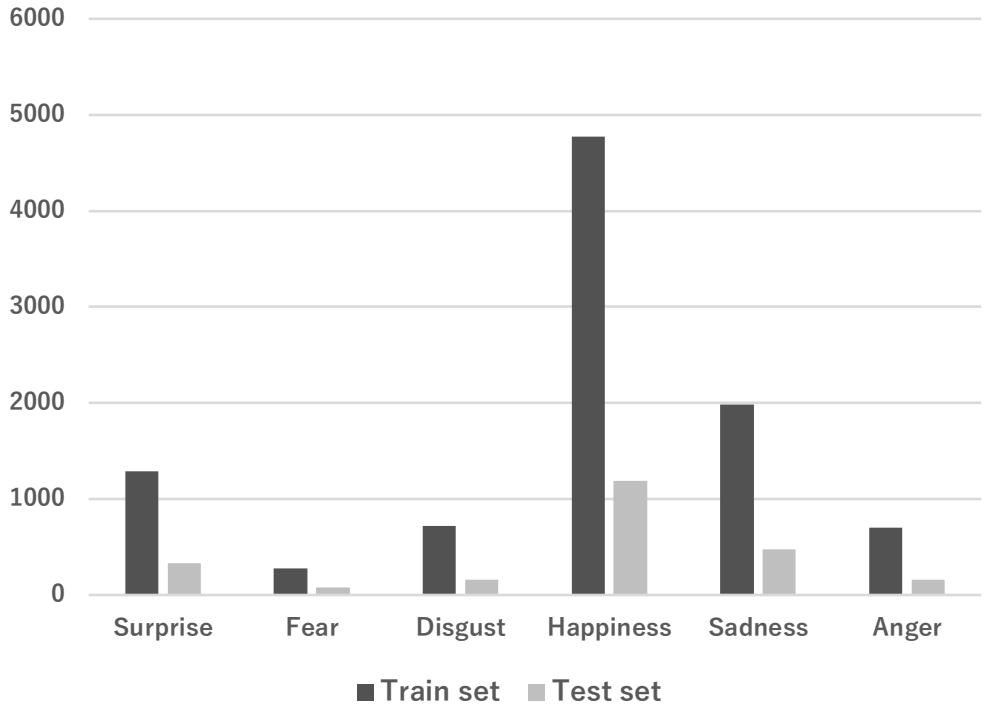


Figure 4.2　Class imbalance.

が学習に十分反映されるようにした。

4.3　使用モデル詳細

本研究では、Swin Transformer(Tiny) を使用する。Tiny モデルは Swin Transformer のモデルの中でパラメータ数が少なく軽量であり、計算資源が限られた環境や小規模データセットでの学習に適しているころから本研究で採用した。

さらに、事前学習済みモデルを利用し、最終分類層を 6 クラス分類用に置き換えることで、公開データセット (RAF-DB）への適用を可能にした。

なお、本研究でのモデル実装には PyTorch を用いた。

4.4　環境設定

4.4.1　学習環境

OS:Google colab 上の Linux

GPU:NVIDIA T4(16GB GDDR6)

実行端末：MacBook Air (M2,macOS 13.2)

使用言語：Python

4.4.2　学習条件

・Model:Swin-T(pretrained on ImageNet)

・class：6(Surprise, Fear, Disgust, Happiness, Sadness, Anger)

・Loss: CrossEntropyLoss with class weights

・Optimizer : AdamW

・学習率（初期値）：1.0×10^{-4}

・学習率スケジューラー：CosineAnnealingLR

・バッチサイズ：64

・エポック数：25

・Early Stopping : 5 エポック (検証損失が 5 エポック改善しなければ学習を打ち切ることで過学習を抑制)

・Mixed Precision Training : Autocast + GradScaler

4.5　評価指標

本研究では、Swin Transformer を用いた顔表情認識モデルの性能を以下の指標で評価した。

1. 精度 (Accuracy)

・全テストサンプルに対する正解率。

・モデル全体の性能を把握する基本指標として使用した。

2. クラスごとの適合率、再現率、F1 スコア (Precision, Recall, F1-score)

・各感情クラスに対する性能を詳細に確認するために算出。

・不均衡なクラス分布におけるモデルの偏りを評価するのに有用。

3. 混同行列 (Confusion Matrix)

・クラスごとの誤分類状況を可視化する。

・特定の感情が他の感情に誤分類されやすい傾向の把握に用いる。

4. 学習曲線 (Training/Validation Loss and Accuracy Curves)

・訓練中の損失および制度の推移を確認することで、過学習や学習の安定性を評価。

4.6 実験結果

4.6.1 学習過程における損失の推移

Figure 4.3 に、訓練データおよび検証データにおける損失の推移を示す。横軸はエポック数、縦軸はクロスエントロピー損失を示している。

Figure 4.3 より、学習開始時点において、Train Loss は約 1.7 を示していたが、エポックの進行に伴い急激に減少した。特に学習初期の数エポックでは Train Loss の低下が顕著であり、モデルが入力画像と感情ラベルの対応関係を速やかに学習している様子が確認できる。その後も Train Loss は一貫して減少し、エポック 10 以降では減少幅は緩やかになるものの、学習終了時点では約 0.75 まで低下した。

一方、Validation Loss についても学習初期には約 1.5 を示していたが、Train Loss と同様にエポックの進行とともに減少した。エポック 5 前後までは比較的大きな変動が見られるものの、その後は徐々に安定し、エポック 10 以降では概ね 1.1 前後で推移してる。

Train Loss と Validation Loss を比較すると、全エポックを通して Train Loss の方が低い値を示していることが分かる。また、Validation Loss は学習後半において急激な増加を示すことはなく、一定範囲内で推移している。

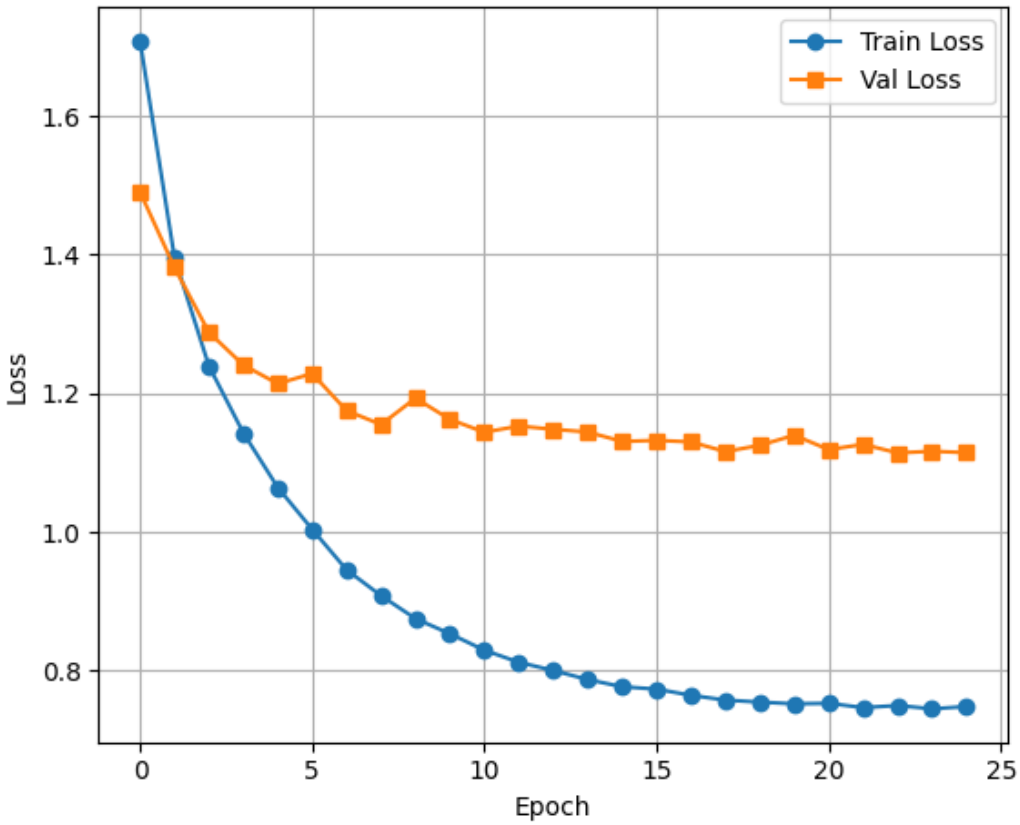


Figure 4.3　Training Validation Loss.

4.6.2　学習過程における精度の推移

Figure 4.4 に、モデルの訓練精度および検証精度の推移を示す。横軸はエポック数、縦軸は分類精度を示している。Figure 4.4 より、訓練制度は、学習開始時点では約 40%であったが、エポック 1 以降急激に上昇し、エポック 5 前後では 80%を超える精度に達している。その後も精度は徐々に向上し、エポック 10 以降では 90%を超える値を示した。最終的には約 99%に達しており、訓練データに対して高い分類性能が得られていることが分かる。

検証精度については、学習開始時点で約 63%であり、訓練精度と同様に学習初期において大きく上昇した。エポック 5 前後では 80%を超え、その後も緩やかに増加している。エポック 10 以降では 85%から 90%程度の範囲で推移し、最終エポック付近では約 90%前後の精度が得られた。

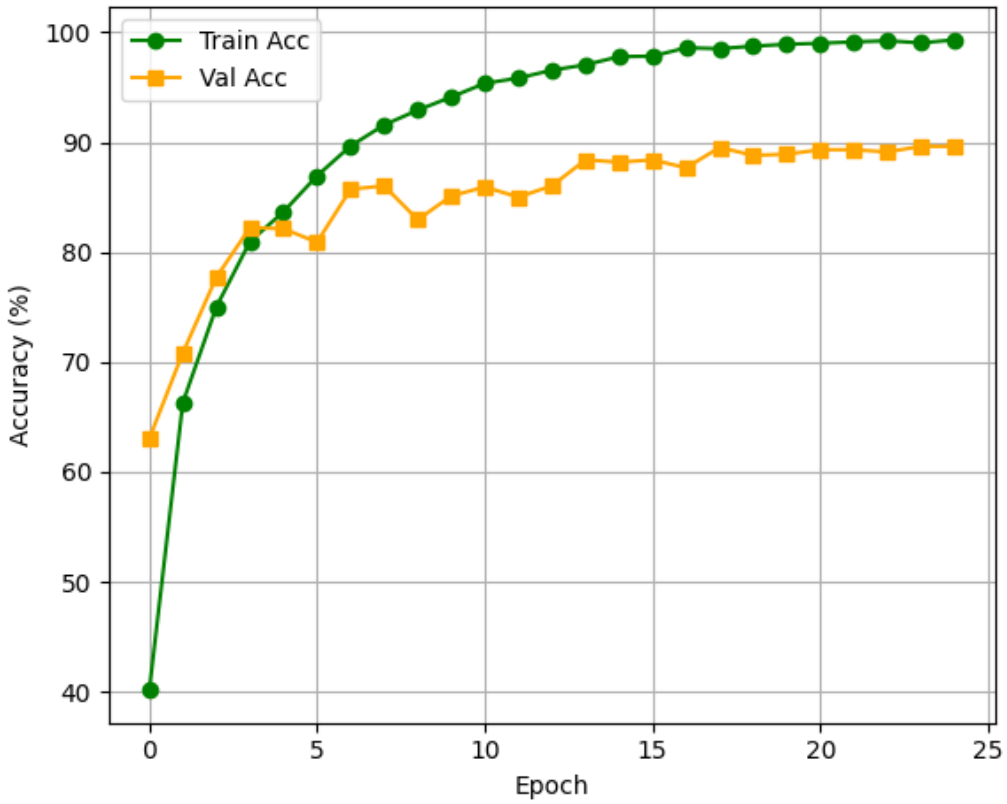


Figure 4.4　Training Validation Accuracy.

訓練精度と検証精度の推移を比較すると、訓練精度は学習の進行に伴いほぼ単調に上昇しているのに対し、検証制度は一定の範囲内で上下しながら推移していることが分かる。

4.6.3　学習率の推移

Figure 4.5 に、学習率の推移を示す。本研究では、学習率スケジューラとして CosineAnnealingLR を使用しており、初期学習率は 1.0×10^{-4} に設定している。Figure 4.5 より、学習率はエポックの進行に従ってなめらかに減少していることが分かる。また、学習初期では比較的大きな学習率が設定されており、エポックが進むにつれて徐々に小さな値へと変化している。エポック 10 前後では学習率は約 6.0×10^{-5} となり、学習後半では 1.0×10^{-5}

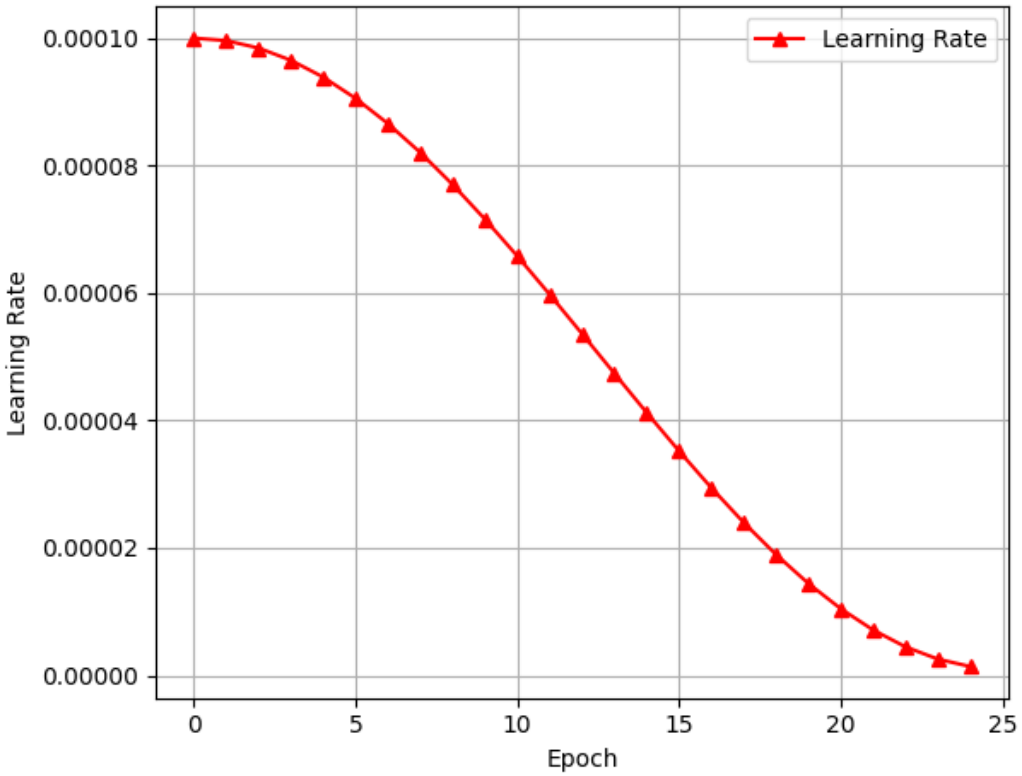


Figure 4.5　Learning Rate Schedule.

以下にまで低下していることが分かる。

最終エポック付近では学習率は非常に小さな値となっており、学習率スケジューラによる制御が適用されていることが確認できる。

4.6.4　混同行列および評価指標による性能評価

Figure 4.6 に、テストデータに対する混同行列を示す。縦軸は正解ラベル（Actual）、横軸はモデルによる予測ラベル（Predicted）を表しており、各要素は対応するクラスに分類されたサンプル数を示している。本実験では 6 クラス分類を行っており、テストデータ数は合計 2388 枚である。

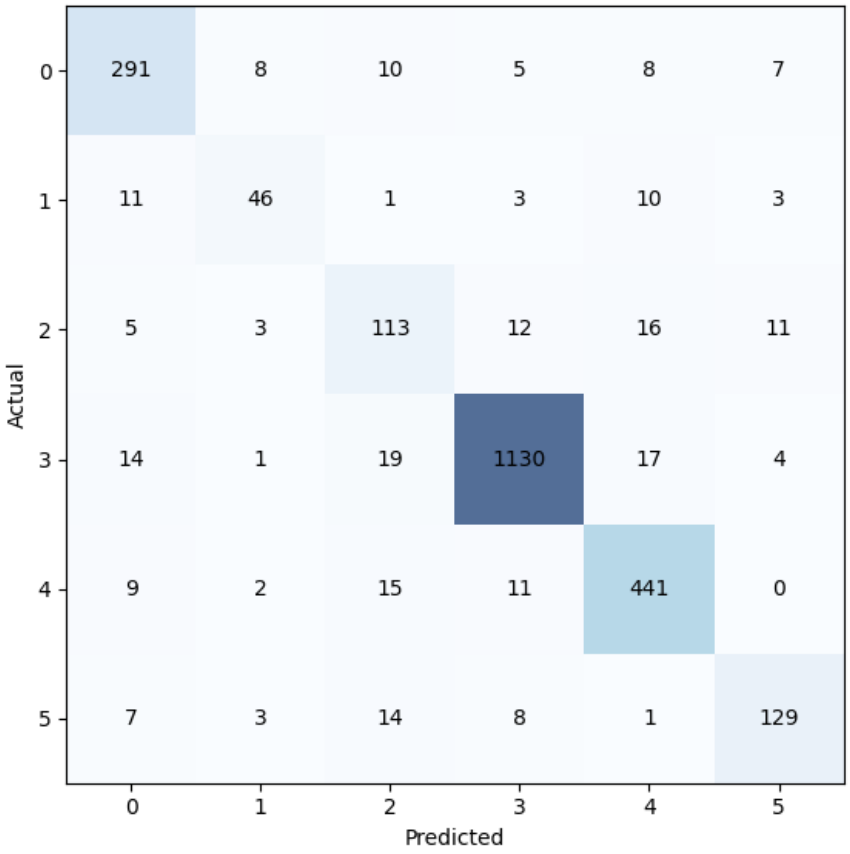


Figure 4.6　Confusion Matrix.

Figure 4.6 より、全体として対角成分に大きな値が集中しており、多くのサンプルが正しく分類されていることが確認できる。特にクラス 3 では正解数が非常に多く、高い分類性能が得られていることが分かる。一方で、すべてのクラスにおいて少数ながら他クラスへの誤分類も確認される。

次に、テストデータに対する分類性能を定量的に評価するため、Accuracy、Precision、Recall、F1-score を算出した。各クラスごとの評価指標を Table 4.2 に示す。

テスト全体における分類精度（Accuracy）は 0.9003 であり、約 90%のサンプルが正しく分類された。

各クラスの評価指標を見ると、クラス 3 およびクラス 4 では Precision、Recall、F1-score

Table 4.2　Classification Performance for Each Emotion Class.

Class	Precision	Recall	F1-score
0	0.86	0.88	0.87
1	0.73	0.62	0.67
2	0.66	0.71	0.68
3	0.97	0.95	0.96
4	0.89	0.92	0.91
5	0.84	0.80	0.82
Macro Avg	0.82	0.81	0.82
Weighted Avg	0.90	0.90	0.90

のいずれも高い値を示している。一方で、クラス 1 およびクラス 2 では他クラスと比較してやや低い値となっていることが確認できる。

Macro Average では Precision が 0.82、Recall が 0.81、F1-score が 0.82 であった。また、Weighted Average ではいずれの指標も 0.90 となっており、クラスごとのサンプル数を考慮した場合でも全体として高い分類性能が得られている。

第5章　考察

本研究では、Swin Transgormer(Tiny) を用いた顔認識モデルを構築し、RAF-DB データセットに適用した場合の学習挙動および認識性能について評価を行った。実験結果を踏まえ、得られた性能の要因や特徴について考察する。

5.1　学習過程に関する考察

学習過程において、tarain loss および Validation loss はエポックの進行に伴い減少し、学習後半においても大きな発散は確認されなかった。このことから、本モデルは訓練データに対して安定した学習を多ないながら、検証データに対しても一定の汎化性能を維持できていると考えられる。特に、学習初期における損失の急激な低下は、ImageNet による事前学習済み重みを利用した転移学習の効果によるものと考えられる。これにより、低レベルな特徴を一から学習する必要がなく、比較的少ないエポック数で効率的な学習が可能となったと考えられる。

一方で、train loss と Validation loss 間には一定の差があり、完全に一致する挙動は示していない。この差は、データ拡張や正規化を施していること、および訓練データと検証データの分布差による影響が考えられる。ただし、Validation loss が学習後半において急激に増加することはなく、Early Stopping の適用も含め、過学習は一定程度抑制できていると判断できる。

訓練精度は学習の進行に伴い一貫して向上し、最終的には非常に高い値を示した。一方で、検証精度は訓練精度ほど単調な上昇を示さず、一定の範囲内で変動しながら推移している。この傾向は、モデルが訓練データの特徴を強く学習している一方で、未知データに対しては一定のばらつきが生じていることを示していると考えられる。

しかし、検証精度が学習後半においても大きく低下することはなく、比較的高い水準で安定していることから、モデルは実用的な汎化性能を持つと考えられる。

5.2 クラスごとの性能差に関する考察

混同行列および評価指標の結果から、クラスごとに認識性能に差があることが確認された。特に、サンプル数の多いクラスでは高い精度および再現率が得られており、十分な学習データが性能向上に関係していると考えられる。一方で、サンプル数の少ないクラスでは誤分類が相対的に多く、精度および再現率が低下する傾向が見られた。この結果は、クラス不均衡が顔表情認識における性能に影響を与えることを示しており、class weight を適用したにもかかわらず、完全な性能差の解消には至らなかったことを意味する。ただし、Weighted Average の評価指標では全体として高い性能が得られていることから、実用上は一定の有効性を持つモデルであると考えられる。

5.3 Swin Transformer 採用の有効性

本研究で使用した Swin Transformer(Tiny) は、比較的軽量なモデルでありながら高い感情認識性能を示した。これは、パッチ分割による局所特徴抽出と、階層的な特徴表現を可能とする構造による効果であると考えられる。特に顔表情認識のように、目や口周辺など局所的な変化が重要となるタスクにおいて、Window ベースの Self-Attention は有効であると考えられる。また、計算資源が限られた環境においても学習および推論が可能であった点は大きな利点であると考える。

5.4 本研究の限界と今後の課題

本研究では、単一のラベルを持つデータセットおよび単一のモデル構成に基づいた評価を行っているため、他データセットへの汎化性能については十分に検証できていない。また、データ拡張手法やモデルサイズの違いによる影響についても体系的な比較は行っていない。

今後の課題としては、異なる表情データセットを用いた追加評価や、より大規模な Swin Transformer モデルとの比較、さらには誤分類サンプルの詳細な分析による性能改善が挙げられる。

第6章　結論

本研究では、Swin Transformer(Tiny) を用いた顔表情認識モデルを構築し、公開データセットである RAF-DB を用いて性能評価を行った。ImageNet で事前学習されたモデルを基盤とし、最終出力層を 6 クラス分類用に変更してファインチューニングを行うことで、比較的少量のデータセットにおいても高い認識性能が得られるか検証した。

実験の結果、テストデータに対して約 90%の分類制度を達成し、Swin Transformer が顔表情認識タスクにおいて有効であることを確認した。また、学習曲線および混同行列を用いた評価により、学習が安定して進行していることや、クラスごとの認識性能に差が存在することが明らかとなった。特に、サンプル数の多いクラスでは高い精度が得られる一方で、サンプル数の少ないクラスでは語分類が比較的多く発生する傾向が確認された。

以上より、Swin Transformer は局所的特徴と大域的特徴を効率的に捉えることが可能であり、顔表情認識において有効なモデルであると結論付けられる。一方で、クラス不均衡や表情間の類似性に起因する認識精度のばらつきといった課題も残されている。

今後の課題としては、データ拡張手法のさらなる検討や、クラス不均衡に対する手法の比較、より大規模なデータセットを用いた評価などが挙げられる。これらにより、顔表情認識モデルの汎化性能および実運用環境への適用可能性の向上が期待される。

参考文献

1) 柳井啓司・中鹿亘・稲葉通将. IT Text 深層学習. オーム社, 2022 年
2) 小高知宏. 機械学習と深層学習-C 言語によるシミュレーション-. オーム社, 2016 年
3) 片岡裕雄・山本晋太郎・徳永匡臣・箕浦大晃・QIU YUE・品川政太朗. Vision Transformer 入門. 技術評論社, 2022 年
4) 岡谷貴之. 機械学習プロフェッショナルシリーズ 深層学習 改訂第 2 版. 講談社, 2022 年
5) SANFOUNDRY. https://www.sanfoundry.com/machine-learning-model-evaluation-metrics-accuracy-precision-recall-f1/?utm_source=chatgpt.com#3(2026/2/12 閲覧)
6) AI Smily. 過学習とは?具体例と発生する原因・防ぐための対策方法をご紹介. https://aismiley.co.jp/ai_news/overtraining/(2026/2/12 閲覧)