卒 業 研 究 報 告 題 目

fine-tuningにおける学習層数による影響

Effect of the Number of Learning Layers in fine-tuning

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

2020E07 岩渕　大和

令 和 8 年（ 2 0 2 6 年 ）　　2 月 1 3 日 提 出

Abstract

In recent years, artificial intelligence (AI) has undergone rapid evolution and is being actively researched.

The use of AI enables high-speed, highly efficient task execution and offers the potential to reduce labor costs, particularly in real-time processing.

On the other hand, achieving both processing speed and classification accuracy remains a challenge in image classification. One approach to address this is fine-tuning after transfer learning.

Fine-tuning is the process of adjusting the weights of specific layers in a pre-trained model to suit a task. By adapting the model itself, this technique enables higher-accuracy results while retaining the benefits of transfer learning.

When performing machine learning using neural networks, it is typically necessary to build a model from scratch. However, this requires training on vast amounts of data and incurs significant costs. Fine-tuning, on the other hand, offers the practical advantage of enabling new tasks simply by adjusting an existing model.

In fact, fine-tuning is used to identify specific terminology or highly specialized data and is employed in chatbots and AI systems that generate text.

This study compares the training time and efficiency required for each layer by varying the number of layers used for fine-tuning after transfer learning for two-class dog-cat classification and multi-class flower classification.

This approach aims to identify the optimal number of layers for training in terms of both time and accuracy, thereby contributing to the efficiency of various systems.

As a result, the impact on the number of learning layers became stronger as the data became more complex due to handling multiple classes or as the amount of data increased.

目 次

第1章　序論

近年、人工知能（AI）は目まぐるしい進化を遂げ、活発に研究されている。特にリアルタイム処理では、これまで人手で行っていた作業を AI に代替することで、高速かつ高効率にタスクを実行でき、人件費の削減なども期待できる。その一方で、画像分類においては処理速度と分類精度の両立が課題となる。これに対する短縮手法の一つとして、転移学習後のファインチューニングが挙げられる。

ファインチューニングとは事前学習させたモデルの特定の層の重みをタスクに合わせて調整するプロセスであり、モデル自体を合わせることにより転移学習の良さを残しながらより高精度な結果を得ることができる手法である。

本研究では、2 クラス犬猫分類と多クラス花分類の転移学習後にファインチューニングを行う層数を変更し、層数ごとに要する学習時間と学習効率を比較する。この際、事前学習済みモデルには、156 層の深さを持つ畳み込みニューラルネットワークであり、事前学習済み画像が 100 万枚以上存在するモデルである MobileNet-v2 モデルを使用した。これにより、時間的・精度的に最適な学習が可能となる層数を明らかにし、様々なシステムの効率化に貢献できると考える。

第2章　ニューラルネットワーク

2.1　ニューロン

ニューロンとは私たちの脳神経ネットワークのことであり、神経系の中心的な細胞の呼称である。

ニューロンは大きく分けて細胞体, 樹状突起, 軸索の三つからなり、軸索から他のニューロンの細胞体や樹状突起と結合し、シナプスを作り、電気的情報がシナプスを経て次の細胞へと伝わる。ニューロンの概略図を Figure 2.1 として示す。このような私たち人間のニューロンをモデル化したものをニューロンモデルといい Figure 2.2 として示す。

2.2　ニューラルネットワークの概要 [1)]

ニューラルネットワークとは、人間の脳の動きを模倣し、データを AI に処理させる機械学習のモデルの 1 つであり、入力層、隠れ層（中間層）、出力層から構成されている。

入力層は全体のデータを入力する層であり、隠れ層（中間層）は入力層と出力層の中間にあり、出力層は全体の結果を出力する。この三つの層のうち入力層は中間層へデータを入力するのみなので、実際に使用される層は中間層と出力層である。

ニューラルネットワークは層数が多くなるほどに表現力が向上し、複雑なパターンを学習することができるが、多すぎると過学習のリスクや、学習時間の増大につながるため塩梅が重要である。ニューラルネットワークにおいて入力層から出力層へ順方向にパラメータを掛け合わせて予測値を計算することを順伝播といい、逆方向に計算することを逆伝播という。ニューラルネットワークの構成を Figure 2.3 として示す。

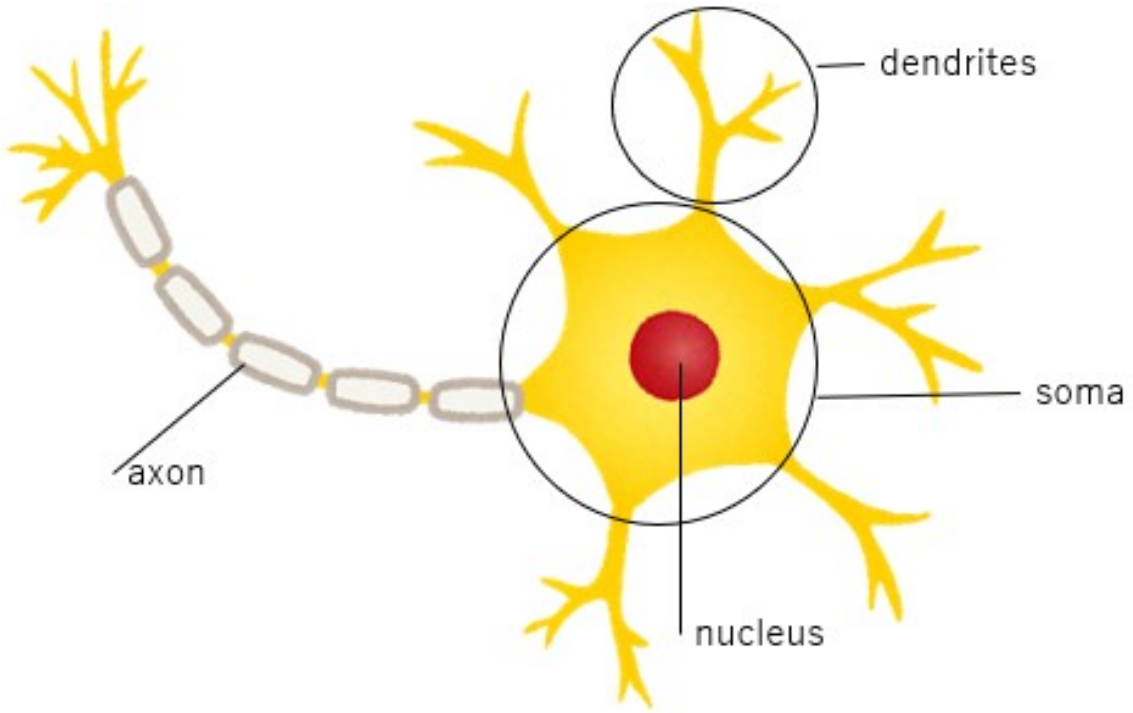


Figure 2.1: neuron diagram.

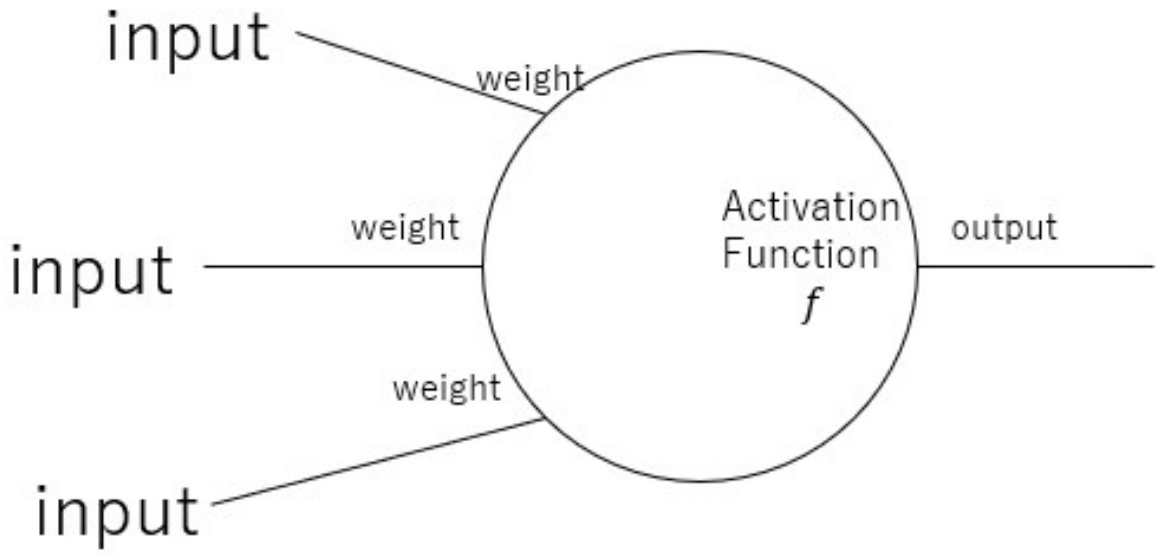


Figure 2.2: neuron model.

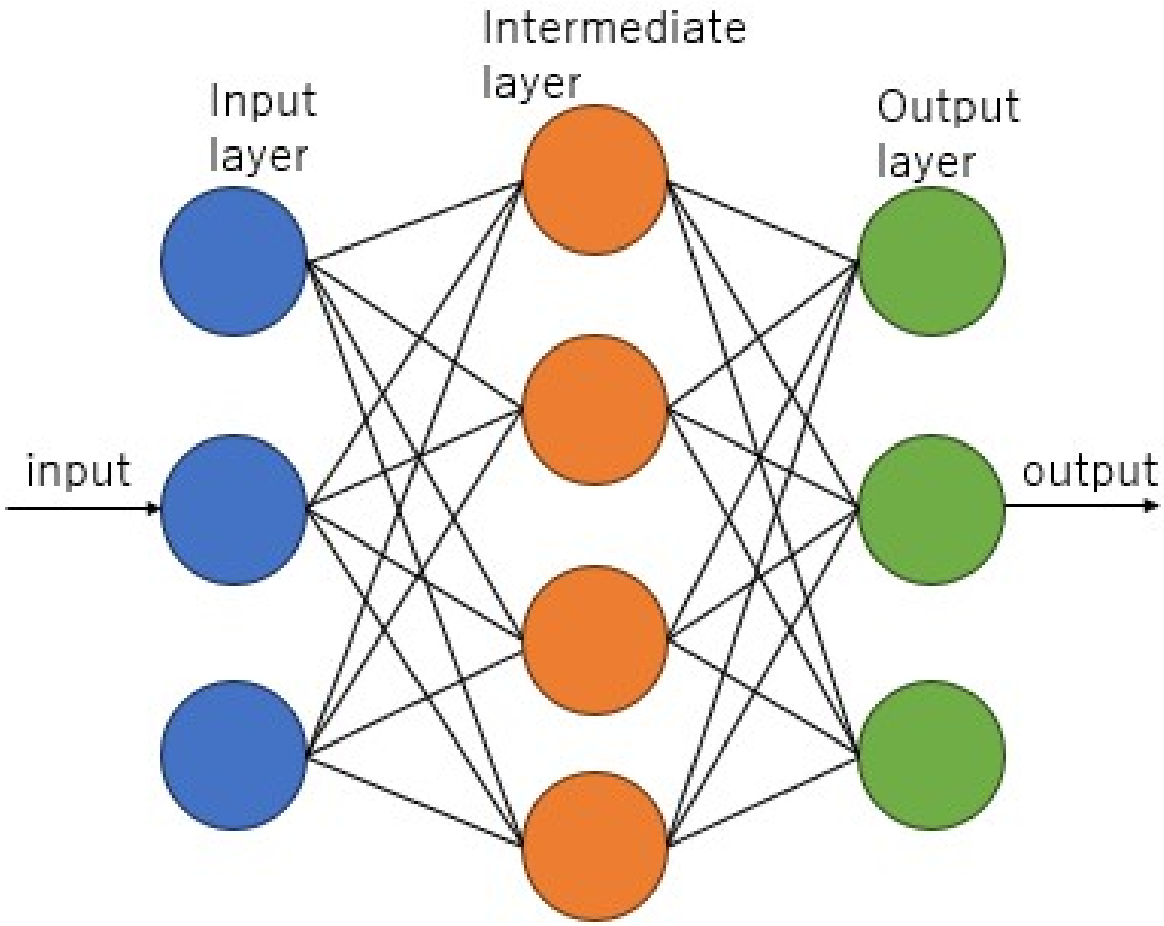


Figure 2.3: neural network.

2.3　活性化関数 [1)]

活性化関数とは、ニューラルネットワークの各ノードで適用される数学的関数のことであり、出力層に対して効果的に働くシグモイド関数や多クラス分類に対して出力された複数の値から最も確率の高いものを答えとするソフトマックス関数などがあり、以下の目的がある。

1. 非線形の導入：非線形の関数である活性化関数を用いることで線形変換だけで解決できない問題を解決している。
2. 特徴抽出の促進：解きたい問題に合わせてフォーマットを整えることで、データのパターンを学習しやすくする。

2.3.1　シグモイド関数

シグモイド関数とは 2 クラス分類問題に使われる関数であり、入力した値が大きければ大きいほど 1 に近づき、小さいほど 0 に近づく関数で式 (2.1) に表される。

$$z(u) = \frac{1}{1 + \exp(-u)} \tag{2.1}$$

この式を図にあらわしたものが Figure 2.4 であり、点対称の S 字の滑らかな曲線となっている。

2.3.2　ソフトマックス関数

ソフトマックス関数は、多クラス識別モデルの出力の合計確率が 1 になるように入力を変換する関数であり、式 (2.2) で表される。

$$\mathrm{f}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{2.2}$$

ソフトマックス関数の性質は以下のようなものがある。

1. 正規化：入力ベクトルの要素を正規化することで、合計値 1 の確率分布に変換し、どのクラスに入っているのかを確率として解釈する。
2. ソフトマックス関数は指数関数であり、入力値の差が大きく反映される。これによって入力値が大きい要素の出力確率が大きくなる。

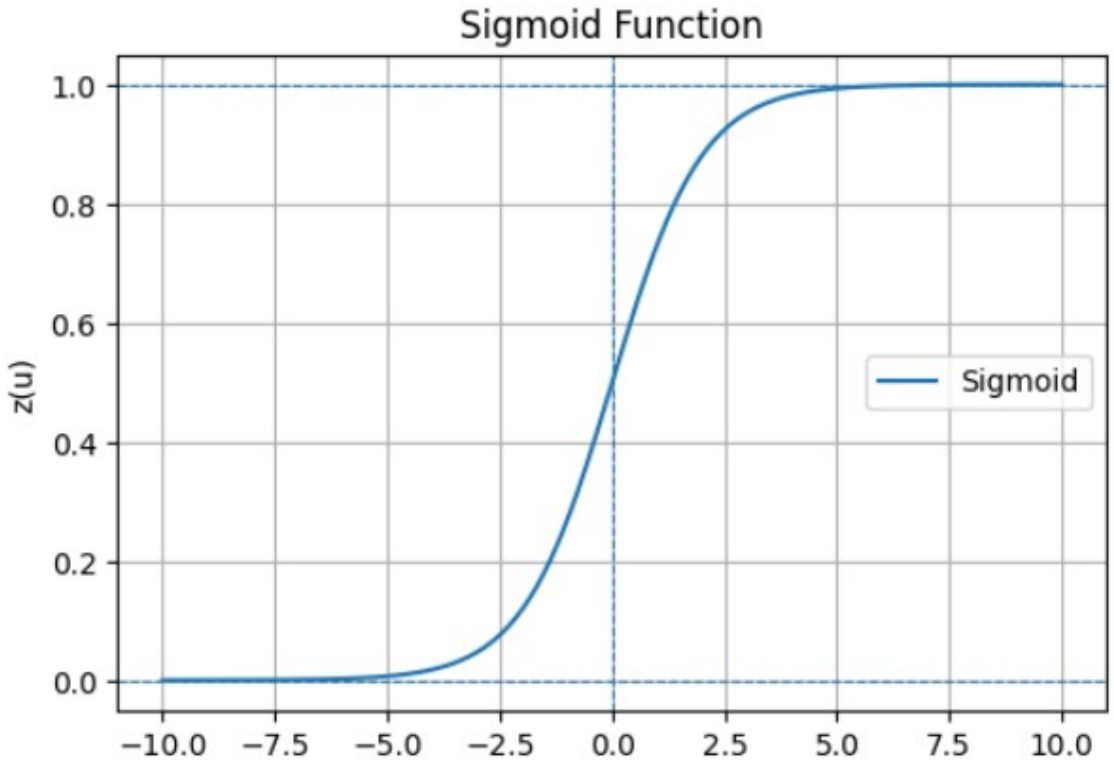


Figure 2.4: sigmoid function.

2.4 損失関数

機械学習モデルは入力されたデータから結果を予測するが、最初の段階では予想が外れていることが多いのでどの程度間違っているのかを定量的に表すものが損失関数である。損失関数の出力が小さいほどモデルの予測が優れていることの証拠となる。

交差エントロピー誤差は教師あり画像分類では必須の関数であり、機械学習でもよく用いられる。この誤差が小さいことが精度の向上であり重要である。交差エントロピーは真の確率分布 p と推定した確率分布 q の比較として式 2.3 で示され、p と q の確率分布が似ていると交差エントロピー誤差は小さく、似ていないと大きくなる。

$$H(p,q) = -\sum_{x} p(x) \log q(x) \tag{2.3}$$

2.5 分類問題

ニューラルネットワークでは分類問題という推論で出力された予測値により、事前に定義された複数のクラスの内のどれに最も該当するかを判別するものである。分類問題は出力層の性質上大きく分けて二つに分けることができ二値分類と多クラス分類である。

2.5.1　二値分類

このニューラルネットワークでは 0 か 1 かの二値で分類するため二値分類と呼ばれており、出力層のニューロンが出力する値を 0.0〜1.0 の範囲に収める必要がある、そのため活性化関数であるシグモイド関数を使っている。

2.5.2　多クラス分類

0 か 1 かで表すことができるものは限られておりそれ以上の多クラスを分類することを多クラス分類といい、0 をスタートとする番号の場合はクラスインデックスとよばれ、主にカテゴリー変数 (例えば 0=青,1 ＝赤など) に使われる。このような割り当てをカテゴリー変数エンコーディングといい、特にクラスインデックスにエンコーディングする場合は数値エンコーディングという。

2.6　過学習 [3)]

2.6.1　過学習とは

過学習とは過適合とも言い、機械学習を行う際に学習データと AI が適合しすぎることによって学習データのみに最適化されてしまって汎用性のある問題に対処できず正確な結果が出ないことである。過学習の原因として、学習データが少ないことによって少ないデータのみで参照するために起こるもの、学習データに偏りがあり都合の良いデータのみを学習してしまい正しく予測できないもの、モデルが複雑すぎてノイズまで学習してしまい過学習が起こるものなどがある。

2.6.2　過学習の解決法

次のような方法で過学習を抑制, 解決する方法がある。まず、学習データの数を増やす方法である。学習データが多いほどにバリエーションも増えるため、未知のデータに近づくことで過学習を抑制できる。次に正規化する方法がある。これは複雑なモデルを単純なモデルへ変化させていく手法であり、L1 正規化と L2 正規化がある。

1. L1 正規化：不要な特徴量を減らし、必要な変数のみをモデルに利用したいときに使える方法である。
2. L2 正規化：一つ当たりの変数の影響を抑えることでモデルを滑らかにし過適合を抑制する方法である。

2.7　エポックとバッチ [1)]

エポックとは、データの学習回数であり、1 エポックごとに学習データのすべてを使う。

バッチとは、入力時の正解のペアの集まり (サンプル) のことであり、1 回の学習で 1 つのバッチが使用される。

バッチサイズはバッチに含まれているサンプルの数バッチ内のすべてのサンプルを使用し重みづけとバイアスが更新され、バッチサイズは学習中は一定である。

バッチサイズの例としては訓練データのサンプル数が 1000 個, バッチサイズが 50 とすると 1 エポックにつき 20 回の更新のみで完了するためバッチサイズの設定によって学習時間やパフォーマンスに大きく影響が出る。この関係を Figure 2.5 として示した。

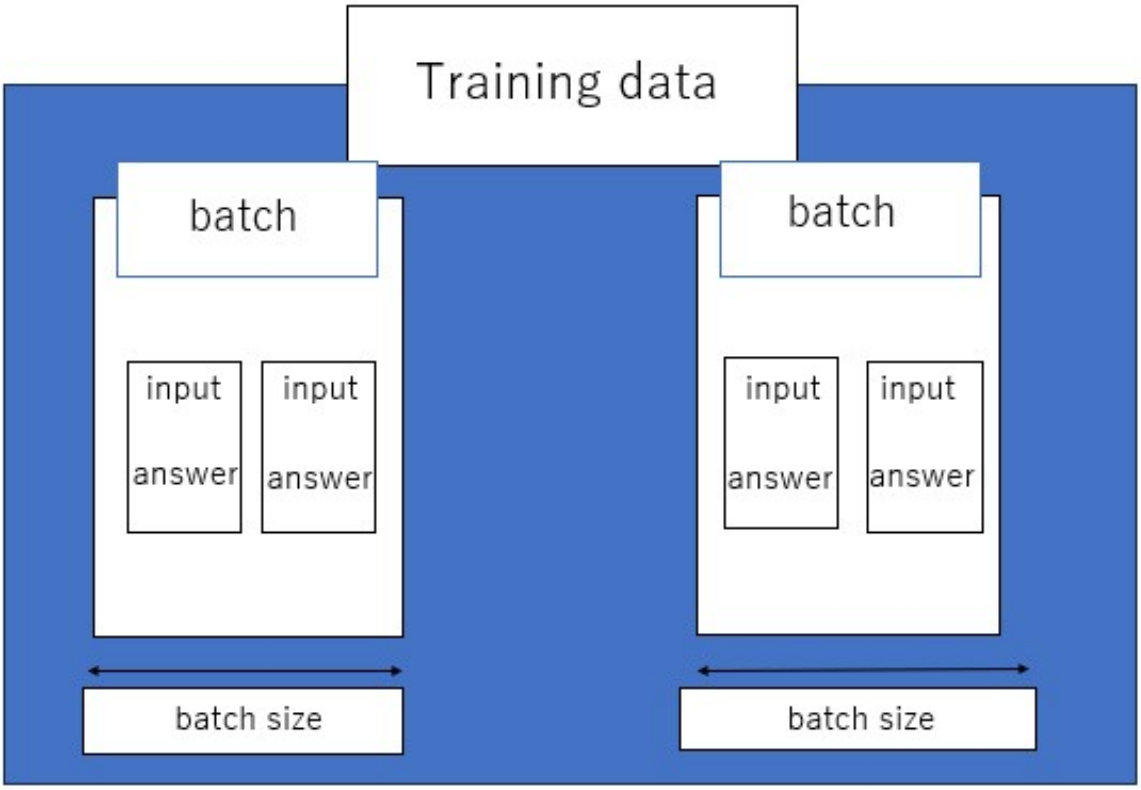


Figure 2.5: The relationship between training data and batches.

2.8　勾配降下法 [1)]

勾配降下法とは、誤差逆伝播法に用いられる手法でパラメータの修正量を決定する。勾配降下法のイメージ図を Figure 2.6 として示した。このグラフでは、横軸 w が重み、縦軸 E が誤差であり、$\frac{\partial E}{\partial w}$ は誤差 E を重み w で偏微分したもので曲線の傾き (勾配) を表している。重みの値で誤差は変化するが、実際に曲線そのものを知ることはできないた

め、曲線の傾きに応じて重みを変化させる。この曲線を降下するように重みを変化させていけば誤差を小さくできる。

そのためニューラルネットワークにおいてすべての重みとバイアスを更新するためには、その誤差の勾配を求めることになる。

勾配降下法の例として最も単純な確率的勾配降下法を挙げる。確率的勾配降下法は次の式 (2.4) と (2.5) で表される。

$$w \leftarrow w - \eta \frac{\partial E}{\partial w} \tag{2.4}$$

$$b \leftarrow b - \eta \frac{\partial E}{\partial b} \tag{2.5}$$

ここで e は重み、b はバイアス、E は誤差で矢印はパラメータの更新を表し、$\frac{\partial E}{\partial w}$ および $\frac{\partial E}{\partial b}$ は勾配である。また、η は学習係数といい、学習の速度を決める。

ニューラルネットワークにおけるすべての勾配を求めた後に上記の式に基づいてすべての重みとバイアスを更新する。

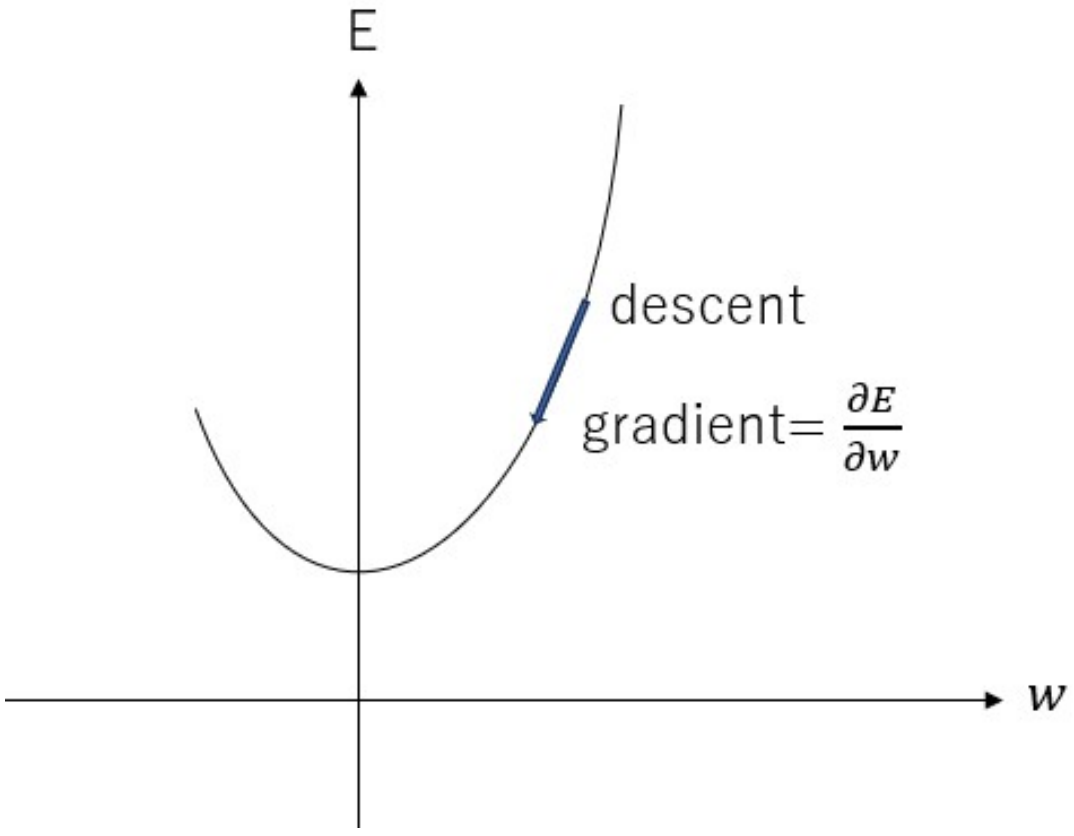


Figure 2.6: Schematic Diagram of Gradient Descent Method.

第3章　転移学習[4)]

3.1　転移学習とは

大規模データで事前学習されたモデルを別のモデルへ再利用する方法であり、少ないデータで高い精度を出すことができ、学習にかかる時間を大幅に削減すことができる。画像分類において、大規模データセットで学習することで、医療などの分野に効率よく適応でき、多岐に渡る分野での活躍が期待されている。

3.2　転移学習の仕組み

転移学習は以下のステップに分けられる。

1. 事前学習：大規模なデータセットを用いて、モデルに基本的な特徴抽出をさせ、汎用パターンや構造を理解させる段階であり、ここで転移させるための基礎を作る。
2. 転移：事前学習済みのモデルを、新しいタスクへ適用する段階であり、モデル全体を利用するか、低層部分をそのまま利用し、高層部分だけを変更して学習したいデータに合わせる。
3. 調整：新しいタスクに合わせ、モデルのパラメータを調整し、高精度のモデルを作成する。
4. 評価：実際にプログラムを動かし、精度, かかった時間, 汎用性などから評価し、適宜修正する。

3.3　ファインチューニング

転移学習の際、新たに学習するデータも大きいとかえって時間がかかることがある。転移学習のひとつであるファインチューニングは、事前学習させたモデルの特定の層の重みをタスクに合わせて調整するプロセスであり、モデル自体を合わせることにより転移学習の良さを残しながら、より高精度な結果を得ることができる。

ファインチューニングを行う際のプロセスも転移学習のようにステップがあり、

1. タスクの明確化：ファインチューニングを行えるのは画像分類問題だけではないため適切なデータセットを用意するのにどういったタスクをファインチューニングするのかを決める。
2. データセットの準備とベースモデルの選択：ファインチューニングを用いる際に

は高品質の少量のデータが理想的であり、パターンが豊富でデータに多様性があるものを選ぶ。ベースモデルは小規模タスクなら軽量モデルでも良いが、できるだけ大規模モデルを選ぶ。

3. 評価：実際にプログラムを動かし、正解率, かかった時間, 適合率などを評価し、適宜調整する。特に正解率や適合率にかかわるデータセットの質が重要である。

第4章　畳み込みニューラルネットワークの概要[2)]

畳み込みニューラルネットワーク (CNN) とは、特に画像データの処理に特化したディープラーニングモデルのことで、人間の視覚野の仕組みをヒントにしており、画像認識のみならず、画像の生成や物体の検知などに用いられている。畳み込みニューラルネットワークの概略図を Figure 4.1 として示す。畳み込みニューラルネットワークには、畳み込み層、プーリング層、全結合層があり、それぞれ以下の役割を担ている。

1. 畳み込み層：畳み込みニューラルネットワークの核となる層。画像から特徴を抽出する役割がある。フィルターという小さな行列が入力画像をスキャンするように各位置で画像の積和を計算することで画像のエッジや形状などの特徴を検出するように学習する。畳み込み層の仕組みを Figure4.2 として示す。
2. プーリング層：畳み込み層の後に配置される層で畳み込み層で特徴を抽出された画像では識別対象が画像の端にいる場合うまく行えない場合があり、それをサイズを縮小することによって計算量を削減し、モデルの位置不変性を高める。代表的なものに Max プーリングというものがあり、指定された領域内の最大値だけを取り出す手法であり Figure 4.3 として示している。
3. 全結合層：畳み込み層とプーリング層によって抽出された画像を 1 次元のベクトルに変換し、全結合型ニューラルネットワークの層へ渡す層であり、入力データがどのクラスに入るかを決定し、最終的な分類や回帰を行う。全結合層はすべての情報を受け取り、分類するという性質上畳み込み層とプーリング層での処理を何度か繰り返した後に処理が行われる。

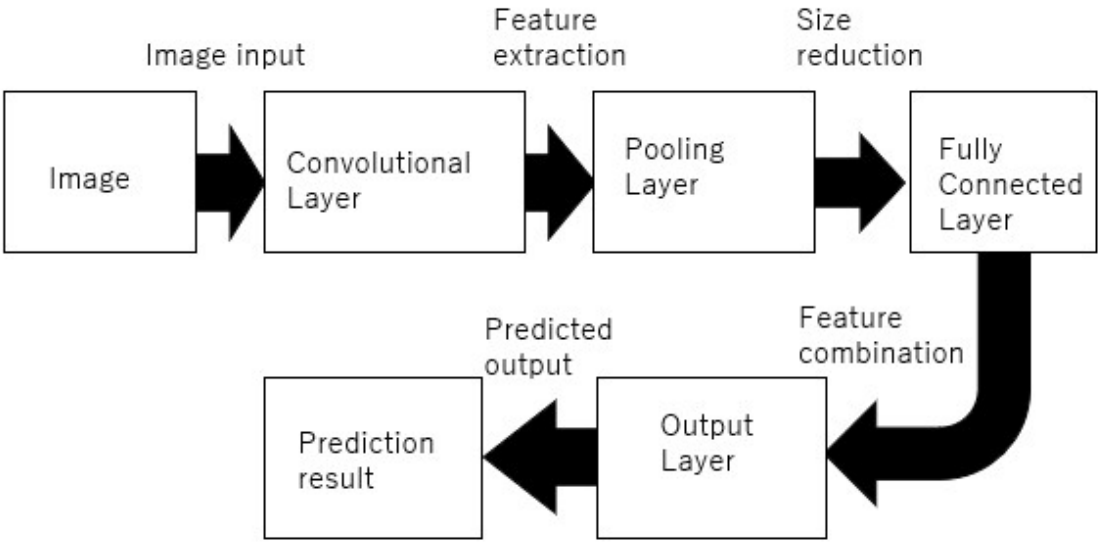


Figure 4.1: Figure 4.1 Overview of CNN.

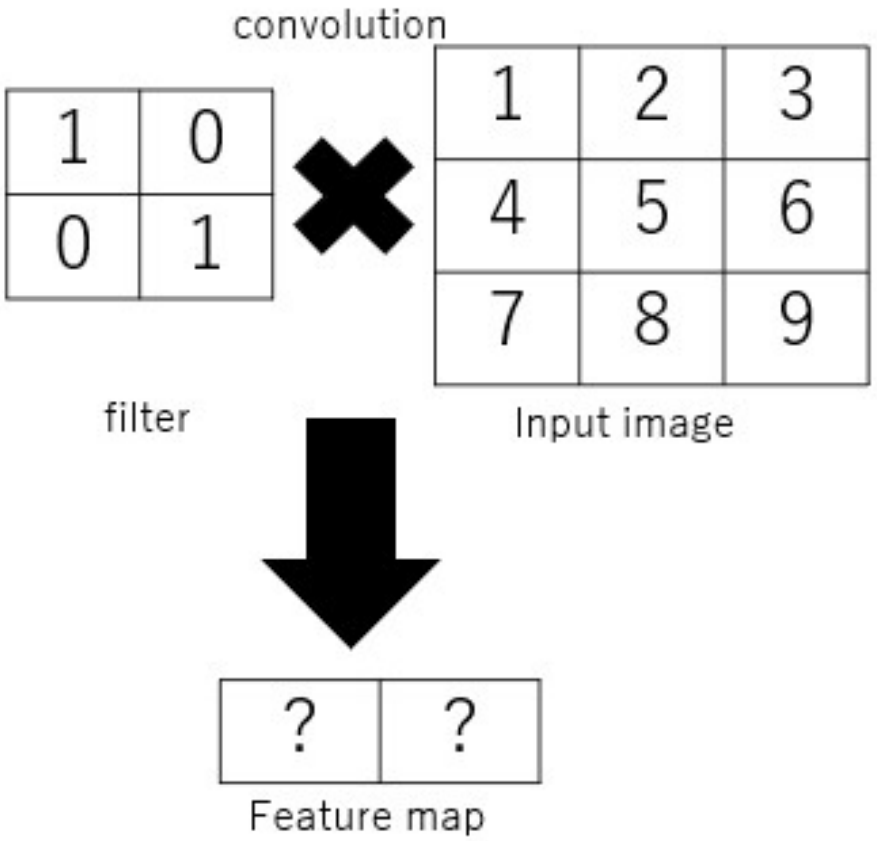


Figure 4.2: Figure 4.2 Process of feature extraction.

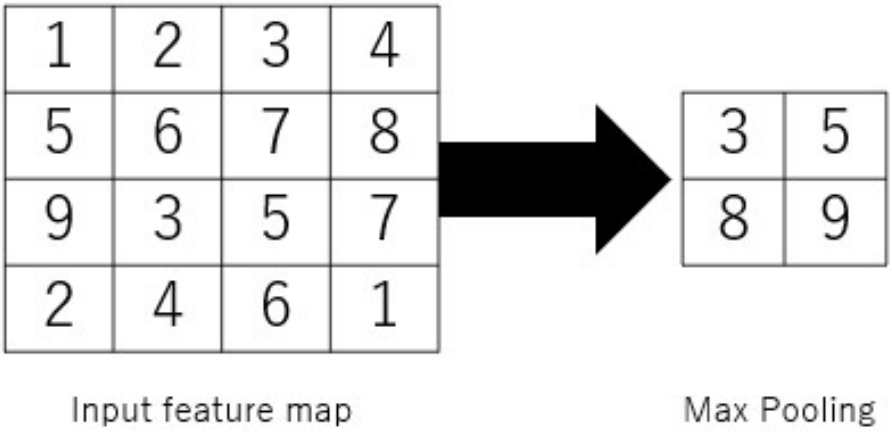


Figure 4.3: Figure 4.3 Max Pooling.

第5章　実験

5.1　実験方法

本実験では、転移学習とファインチューニングを用いて、事前学習済み畳み込みニューラルネットワークである MobileNet-v2 モデルを使い分類問題を行う。

この際、過学習や学習済みの特徴を壊さずにファインチューニングを行うため一度全ての学習層を凍結させており、そこから解凍する層を変化させ、学習層数が正解率とかかった時間にどのように影響したかを調べた。このとき、学習層数の上位を出力層とし、凍結している学習層数の上位から解凍していきそこから学習し、段階的に探っていった。

MobileNet-v2 モデルとは、156 層の深さを持つ畳み込みニューラルネットワークであり、100 万枚以上の事前学習済みモデルである。分類可能な画像は動物や植物など多岐に渡り、1000 のカテゴリーがある。

今回の実験では、MobileNet-v2 の上位の分類層を除いた 154 層に対し，新たに分類層を追加したモデルを用いた。なお、転移学習, ファインチューニングの回数ともに 3 回としているが、エポック数は多いと過学習のリスクがあり、計算時間も長くなってしまうため最低限の回数にしている。

今回の実験では、複数のデータセットを使用しているがデータセットは 2 クラス犬猫分類、多クラス花分類ともに Google が公開している TensorFlow Datasets から用いている。

5.2　実験環境

実験を行った環境は、以下の通りである.

- CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (2.80 GHz)
- GPU: Intel(R) Iris(R) Xe Graphics
- メモリ: 16GB
- OS: Windows 11 Pro ver. 24H2
- 使用ソフト: Google Colaboratory Python 3

5.3　犬猫分類の結果と考察

今回の実験では、2 クラスの犬猫データセットを用いた。

Table 5.1: Layer settings, validation accuracy, time, and efficiency.

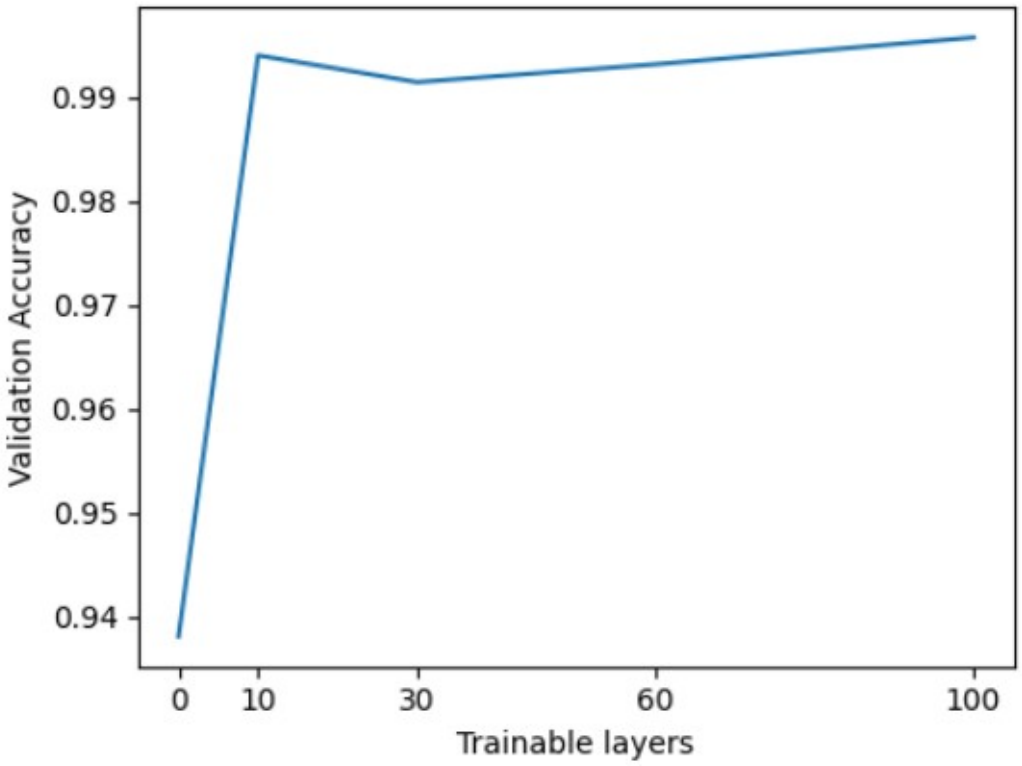
Layer settings	Validation Accuracy	Time (s)	Efficiency (Acc/Time)
0	0.938	235.8	0.003979
3	0.968	237.3	0.004079
4	0.970	225.9	0.004293
5	0.994	222.7	0.004463
6	0.991	249.0	0.003979
7	0.996	245.9	0.004051
10	0.991	234.2	0.004231
15	0.995	234.2	0.004248
20	0.996	253.9	0.003923
25	0.991	251.9	0.003935
30	0.992	335.4	0.002958
60	0.993	308.7	0.003216
100	0.996	396.5	0.002512

データ総量は 23262 枚であり、データ量が大きいためそのうちの 10%（2,326 枚）を学習用、20%から 30%(2,326 枚) の間にあるものを検証用として用いた。学習層数上位 (0,10,30,60,100) 層を学習し表したものを Figure 5.1、Figure 5.1 の中から結果の良かった学習層数上位 (5,10,15,20,25) 層を学習し表したものを Figure 5.2、Figure 5.2 からさらに結果の良かった学習層数上位 (3,4,5,6,7) 層を学習し表したものを Figure 5.3、最後に全てのデータを合わせグラフに表したものを Figure5.4、表に表したものを Table 5.1 とした。

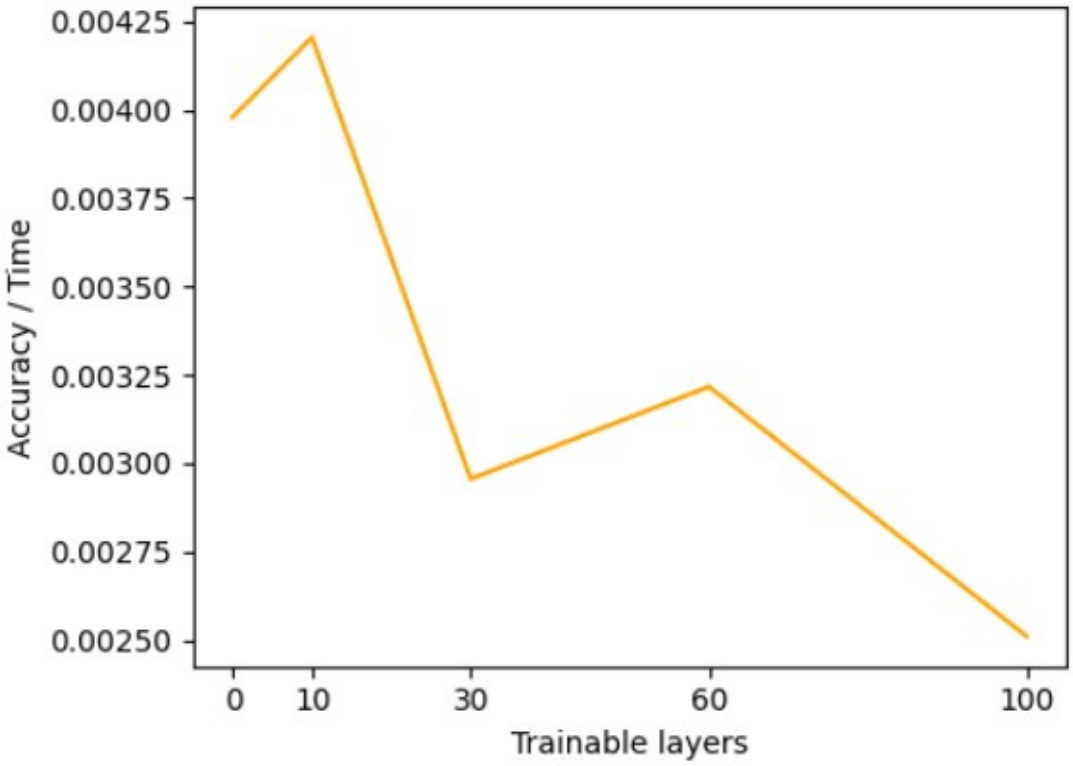
ここで、横軸は解凍した学習層数であり、縦軸は (a) のグラフが正解率,(b) のグラフが正解率を時間で割ることにより算出した効率となっている。Figure 5.4 と Table 5.1 から学習層数が 5 層を超えると正解率が横並びになり、層数が増えるほど計算時間が長くなっている。

今回使用した犬猫分類において最も正解率と計算時間の効率の良い学習層数は 5 付近と比較的浅い層であることが分かったが、原因としては以下のものを考えた。

1. タスクが元の事前学習タスクに近い：事前学習済みモデルがすでに汎用的な特徴を学習済みであり、クラス分類の際その特徴で識別可能になっている。

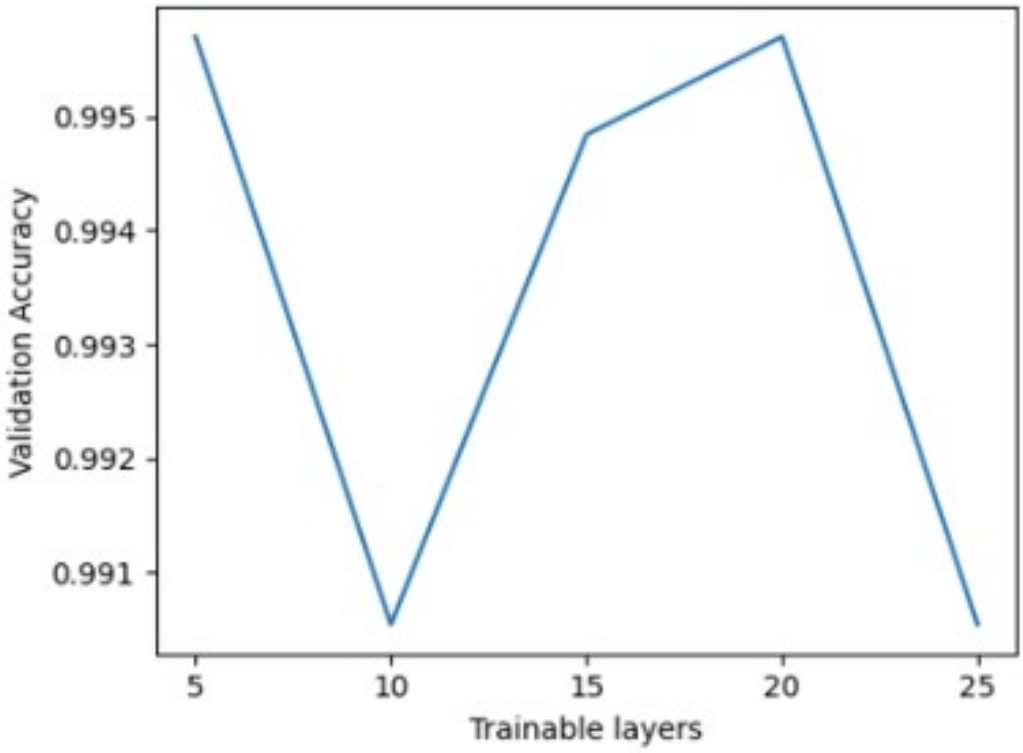


(a) Layers and Accuracy.

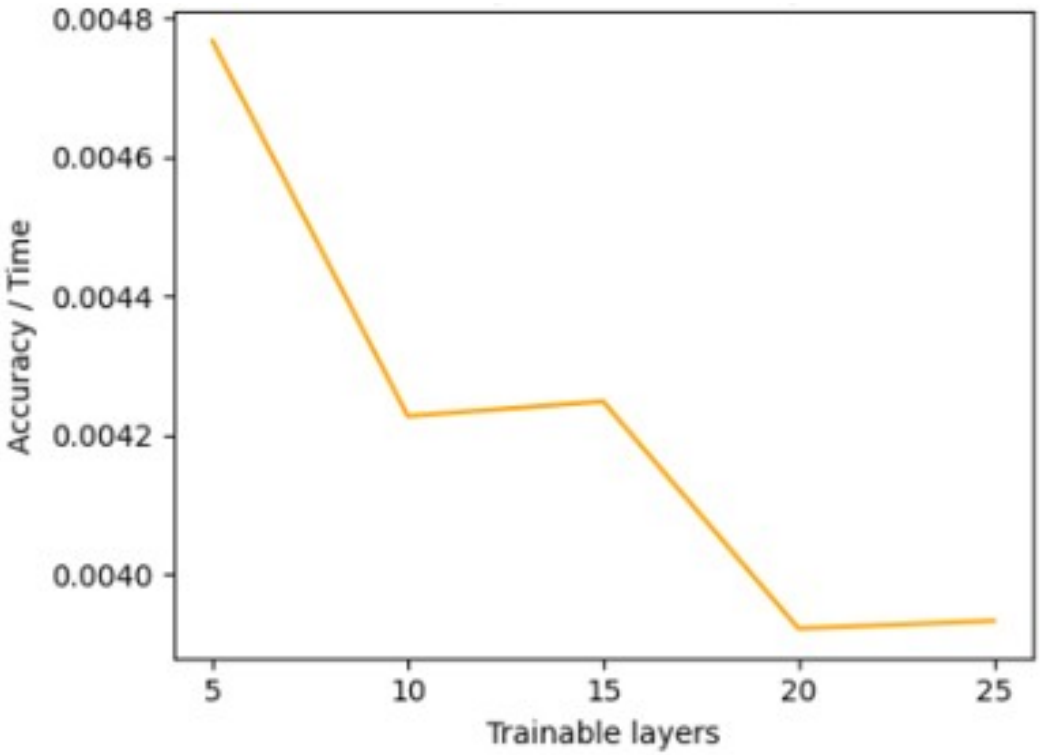


(b) Layers and Efficiency.

Figure 5.1: (0,10,30,60,100) Layer Learning Results.

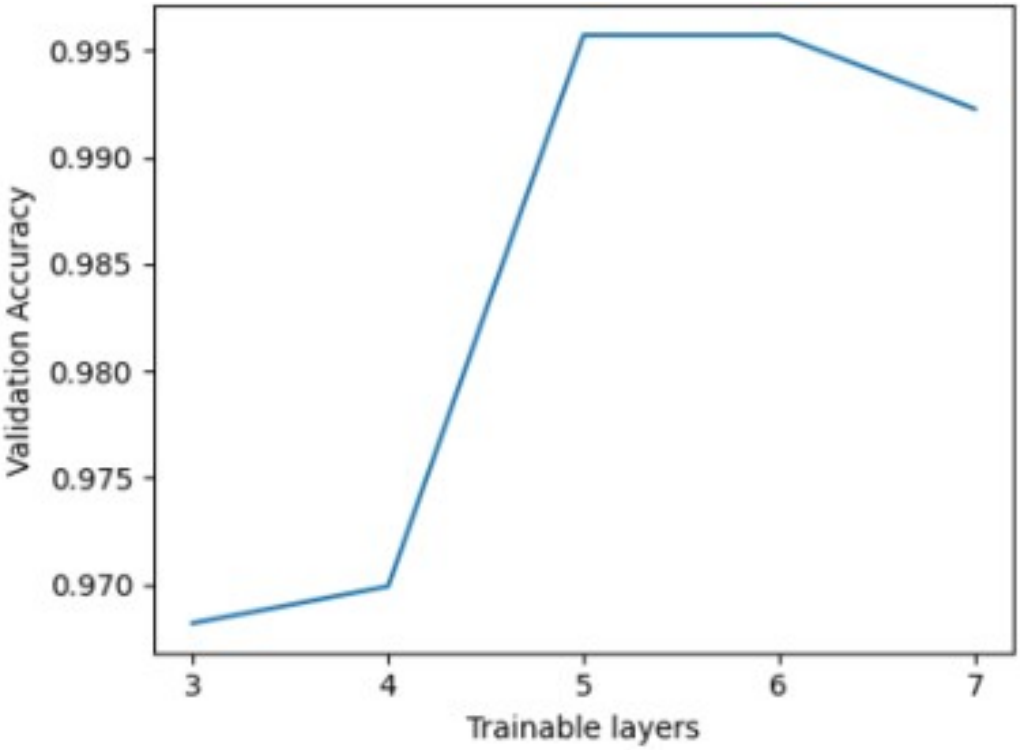


(a) Layers and Accuracy.

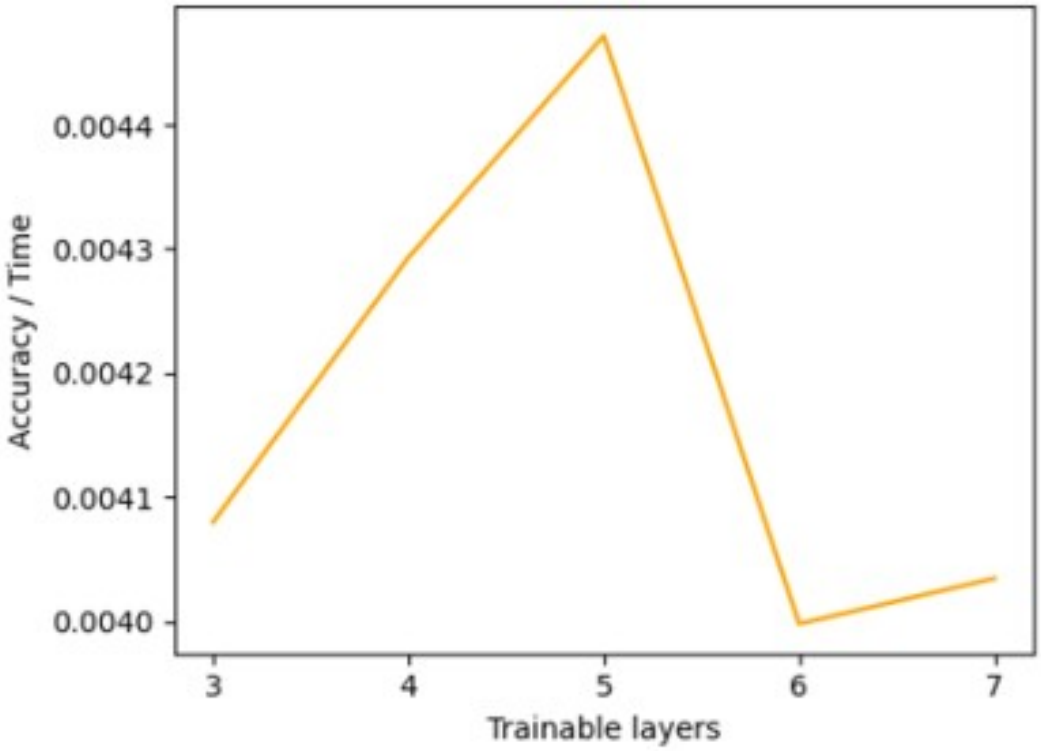


(b) Layers and Efficiency.

Figure 5.2: (5,10,15,20,25) Layer Learning Results.



(a) Layers and Accuracy.



(b) Layers and Efficiency.

Figure 5.3: (3,4,5,6,7) Layer Learning Results.

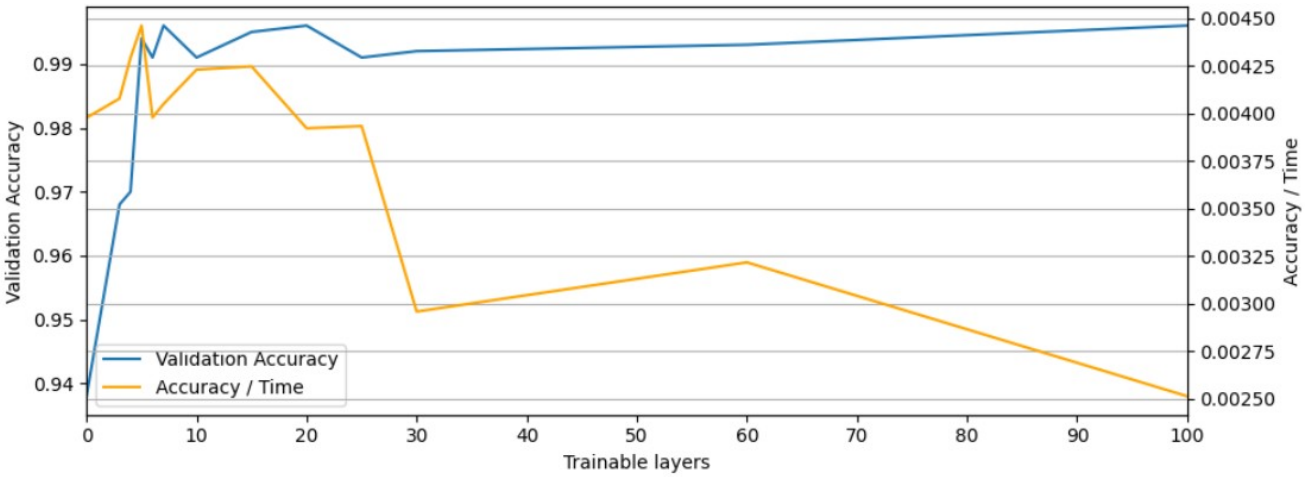


Figure 5.4: Combined Layer Learning Results.

2. 計算時間の関係：単純に学習層数が少なければ、計算時間は短くなり正解率が近くなればなるほど効率の面で有利になる。

このうち今回の実験では、分類が比較的簡単な犬猫分類としているため、タスクが元の事前学習タスクに近くなり今回の結果につながったと考えた。

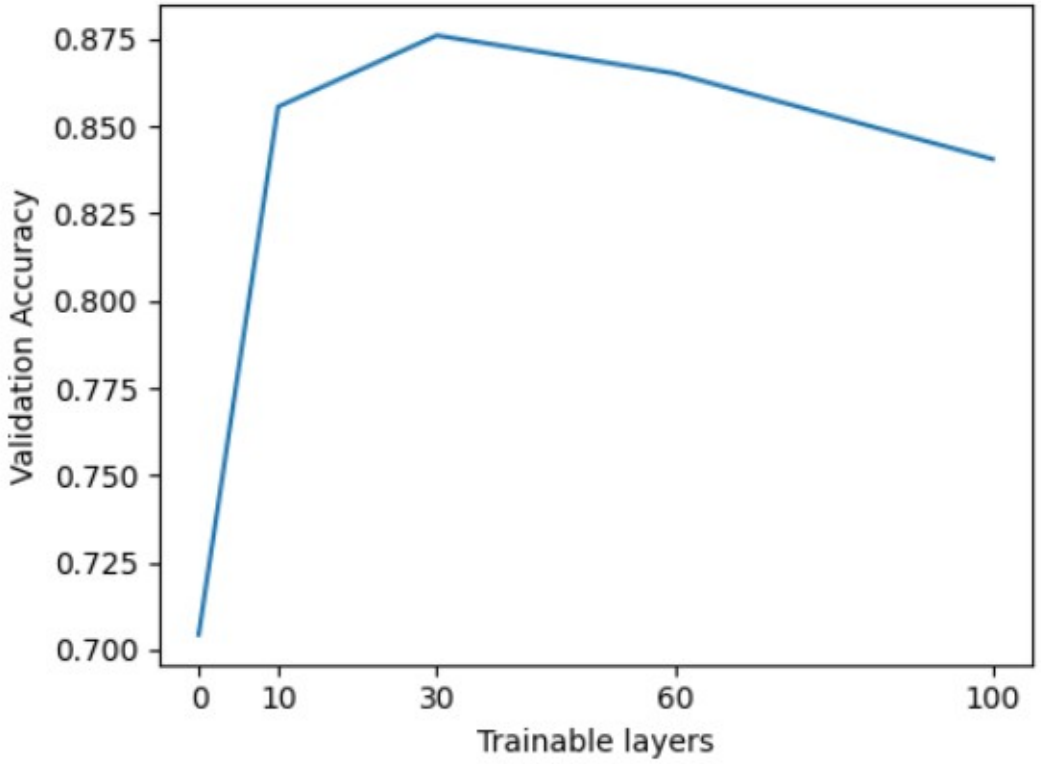
5.4 多クラス分類の結果と考察

犬猫分類問題は 2 クラス分類であり、誤パターンが少なく、学習に必要なデータ要求量が少ないなどの理由により画像分類が比較的簡単なものである。そのため別のデータセットを用意し、多クラス分類をしていく、今回は 5 クラスの花データセットを用意した、データ総量は 3670 枚であり、そのうち 80%（2,936 枚）を学習用、20%（734 枚）を検証用として用いた。

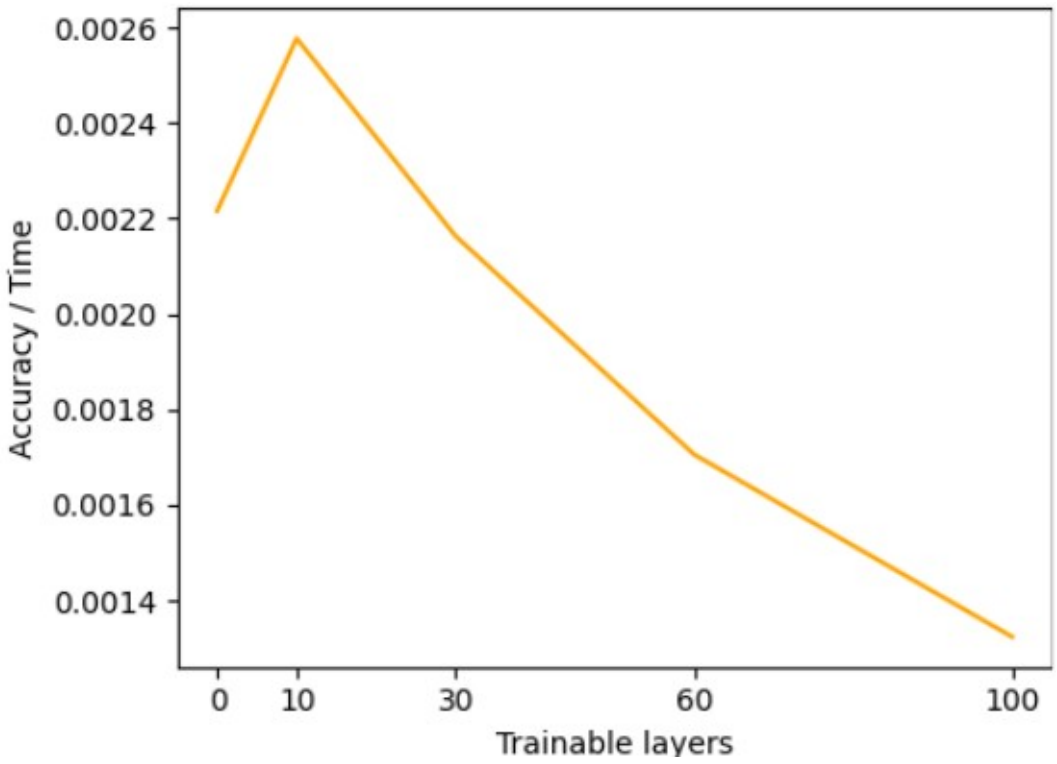
学習層数上位 (0,10,30,60,100) 層を学習し表したものを Figure 5.5、Figure 5.5 の中から結果の良かった学習層数上位 (5,10,15,20,25) 層を学習し表したものを Figure 5.6、Figure 5.6 からさらに結果の良かった学習層数上位 (18, 19, 20,21,22) 層を学習し表したものを Figure 5.7、最後に全てのデータを合わせグラフに表したものを Figure5.8、表に表したものを Table 5.2 とした。

Figure 5.8 と Table 5.2 から学習層数は 5 層から正解率が横並びになり、効率の良い学習層数は 22 層までになったが、通常学習層数が深くなるほどに計算時間が長くなるのにもかかわらず、22 層周辺の層の計算時間を見てみるとその前の層よりも計算時間が短くなってしまい不安定な結果になった。

しかしながらグラフと表から得られたデータを参照すると犬猫分類の結果と比べより

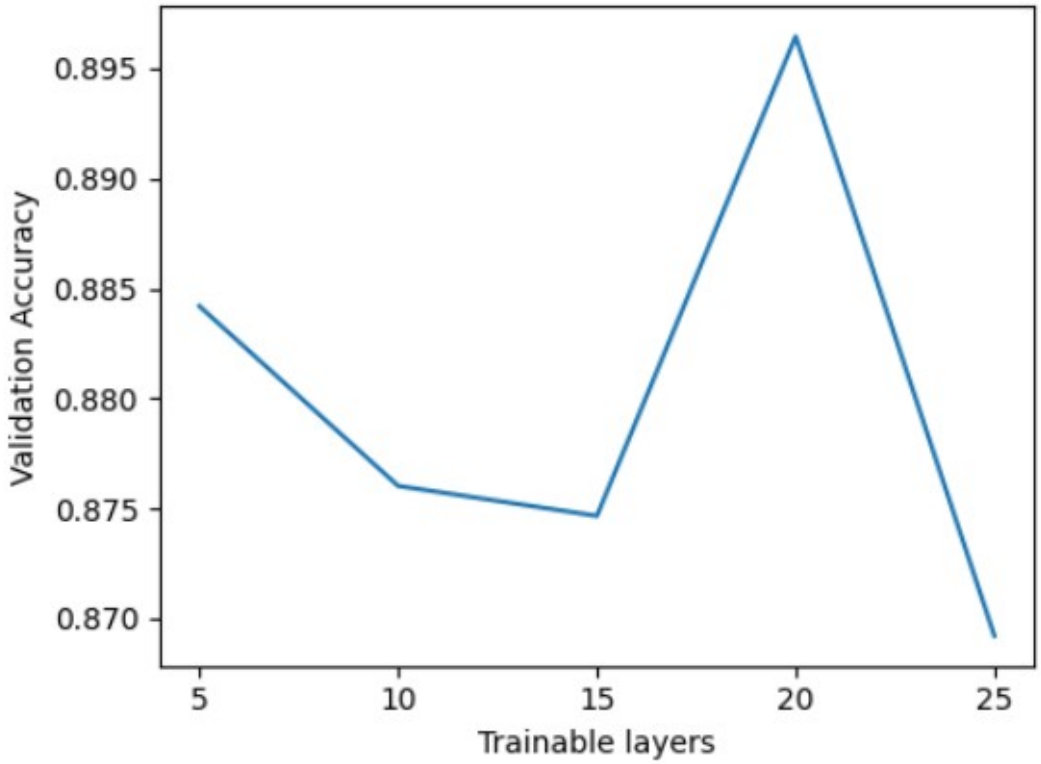


(a) Layers and Accuracy.

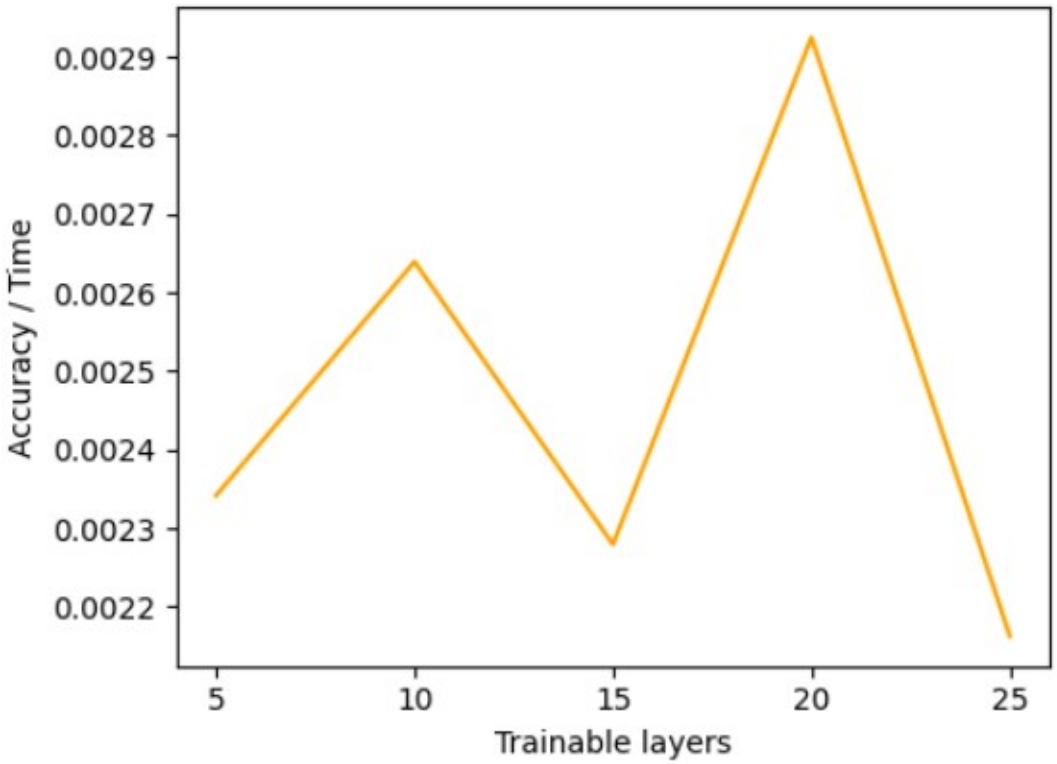


(b) Layers and Efficiency.

Figure 5.5: (0,10,30,60,100) Layer Learning Results.

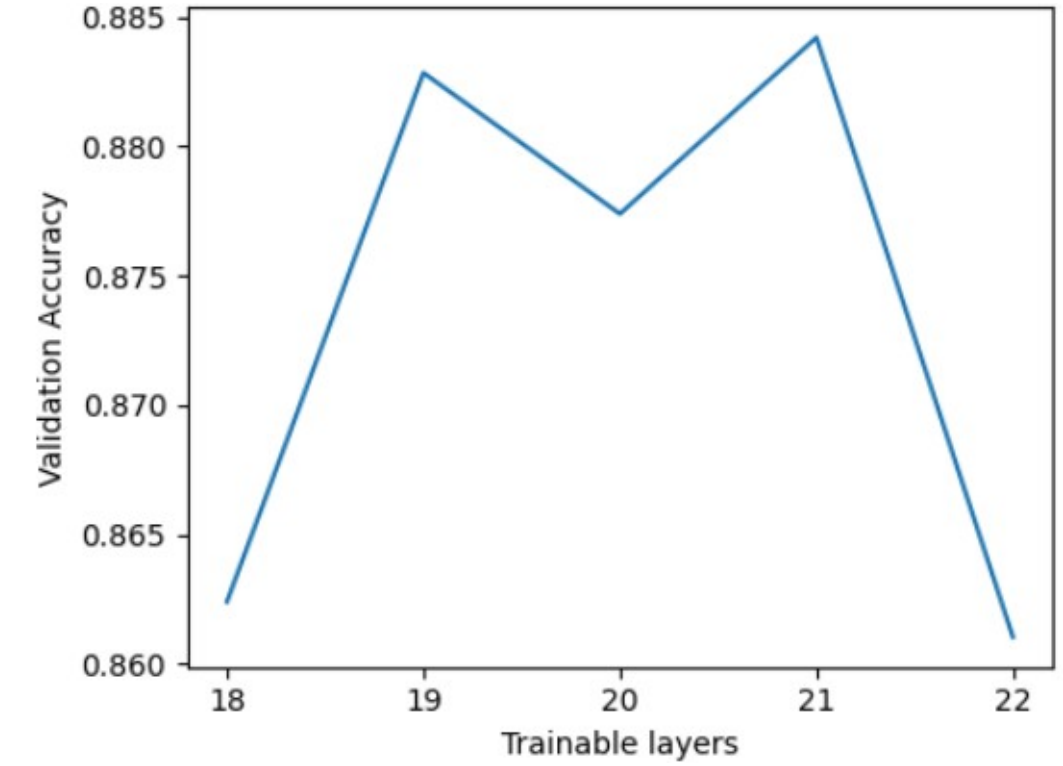


(a) Layers and Accuracy.

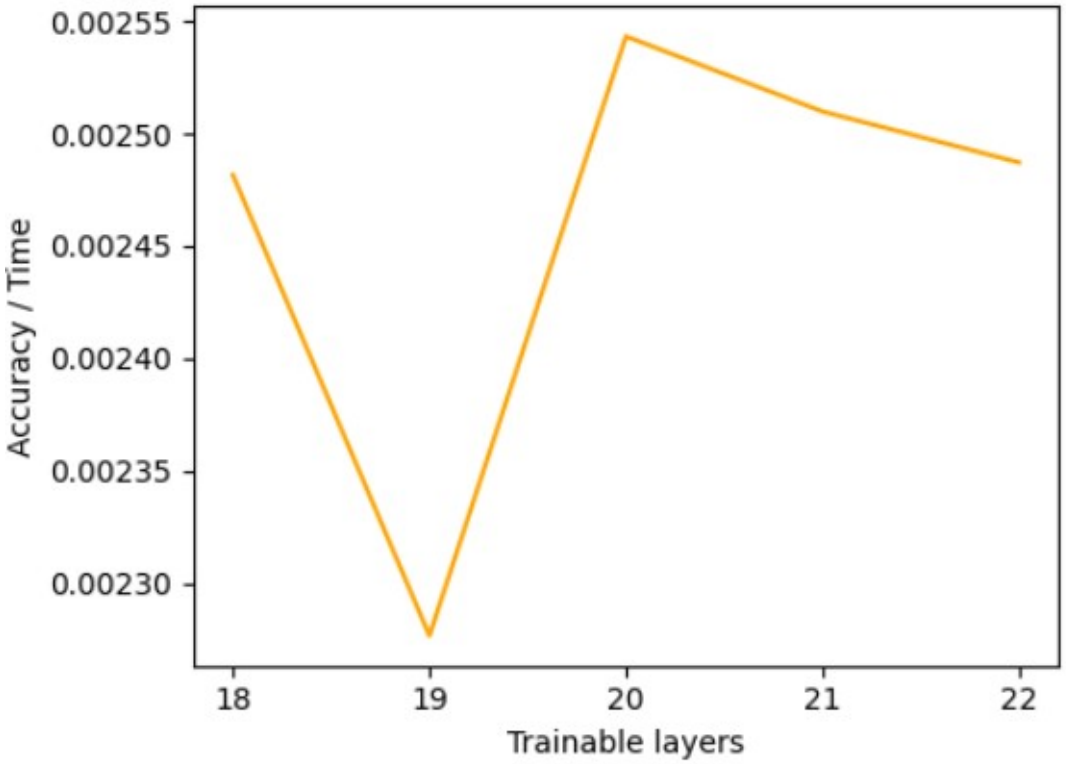


(b) Layers and Efficiency.

Figure 5.6: (5,10,15,20,25) Layer Learning Results.



(a) Layers and Accuracy.



(b) Layers and Efficiency.

Figure 5.7: (18,19,20,21,22) Layer Learning Results.

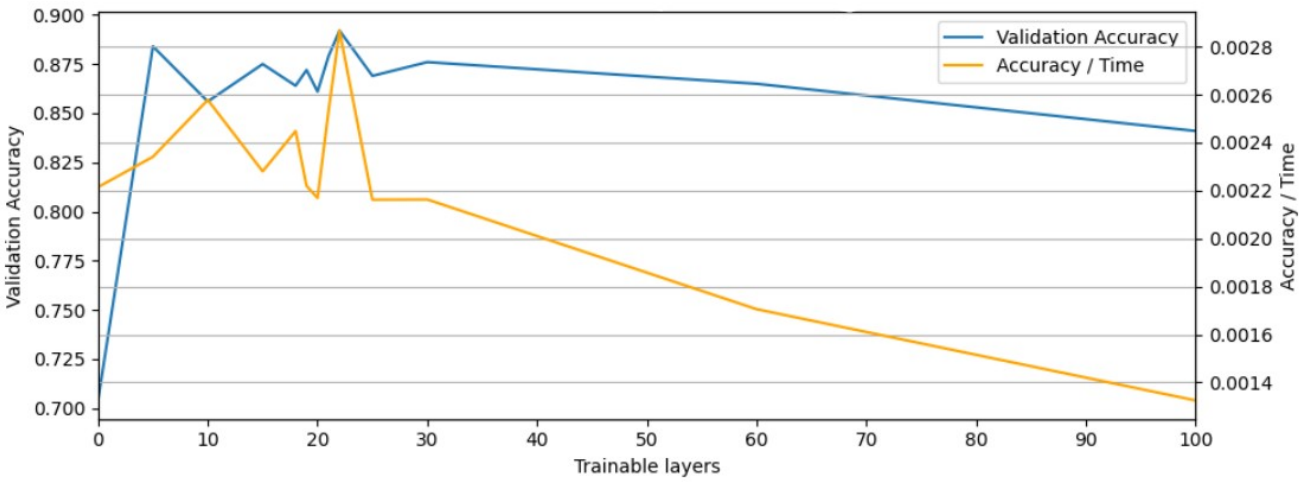


Figure 5.8: Combined Layer Learning Results.

深い層まで学習する方が効率が良いことが分かった。

これらのことを実験 5.2 の結果と比べ計算時間が安定しなかった原因として以下のものを考えた。

1. 開発環境：計算時間は開発環境により大きく左右され、特に GPU の並列計算能力は CNN の計算速度に大きく影響している。
2. TensorFlow の最適化の影響：最初のエポックはプログラム構築にかかるメモリの確保やコンパイルの最適化などをするため特に時間がかかる。
3. 実験の規模による影響の幅：行う実験が小規模であれば、比較する計算時間が小さくなり、不安定になりやすい。

このことから今回の実験では、小規模実験でありエポック回数が 3 回と少ないので最初のエポックにかかる時間の差により計算時間が大きな影響を受けたと考えた。

5.5 追加検証

5.5.1 データの規模に伴う変化

実験 5.2 で扱ったデータ量は小規模にして構成していたが、今回の実験では、モデルに使用するデータ量を単純に 2 倍にして扱ってみる。

2 クラスの犬猫データセットから、使用するデータを 10%(2,326 枚) から 20%（4,652 枚）を学習用、20%から 30%(2,326 枚) であったものを 20%から 40%(4,652 枚) へと変更し実験 5.2 で検証した学習層数を用いて追加で検証した。

学習層数と正解率を学習層数と効率を Figure 5.9 として表し、それぞれのグラフを合

Table 5.2: Layer settings, validation accuracy, time, and efficiency.

Layer settings	Validation Accuracy	Time (s)	Efficiency (Acc/Time)
0	0.704	317.9	0.002215
5	0.884	377.6	0.002341
10	0.856	332.0	0.002578
15	0.875	383.7	0.002281
18	0.864	352.9	0.002448
19	0.872	392.9	0.002219
20	0.861	396.9	0.002169
21	0.879	347.3	0.002531
22	0.892	311.1	0.002867
25	0.869	401.9	0.002163
30	0.876	405.0	0.002163
60	0.865	507.2	0.001705
100	0.841	634.8	0.001325

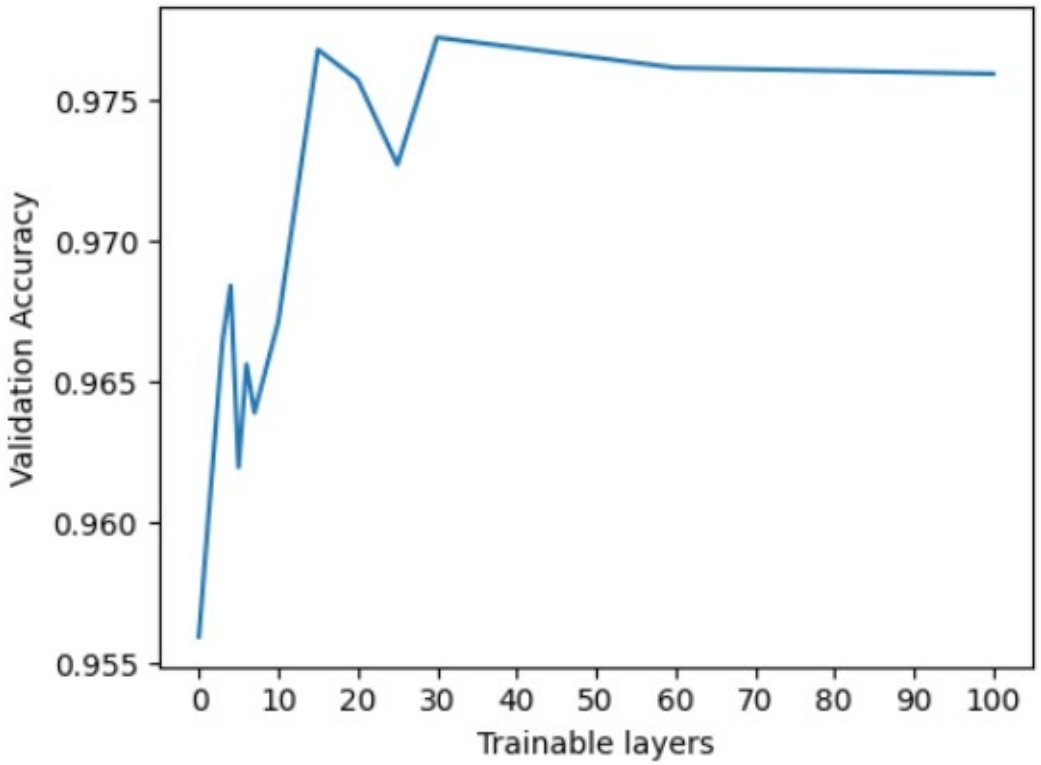
わせたものを Figure 5.10 として表し、実験 5.2 のグラフから効率の値を今回のグラフと合わせ Figure 5.11 として表した。その後各グラフから得られたデータを表にし Table 5.3 として表した。

グラフ Figure 5.11 から、今回の実験で得られたグラフは学習層数が増えるほど効率が落ちていくため、実験 5.2 の場合よりも効率の良い学習層数の決定が容易となった。しかし、学習層数が 1 桁の少ない範囲では、データが曖昧になってしまった。

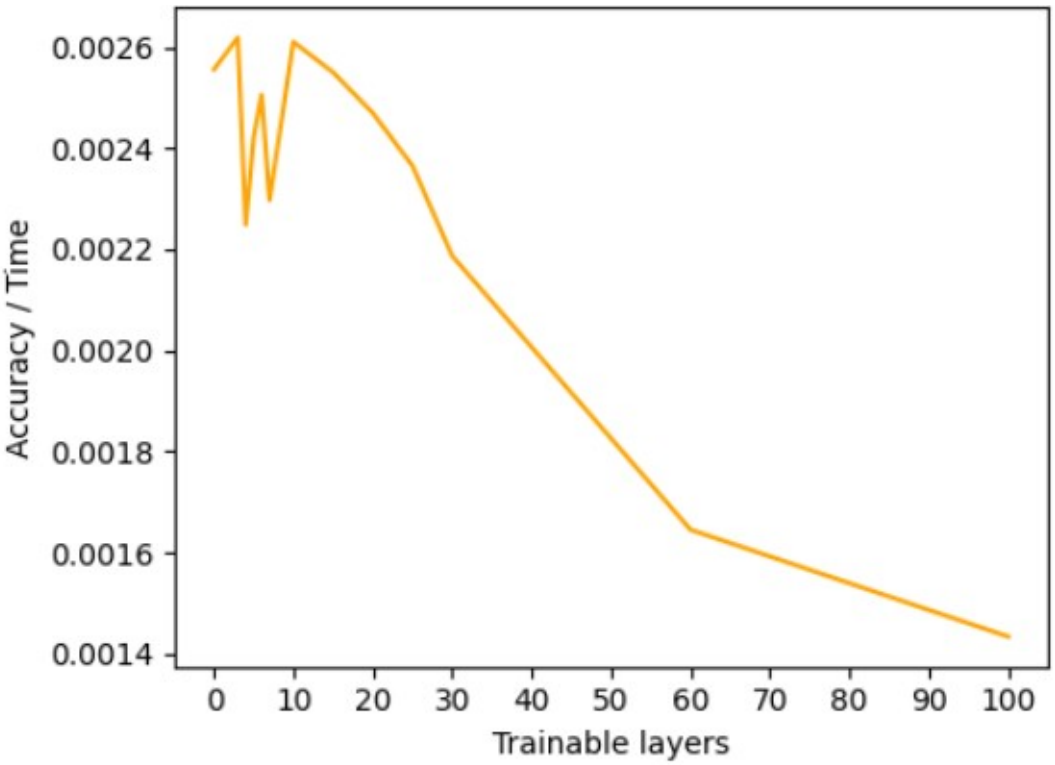
今回の実験によって得られた Table 5.3 から、効率の良いデータは学習層数が 10 を超えるまではほとんど横並びになっており、実験 5.2 の結果と比べると効率の良い学習層数が変化していることが分かった。このことから、モデルの規模はファインチューニングにおける効率の良い学習層数を決定するうえで重要になるのではないかと考えた。

5.5.2 エポック回数の増加に伴う変化

今回の実験では、エポック回数を少なくしたエポック回数を 3 回から 6 回に増やし、実験 5.3 の学習層数を用いて追加で検証した。



(a) Layers and Accuracy.



(b) Layers and Efficiency.

Figure 5.9: Big data Learning Results.

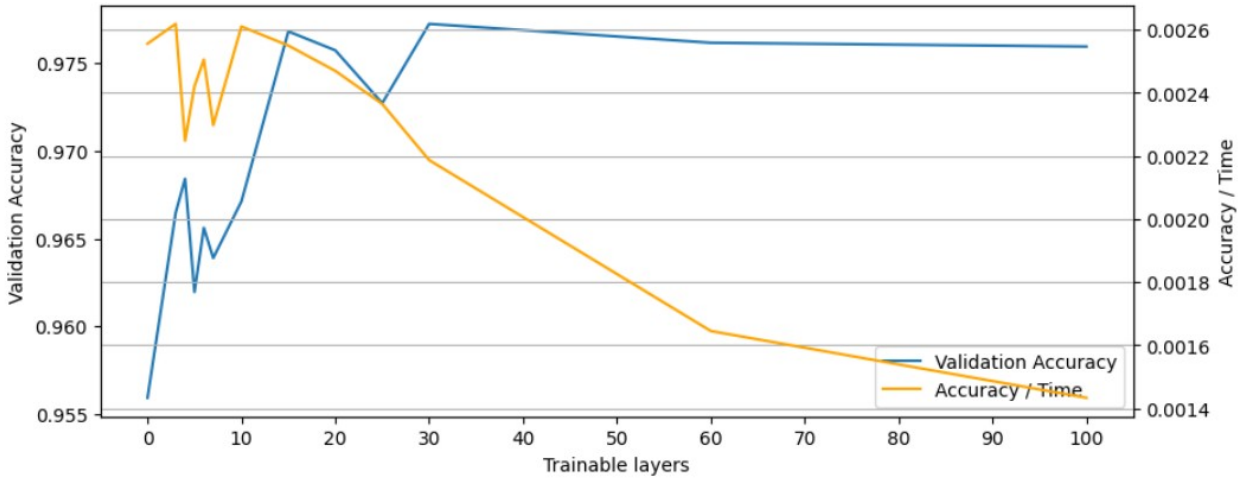


Figure 5.10: Combined Big data Learning Results.

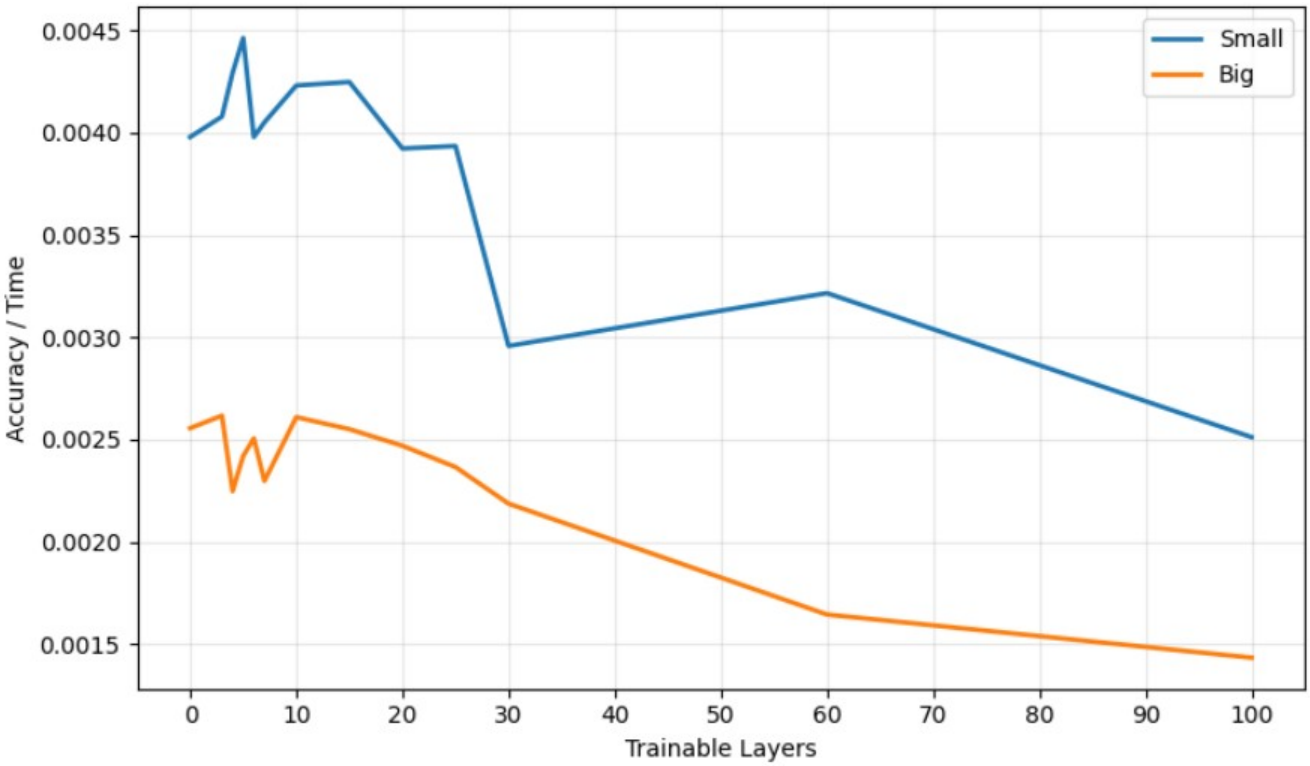


Figure 5.11: Combined Small and Big data Results.

Table 5.3: Small and Big data.

	Small			Big		
Layers	Acc	Time(s)	Eff	Acc	Time(s)	Eff
0	0.938	235.8	0.003979	0.956	374.0	0.002556
3	0.968	237.3	0.004079	0.966	369.1	0.002617
4	0.970	225.9	0.004293	0.968	430.7	0.002248
5	0.994	222.7	0.004463	0.962	397.4	0.002420
6	0.991	249.0	0.003979	0.966	385.4	0.002506
7	0.996	245.9	0.004051	0.964	419.5	0.002298
10	0.991	234.2	0.004231	0.967	370.5	0.002610
15	0.995	234.2	0.004248	0.977	383.1	0.002551
20	0.996	253.9	0.003923	0.976	395.1	0.002470
25	0.991	251.9	0.003935	0.973	411.3	0.002366
30	0.992	335.4	0.002958	0.977	446.9	0.002187
60	0.993	308.7	0.003216	0.976	593.3	0.001645
100	0.996	396.5	0.002512	0.976	680.7	0.001434

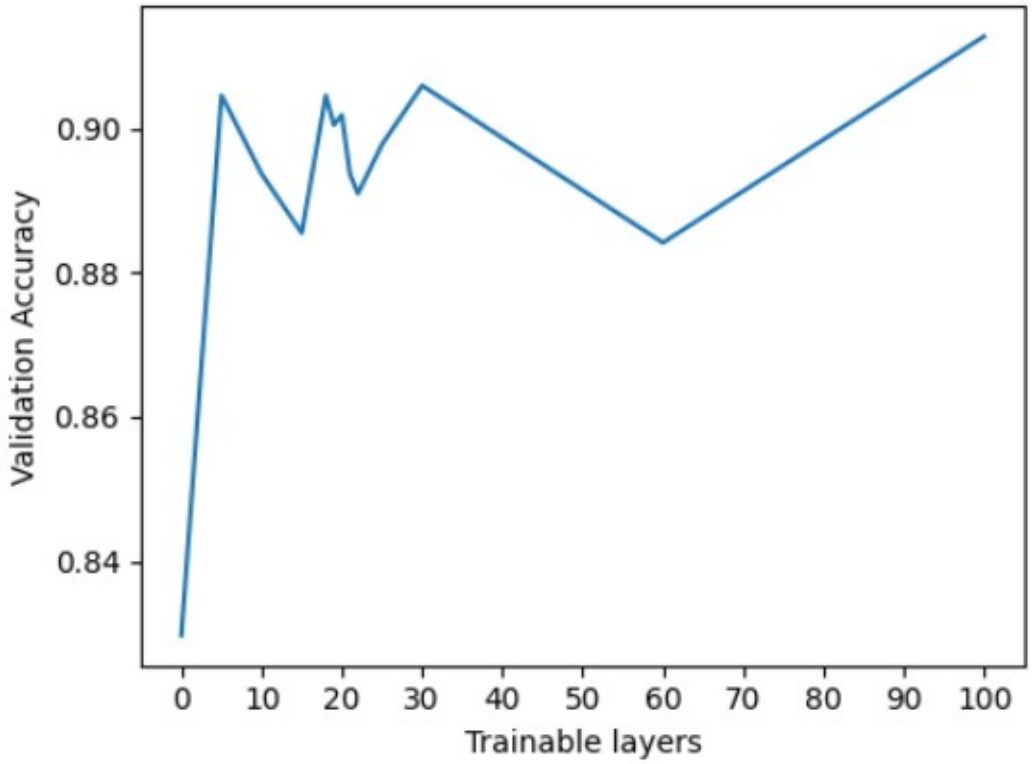
Table 5.4: Epoch=3 and Epoch=6 Results.

	Epoch = 3			Epoch = 6		
Layers	Acc	Time(s)	Eff	Acc	Time(s)	Eff
0	0.704	317.9	0.002215	0.830	645.7	0.001286
5	0.884	377.6	0.002341	0.905	756.1	0.001197
10	0.856	332.0	0.002578	0.894	703.9	0.001270
15	0.875	383.7	0.002281	0.886	653.1	0.001357
18	0.864	352.9	0.002448	0.905	713.2	0.001269
19	0.872	392.9	0.002219	0.901	759.9	0.001186
20	0.861	396.9	0.002169	0.902	794.0	0.001136
21	0.879	347.3	0.002531	0.894	556.3	0.001607
22	0.892	311.1	0.002867	0.891	665.9	0.001338
25	0.869	401.9	0.002163	0.898	685.5	0.001310
30	0.876	405.0	0.002163	0.906	657.1	0.001379
60	0.865	507.2	0.001705	0.884	818.8	0.001080
100	0.841	634.8	0.001325	0.913	1113.4	0.000820

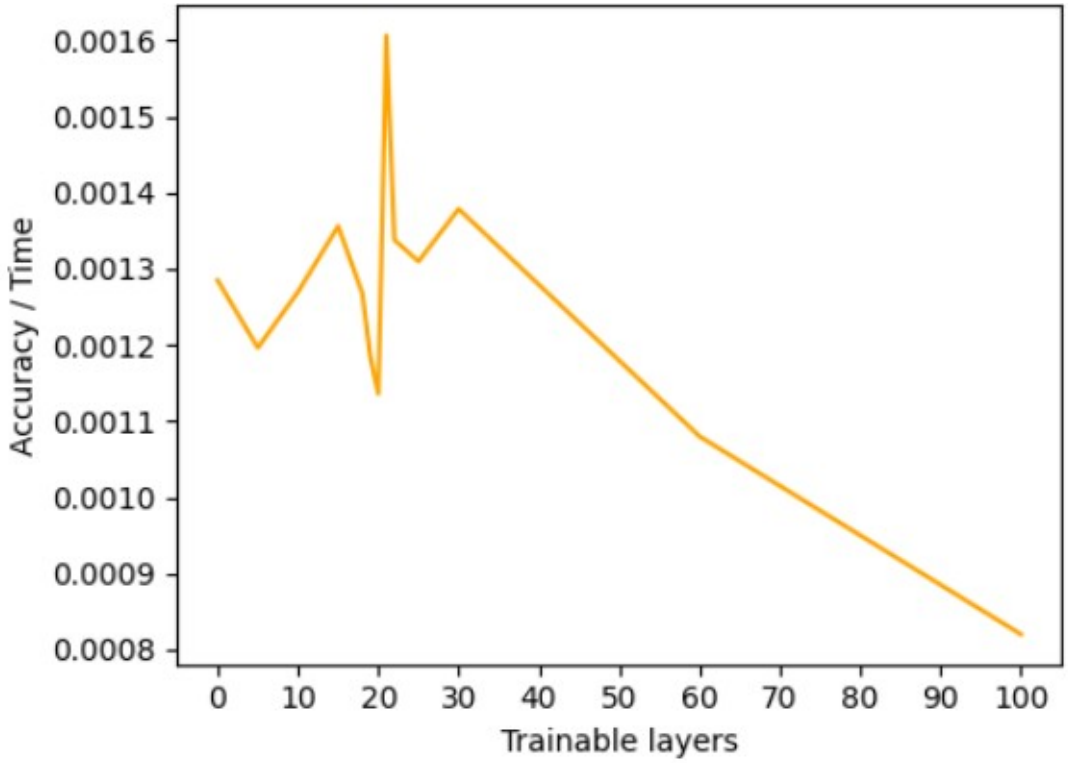
学習層数と正解率を解凍層数と効率をグラフに表したものを Figure 5.12 それぞれのグラフを合わせたものを Figure 5.13 として表し、実験 5.3 の効率と今回得られた効率のグラフを合わせたものを Figure 5.13 として表し、各グラフから得られたデータを表にし Table 5.4 として表した。表から実験 5.3 と比較して効率の良い学習層数はおおむね同じ結果となった。

エポック回数を 2 倍にした結果、計算時間も約 2 倍となり、全体の効率は半分となった。グラフと比較してみてもグラフの表れ方は似通った結果となり、最も効率の良かった学習層数は 21 と実験 5.3 の結果である 22 と 1 層しか違いが無かったため、エポック回数を多少増やしても計算時間が長くなり、かえって効率が悪くなるのみであるということが分かった。しかし、実験 5.3 と比較してエポック回数を増やしたデータの方が浅い学習層数の効率の変化の起伏が緩やかになった。

このことからエポック回数はファインチューニングにおける学習層数の全体の効率を



(a) Layers and Accuracy.



(b) Layers and Efficiency.

Figure 5.12: epoch=6 earning Results.

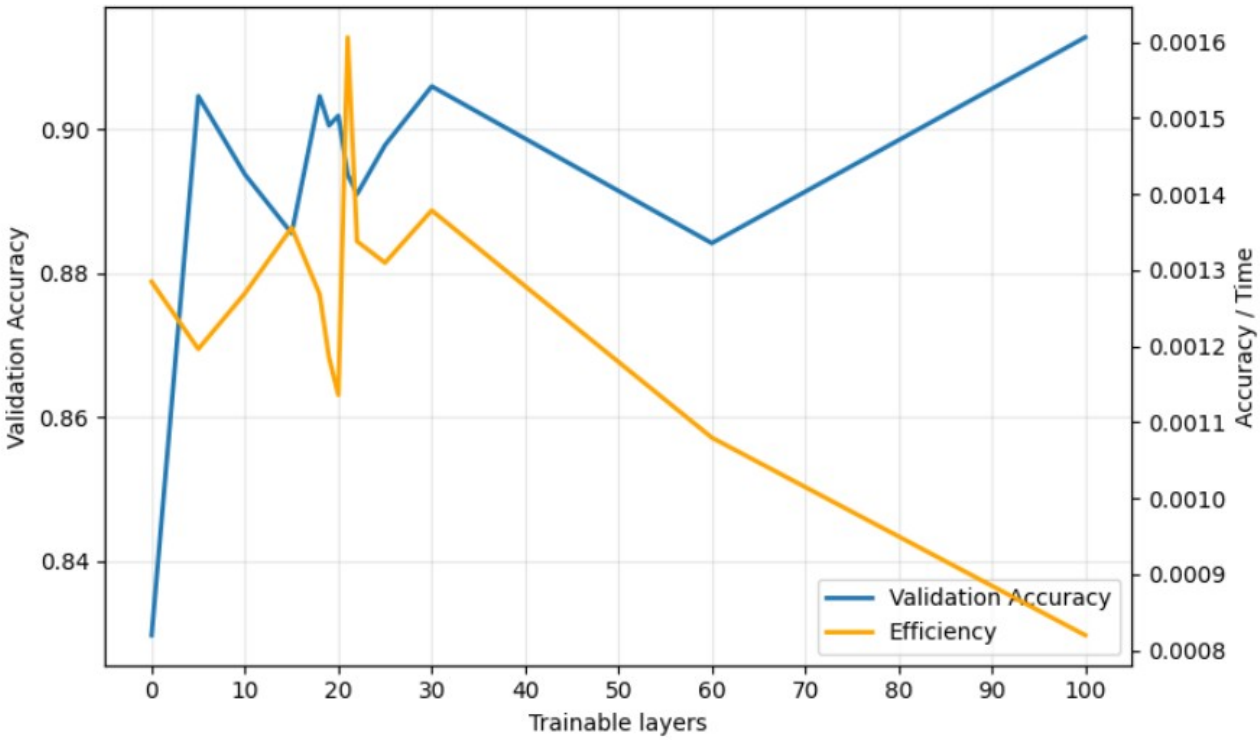


Figure 5.13: Combined Layer Learning Results.

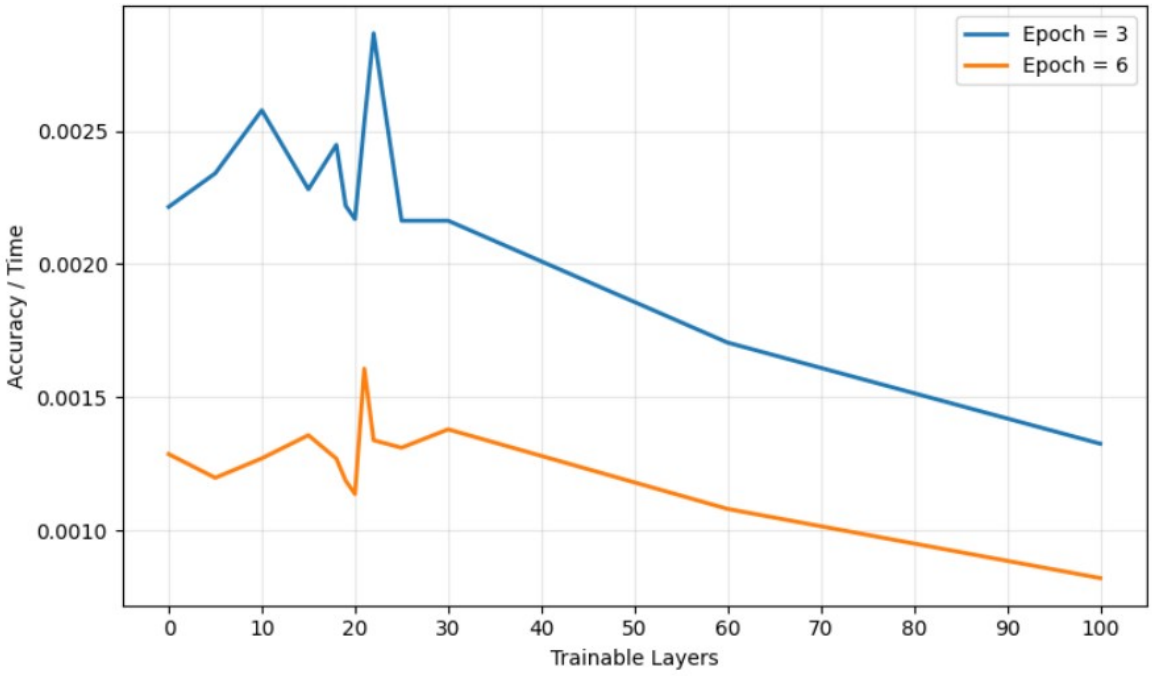


Figure 5.14: Combined epoch Results.

測る際のグラフの起伏を安定させることに効果はあるが、学習層数の決定にはあまり関係が無いと分かった。

5.6 考察

今回の実験でのすべての結果で最も効率の良かった層を抜粋し表として表したものを Table 5.5 とした。

表より、画像分類問題に関して扱うデータが複雑になるクラス数を増やすことや学習に使用するデータ量を増やすと浅い学習層数ごとの正解率が落ちていき、効率の最もよい層数にいたるにはより深い層まで学習する必要があり、学習層数に大きく影響されることが考えられる。一方で、エポック回数を増やしても正解率にはほとんど変化が見られず計算時間のみが伸びて効率が下がってしまったためこちらは学習層数にあまり影響しないということが考えられる。

Table 5.5: Number of layers, accuracy rate, and results for each experiment

Experiment	Layer	ACC	EFF
5.3	5	0.994	0.004463
5.4	10	0.967	0.002610
5.5.1	22	0.892	0.002867
5.5.2	21	0.894	0.001607

第6章　結論

本研究では、ファインチューニングの学習層数による影響についての研究を行い、目標の一つとして最も効率の良い学習層数を探索していった。その過程で分かったことは、以下のことが挙げられる。

1. 扱うデータが複雑であるほどに効率の良い結果を得るために必要な学習層数はより深くなる。
2. 小規模なデータ量によるファインチューニングは1層ごとの計算時間が不安定になってしまう。
3. エポック回数はより正確なデータを取るためには有用だが、学習層数にはあまり影響しない。

扱うデータが複雑であるほどに学習層数はより深くなるとは、本研究で取り扱った2クラス犬猫分類と5クラス花分類において効率の良かった学習層数に大きな開きがあったことであり、2クラス分類のように識別が容易なデータは浅い層数、多クラス分類のように識別が難解なものであると深い層数まで学習が必要である。

今回の実験では、小規模なデータ量のものを用いたため学習層数ごとの計算時間が不安定になり、良いデータであったとは言えず、この結果を他のプログラムに適用することは推奨できない。

しかし、大規模なデータ量のものを用意することで学習層数ごとの計算時間の差を小さくし、より効率の良い学習層数を得られると考えられることが分かった。

参考文献

1) 我妻幸長, はじめてのディープラーニング 2　Python で実装する再帰型ニューラルネットワークと VAE、GAN,SB クリエイティブ株式会社,2020
2) 田村雅人・中村克行,Python で学ぶ画像認識　機械学習実践シリーズ, 株式会社インプレス,2023
3) AISmily, 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/, (2026/02/10 閲覧)
4) AI 実装検定のブログ, ファインチューニングと転移学習の違いについて,https://kentei.ai/blog/archives/1999, (2026/02/11 閲覧)

謝辞

本研究をするにあたって、研究の方向性や目標の設定など様々な指導をしてくださった出口利憲教授に厚く御礼申し上げます。