筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

写真共有サービスのディープラーニングを用いた分類・分析システムの開発 –学習機と識別機の実装・画像の分類器の作成–

羊　寛

修士（工学）

（コンピュータサイエンス専攻）

指導教員　高橋　伸

2017 年 3 月

概　要

Tunnel 株式会社が運営する RoomClip は、インテリアの写真の共有に特化したソーシャル・ネットワーク・システム（SNS）である。RoomClip には、2016 年 6 月の時点で 140 万枚という膨大の数の写真データがアップロードされたが、それを検索するのはユーザーが手動に追加したタグが必要である。だが、そのタグ機能にいくつかの問題点があり、Tunnel 社は、写真の内容をディープラーニングの技術で分類し、検索とタグ機能を向上できる手段を準備したいと考えた。本研究開発プロジェクトは、ディープラーニングの手法を運用し、RoomClip に投稿された写真の内容を認識し、自動分類するシステム「D-ROOM」を開発した。

筆者は、本システムの礎石となる識別機と学習機の開発を担当し、教師データの学習により識別に必要な分類器を作成した。更に、識別の精度向上のための実験、およびプロジェクト全般の技術サポートを行った。本報告書は、プロジェクトの背景、プロジェクトの目標と方針、プロジェクトの体制、使用する手法と技術、システム全体の構成および筆者の担当部分の詳細について述べる。

目　次

図目次

表目次

第 1 章　はじめに

本報告書は、筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻の高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム（以降「高度 IT コース」と略称）における、研究開発プロジェクトとして実施した「写真共有サービスのディープラーニングを用いた分類・分析システムの開発」の成果を述べるものである。

1.1　プロジェクトの背景

SNS(Social Networking Service)とは、WWW 上の会員制コミュニティサービスの総称である。会員同士で紹介しあうことで、相互に友人として登録するというシステムを採用しているものが多い。各種の情報を発信できる汎用型の SNS が存在する一方、特定の内容に特化したものも存在する。

総務省の平成 26 年版情報通信白書[1]によると、近年、10～20 代の若年層をはじめとした幅広い層で SNS の利用率は大幅に上昇しており、特に Facebook と Twitter など国際的に有名な SNS では、20 代の利用率が Facebook で 57.0%、Twitter で 47.1%に達した。ICT 総研の「2016 年度 SNS 利用動向に関する調査」[2]によると、日本について、2015 年末に SNS の利用者数が 6,872 万人、2017 年末に 7,486 万人へ拡大する見込みである。数多くある SNS の中には、Twitter、Facebook など汎用型 SNS 以外に、特定の運用分野に特化した SNS も流行している。

Tunnel 株式会社（以降、Tunnel 社と略称）が運営する RoomClip[3] は、2012 年 5 月からサービス開始した部屋のインテリア写真の共有に特化した SNS である。RoomClip の利用者がインテリア写真を投稿でき、他のユーザーとの交流、さらにコンテストに参加することもできる。

RoomClip は、日本のインテリアに関する SNS の中にも規模が最大である。2016 年 6 月現在、RoomClip に投稿された写真が 140 万枚を超えて、登録したユーザーの数も 85 万を超えた。本報告書の作成が始まった 2016 年末では、投稿された写真の数が更に増加し、150 万枚に達した。今後、インテリア業界とのコラボレーションにより、その規模が更に大きくなる可能性が高いと予想できて、他の写真共有サービスとの競争も激しくなるであろう。

1.2　顧客が抱く問題と要望

本プロジェクトは、前述の通り、本社の所在地が東京都文京区にある Tunnel 社が顧客となる。Tunnel 社が 2012 年 5 月に、インテリア写真の共有に特化した SNS サービスの RoomClip を開始した。4 年間の発展で、RoomClip はインテリアに関する日本最

大級の SNS になった。

現在、RoomClip は写真共有サービスとしての基本機能（ユーザー管理、写真のアップロード、タグの追加と管理、写真の検索、フォロー、コメント）を完備している。図 1.1 は、RoomClip の Web アプリケーションの写真の画面を示している。

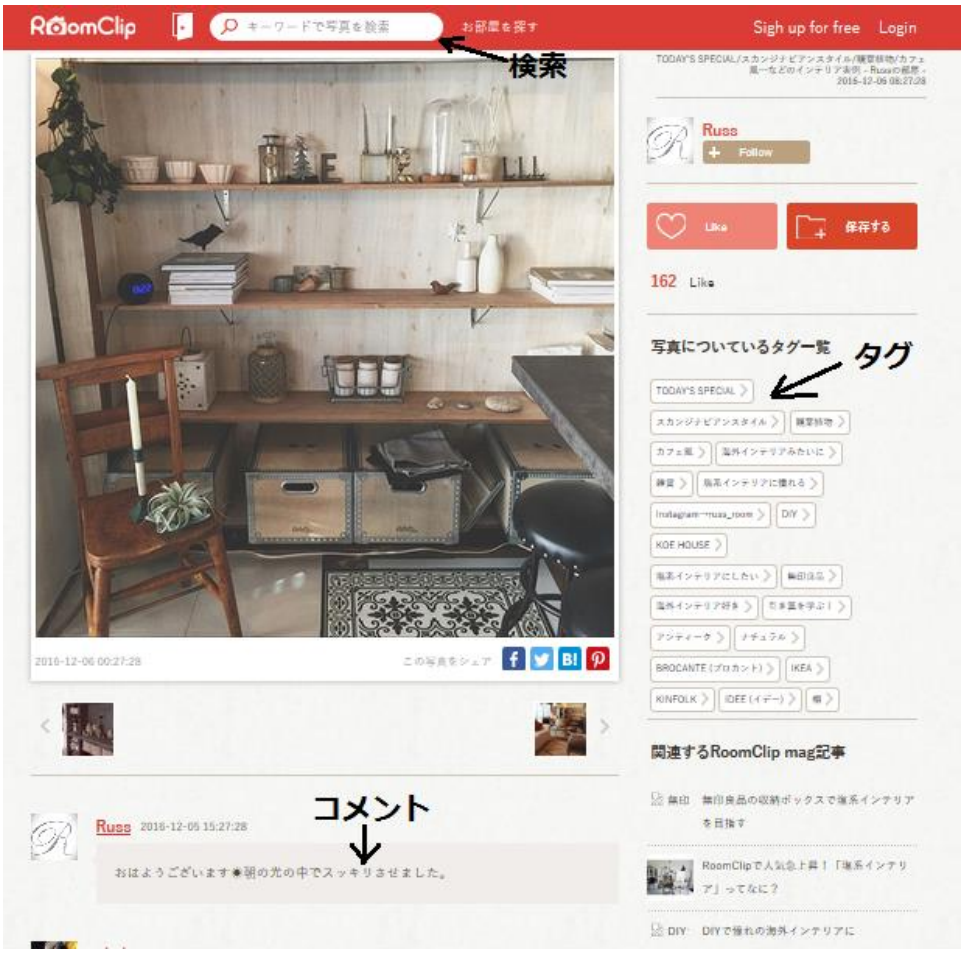

図 1.1　RoomClip の写真表示の画面

しかし、現在の RoomClip には、ユーザー体験に影響する問題点がいくつかある。

1)写真の内容で検索する機能がない

RoomClip にある膨大な数の写真を検索するのは、写真をアップロードしたユーザーが入力した写真のタイトル、説明文とタグなど写真の補足情報が必要である。しかしながら、RoomClip のユーザーはインテリアの専門家では限らないので、入力された補足情報、特にタグの正確性は必ずしも正しいとは言えない。これで、検索の精度が下がっており、RoomClip の運営者が現在の運営状況を分析する際、写真のサンプルを採取する手間が増える。もし、写真を補足情報のみならず、その内容を用いて検索できれば、この問題の軽減は可能になる。

Google や Baidu など海外の大手 Web 系企業は、既に画像の内容による検索機能

を実現した。そのため、RoomClip に似ている機能を実現するのは、技術上には問題がないと考えられる。

本書の第 3 章で、開発チームのメンバーの視点からこの状況を詳しく分析した。

2)新しい面白い機能が求められる

RoomClip と他の写真共有サービス（Instagram、Flickr など）と比べると、インテリア写真に特化した以外、機能面ではほぼ同等である。使用する技術も、ほぼ既存のものであり、技術的には他のサービスと差別が小さい。インテリア業界とのコラボレーションというビジネスモデル以外に、独特な新機能の導入により更なる差別化は必要である。

Tunnel 社は以上の問題を考慮し、最近が話題になったディープラーニングの技術を導入したいと考えた。その前、Tunnel 社はディープラーニングによる画像認識のプロトタイプ「インテリアスタイル判定器」[4]を作ったが、現在ではそれを新しいサービスとして RoomClip に実装していない。今回、Tunnel 社は、検索とタグ機能の向上に向かって、ディープラーニングによる画像認識の技術を運用し、RoomClip の画像を使って新しい機能を実現するという要望を本プロジェクトのメンバーに提出した。

ディープラーニングに関する紹介と先行研究は、本書の第 4 章で述べる。

1.3　分析対象となるデータ

本プロジェクトの分類と分析対象となるデータは、既に RoomClip にアップロードされたインテリア写真のデータと、ユーザーが新しくアップロードするインテリアの写真のデータである。

本書の第 6、7 章で述べるが、ディープラーニングでインテリア画像を分類するには、各分類に一定の数のサンプルデータ、すなわち「教師データ」が必要である。

Tunnel 社は、登録ユーザーの情報（一部の個人情報が削除されたもの）、写真とタグの情報、写真とタグの関連、ユーザーと写真の関連を含めた、2016 年 6 月時点の RoomClip のデータベースの一部を本プロジェクトに提供した。さらに、RoomClip にアップロードされた全画像のアクセス権も、我々開発チームに提供した。そこで教師データは、既にアップロードされた RoomClip の写真データから抽出できるようになり、その写真データの全てを分析することもできるようになった。

本プロジェクトで開発するシステムは、ユーザーが新しくアップロードしたインテリア画像をディープラーニングで作成した分類器に通して、その分析結果に基づくサービスを提供する。実現するサービスは、次の章で紹介する。

1.4　本報告書の構成

本報告書は、本文全 11 章と、一部補足のドキュメントなど付録で構成されている。第 2 章では、プロジェクトのメンバーが考えて決めたプロジェクトの目標とアプローチについて詳しく述べる。第 3 章では、プロジェクトの体制、役割分担、連絡手段とスケジュールなど運営面について述べる。第 4 章では、本プロジェクトに使用する手法と技術を簡単に述べる。

第 5 章で、本プロジェクトで開発したシステム全体の各部分について説明するが、筆者が担当した学習機と識別機については、第 6 章で詳述する。

第 7 章は、インテリア写真を分類するためのデータセットの構成と、分類器を作成するための学習の実験について話した。

第 8 章は、筆者が行った分類の精度向上のための試行錯誤について詳しく述べるが、第 9 章ではこの開発で筆者がチーム全体への技術サポートについて話す。

最後に、第 10 章と第 11 章では、本プロジェクトの振り返りと今後の展開について述べ、全書をまとめる。

付録では、各データセットのファイルの取得方法、プログラム中身の説明、ファイルの一覧、実装した各プログラムの使い方、サーバーに配置する際の設定など補足情報を添付する。

第 2 章　プロジェクトの目標と方針

前述の Tunnel 社の要望に基づき、本プロジェクトは、ディープラーニングの技術を利用して RoomClip にアップロードされた大量の画像の内容を認識し、自動分類してその結果を分析するシステムの開発を行う。

我々開発チームは、Deep Learning の頭文字「D」と、RoomClip の「Room」で、開発するシステムを「D-ROOM」と名付けた。本書は、本プロジェクトで開発するシステムを D-ROOM と呼称する。

2.1　現状の分析

筆者を含む本プロジェクトのメンバーは、RoomClip を実際に利用し、その体験に基づいて RoomClip のユーザーと経営者が抱く問題を自分の視点から分析した。

RoomClip について最も重要な機能は、写真のタグ管理と写真の検索だと言える。利用者が写真を RoomClip にアップロードする時、自由にその写真にタグを追加できる。他の利用者は自分が閲覧したい写真を検索する時、そのタグが最も優先度が高いキーワードになる。

しかし、RoomClip の利用者の中に、そのタグ機能を有効に活用しない人は少なくない。自分が共有したいインテリアの写真のスタイルが知らなかったり、写真にある物を表すタグだけ一つか二つ追加したりする利用者が多い。

さらに、写真全体の雰囲気を考えず、一つ二つオブジェクトだけで、インテリア全体のスタイルを表すタグを画像に追加する利用者も居る。ここで二つの例を挙げる。

◎コカコーラの缶があるだけて、畳が見える伝統的な和室を「アメリカン」というスタイルを表すタグを付ける。
◎芝生の写真に白い犬がいるだけで、画像に「ホワイトインテリア」というスタイルを表すタグを付ける。

RoomClip に対して、そのような状況による影響を無視するのはいけない。現在、RoomClip の検索機能に使われる最優先の検索条件は、画像に付けられたタグである。ユーザーが画像にタグを追加しなかったり、間違ったタグを追加したりことにより、検索の精度にインパクトがある。

2.2　実現する機能

RoomClip の現状を改善するため、我々チームが D-ROOM で実現したい機能は、主

に二つである。その一つは、システムにアップロードされた部屋のインテリア画像を、ディープラーニングを用いてスタイルに対応するカテゴリに分類し、その分類結果により、ユーザーにスタイルに関するおすすめのタグを提示する機能である。

もう一つ実現するのは、このサービスで入力した全ての写真、そして RoomClip に既にアップロードされた全ての写真を分類し、生成した分類結果を統計し、図と表で RoomClip の運営者に示す統計の機能である。

さらに、全ての分類結果を用いて、分類別に画像の一覧を表示、類似の画像の検索などユーザー、運営者両方向けの機能にも実現する。

筆者が担当したのは、これらの機能の礎石である画像分類バックエンド、すなわち画像を分類できる識別機をディープラーニングの技術で実現する。その識別機は、分類器というデータモデルが必要なので、それをディープラーニングで生成する学習機も必要である。詳しくは、本書の第 6 章と第 7 章に述べる。

2.3 本システムで実現する価値

本プロジェクトで実現するシステム「D-ROOM」は、以下の価値を顧客に提供できる。

タグの推薦機能により、RoomClip のユーザーがタグ機能をさらに正しく利用できる。これで、RoomClip の検索機能がより有効になるであろう。ユーザーが RoomClip 検索を利用する可能性が高まり、インテリアのスタイルの知識を得て、新しい発見と出会いも増える。

RoomClip の運営者は、D-ROOM が出力した現在の分類結果の統計により、RoomClip にアップロードされる画像にどのようなスタイルが多い、ユーザーにどのようなスタイルが流行していることが分かる。さらに、現在のインテリアの市場状況をより正確に把握し、より面白いイベントを提供でき、RoomClip の知名度とユーザー数を向上できる。

海外の Web 業界の大手、例えば、Google[1]や Baidu[2]、Yahoo![3]などが現在話題となったディープラーニングの技術を積極的に注力している。我々は、現在のトレンドとなるこの技術を利用した機能の追加により、既存技術で実現した RoomClip に新しい面白さを加え、他の写真共有サービスと更に差別化することができると考えた。

---

[1] https://deepmind.com/applied/deepmind-for-google/
[2] http://research.baidu.com/institute-of-deep-learning/
[3] http://yahoohadoop.tumblr.com/post/139916563586/caffeonspark-open-sourced-for-distributed-deep

第 3 章　プロジェクトの体制

3.1　プロジェクトの関係者構成

本プロジェクトの顧客は、Tunnel 株式会社（以下、Tunnel 社と略称）である。筑波大学大学院システム情報工学研究科の岡瑞起准教授および橋本康弘助教が課題担当教員とし、同研究科コンピュータサイエンス専攻高度 IT コースの藤田、羊、張および ZHANG の 4 人が開発チームを結成した。

本プロジェクトに当たり、関係者の構成は表 3.1 で示す。

表 3.1　プロジェクトの関係者一覧

所属	担当	氏名
Tunnel 株式会社	顧客	平山知宏　氏
筑波大学大学院　システム情報工学研究科	課題担当教員	岡　瑞起　准教授
		橋本康弘　助教
	開発チームのメンバー	藤田卓也
		羊　寛
		張　墾
		ZHANG SHUNAN

3.2　開発チームの役割分担

プロジェクトの初頭、チームメンバーの具体的な役割分担が決定されなかった。プロジェクトの目標と方針を決めた際、各メンバーの得意分野により、チーム内に二つの班に分け、役割分担を決定した。

表 3.2　チームメンバーの役割分担

班	内容	氏名	担当の範囲
バックエンド班	画像分類バックエンドの開発	藤田卓也	チームリーダー、学習と識別の速度向上、機材の管理
		羊　寛	学習機と識別機の実装、識別の精度向上、チーム全体の技術サポート
サービス班	画像分類を用いるサービスの開発	張　墾	データベースの設計、構築と管理、教師データ作成協力
		ZHANG SHUNAN	フロントエンドの開発、データベースの設計

筆者の担当範囲に関する詳しくは、後ほど第 5 章で述べる。

3.3　連絡手段

開発チームは、長期休暇の期間以外、原則的に週に 1 回課題担当教員と会議を行い、進捗の報告、事務上の連絡とプロジェクトに関する討論を行う。議事録は、原則的に筆者が書くが、Google のファイル共有サービス Google Drive を利用したので、開発チーム全員と課題担当教員はそれの閲覧や編集が可能である。課題担当教員や開発チームのメンバーが学校に来ない場合、Skype を利用して会議に参加する。

顧客の Tunnel 社との連絡は、主に Slack[1]のグループで行う。それの利用により、顧客、課題担当教員と開発チームはプロジェクトの進捗および他の情報を共有できる。

開発チーム内部は、Google Drive で開発のドキュメント、議事録とメンバー各自の進捗状況を保存し、共有する。

それ以外、開発チームと課題担当教員は、不定期的に Tunnel 社に訪問し、顧客の要望を聞く。その第 1 回の訪問は、2016 年 5 月 27 日に行った。

3.4　プロジェクトのスケジュール

本プロジェクトで行った開発は、時系列で見ると、主に 3 段階がある。本書ではよく「第 1 段階」など言葉を使うので、ここで各段階の時間、その目標と達成状況について述べる。

図 3.1 はそのスケジュールを示すものである。

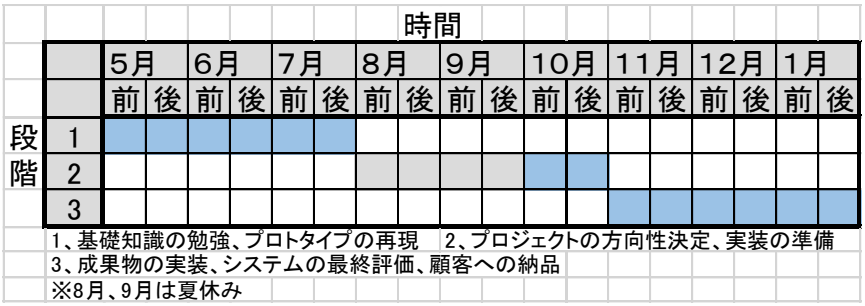

図 3.1　プロジェクトのスケジュール

第 1 段階は、開発チームがディープラーニングに関する知識がなかった。顧客の要望を実現する能力を鍛えるため、その段階の目標は、基礎知識の学習、Tunnel 社のプロトタイプの再現（学習機、識別機、分類器と簡単なフロントエンドの実現）と決めた。この段階の目標は、スケジュール通り 7 月末に完成した。第 6、7 章で詳述するが、筆者がこの段階に、プロトタイプに基づいて学習機と識別機を実装し、インテリア写真のカテゴリ化により、教師データセットを作成し、第 1 回の分類器の生成と画像認識の実

[1] https://slack.com/

験を行った。

基礎知識の勉強が終わってから、第 2 段階の目標はプロジェクトの方向性の決定となった。本来 8 月に完成する予定だが、夏休みでメンバーの不在など原因で、方向性の決定は遅延が出で、10 月に完了した。ちなみに、この段階にメンバーが前述の役割分担を行って、各自に成果物の実装の準備を行った。筆者が所属するバックエンド班は、9 月に入ってから、後述の精度向上の実験が既に開始した。

最後の第 3 段階で、サービス班が全力にフロントエンドの実現に力を入れる一方、バックエンド班は認識精度と速度向上の実験を続けて、フロントエンドに提供する最終版の画像分類バックエンドを完成した。筆者は、データセットを更にチューニングに加えて、学習機と識別機のプログラムの共通部分にも修正を加えて、精度向上の効果を確認できた。

第 4 章　使用する手法と技術

4.1　ディープラーニングの概要

ディープラーニングは、日本語では「深層学習」と呼ばれる。深層学習は、多層のニューラルネットワークを用いた機械学習の方法論である[5]。音声、画像などを対象にする問題に対し、他の方法を圧倒する高い性能を示すことが分かり、多くの研究者や技術者の関心を集めている。

情報処理の観点から見たディープラーニングの本質的な意義は、観測データに内在する本質的な構造、しかも単純な構造ではなく、簡単な構造から複雑な構造へと階層化された潜在構造をデータからの学習によって獲得することにある[6]。インテリアの写真のスタイルのような顕在的ではない特徴を、ディープラーニングによって抽出し分類するのは、我々のプロジェクトには望ましい。

4.2　畳み込みニューラルネット

畳み込みニューラルネット(Convolutional Neural Network, CNN)は、主に画像認識に応用される順伝播型ネットワークである。長年の難問である 1 枚の画像からそこに写る物体のカテゴリ名を認識する物体カテゴリ認識という問題は、CNN で解決しつつある[7]。

Y. LeCun ら[8][9]によると、MNIST という、60,000 枚の手書き数字の画像のデータセットに対して、11 種類の自動分類アルゴリズムで分類の精度、認識の所要時間と学習の所要時間について比較実験を行った。結果として、Y. LeCun らが郵便番号の画像データセットを高い精度で分類できた、順伝播形の畳み込みネットワーク「LeNet 1」に、入力層のサイズの拡大、更に、R. Schapire 氏[10]が提出した第 1 回の分類に確信度が低いデータを再学習する「boosting」など手法を加えた「Boosted LeNet 4」は、11 種類の分類アルゴリズムに誤差が最も低い 0.7%を達成した。

手書き数字以外も、畳み込みニューラルネットによる、実際のオブジェクトの写真の分類への運用にも進んでいる。A. Krizhevsky ら[11]によると、1000 カテゴリがある高解像度の画像データベース ImageNet に対して、多重の畳み込み層(Convolutional Layer)を持つ畳み込みニューラルネットで、合計 120 万枚の画像がある学習用データセットを学習し、第 1 位の誤差が 37.5%、上位 5 位の誤差が 17.0%という先行研究より高い分類の精度を実現した。

今回、我々のプロジェクトに、知識の勉強と実装の時間が限られたことで、D-ROOM がインテリア写真の分類に使用するものは、A.Krizhevsky らのものと似ているが、簡単になった畳み込みニューラルネットである。第 6 章で学習機と識別機の実装について

説明する際、CNN の技術について少し深く述べる。

4.3　ディープラーニングのフレームワーク TensorFlow

TensorFlow[12]は、もともと Google 社内の機械学習研究部門が使っていたフレームワークであったが、2015 年にオープンソースになった。ディープラーニングの研究のため作られ、研究用プロトタイプから運用システムに移行するのも容易である。TensorFlow のコアは Python と C++で作られ、使用時は Python でそのクラスを引用して使う。テキストや画像、音声など入力データの解析、低レベルのデータ構成とアルゴリズムは TensorFlow 内部で既に実装されているので、ローレベルのプログラミングを考えず、機械学習の自身のアルゴリズムの実現に集中できる。

TensorFlow の動作環境は、主に Linux である。畳み込みニューラルネットの学習は膨大の計算量が必要なので、マルチコア CPU だけではなく、GPU で一般の計算を加速する API、nVidia の CUDA[1]も運用できる。第 7、8 章では、その CPU と GPU の学習速度について述べ、GPU による学習速度の向上が示される。

TensorFlow を使用するため、今回 D-ROOM の学習機、識別機および一部のツールは Python 2.7 にて実装した。その詳細は、後ほど第 6、8 章に紹介する。

4.4　Web フロントエンドとデータベース

筆者の担当範囲ではないが、ここで D-ROOM のフロントエンドとデータベースに使用する技術を簡単に紹介する。

フロントエンドを担当するメンバーは、スクリプト言語の PHP と、リレーショナルデータベースの MySQL で Web アプリケーションを作成する経験があった。そのため、D-ROOM の Web フロントエンドは PHP で実現し、データベースは MySQL で構築した。

D-ROOM に識別した画像の数が多いので、Web 上での一覧の表示は、AJAX による HTML と画像を動的に読み込む方法が採用された。

---

[1] https://developer.nvidia.com/cuda-zone

第 5 章　D-ROOM 全体の構成

5.1　概要

D-ROOM というのは、ユーザーがアップロードした写真を識別機で分類し、その分類結果によってタグの推薦、類似画像を表示などユーザー向けの機能、更に今まで分類した全ての写真の分類結果の統計、検索など管理者向けの機能を持つ、Web アプリケーションのフロントエンド、CNN による識別機、データベースで構成するシステムである。

TensorFlow で実現する画像データの識別機は、畳み込みニューラルネットワークを表す「モデル」というデータが必要である。画像データの分類に使うため、我々はこのモデルを分類器と呼称する。この分類器は、同じ TensorFlow で実現した学習機を使って教師データを学習させて作成する。第 7 章で紹介するが、その学習には膨大な計算量が必要なので、Web サーバーではなく、GPU を装着した学習専用の PC で行う。

D-ROOM（以下「本システム」と呼称）の主な構成は、図 5.1 で示す。

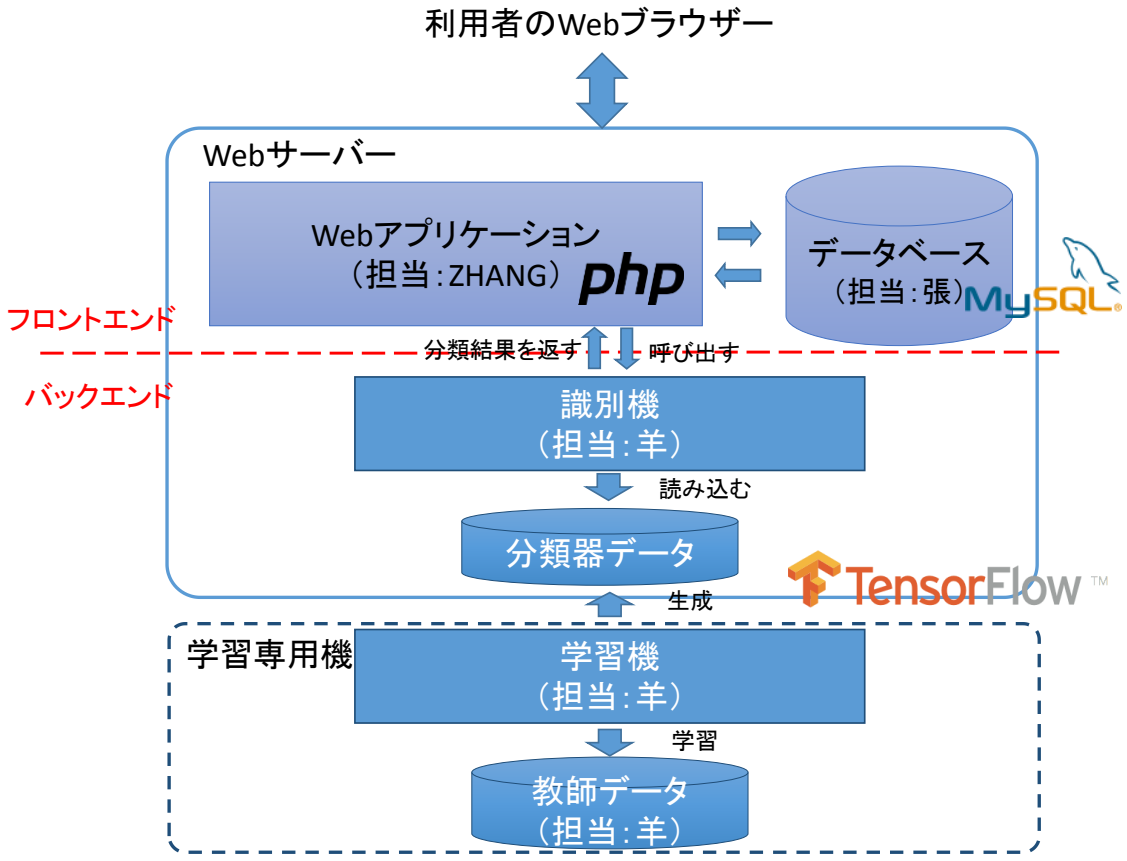

図 5.1　D-ROOM の構成部分

本システムは、主に「フロントエンド」と「バックエンド」の 2 部分に分けられる。

フロントエンド部分は Web アプリケーションとデータベースを含め、開発担当はサービス班である。その一方、筆者が属するバックエンド班が担当したバックエンド部分は、識別機と学習機を含め、学習機に対応する教師データセット、識別に必要な分類器データも含める。

フロントエンドとバックエンドの間のインターフェースというのは、Web アプリケーションが識別したい画像のパスを識別機に送り、識別機が識別の結果を JSON として Web アプリケーションに返すという簡単なものである。

筆者が担当する部分は、主にその学習機と識別機である。分類器データは前述の通り、学習機を使って自動的に生成する。学習機が使用する教師データセットの作成とチューニングも、筆者が担当した。

5.2 バックエンド（学習機と識別機）

学習機の役割は、予め用意した教師データセットを読み込んで CNN のパラメータを学習し、学習が終了した CNN のパラメータを含めるセッションのデータをファイルに保存する。保存したファイルは分類器データとして扱う。

学習に必要な計算量が膨大なので、原則的に学習専用の PC で実行する。

その一方、識別機の役割は、サーバーにある写真の画像ファイルを読み込み、それに対して分類の結果を出力する。Web アプリケーションでユーザーがアップロードした写真を識別（分類）する際、識別機を呼び出す。

識別機は、原則的に Web サーバーで実行する。識別機の実行の必須条件は、学習機により生成した分類器データのファイル以外、Python と TensorFlow がインストールされた Linux のみとなるので、必要な場合、他の PC にも実行できる。

学習機と識別機については、第 6 章で詳しく紹介する。データセットの構成とそれに加えたチューニングは、第 7、8 章で詳述する。

5.3 Web アプリケーション

本システムのもう一つの重要な部分は、Web アプリケーションである。識別機とデータベースなどと情報を交換し、ユーザー（エンドユーザーおよび管理者）に Web ページで機能の実行結果を返す。すなわち、Web アプリケーションは本システムのユーザーインターフェースである。

本システムの機能は、第 2 章で述べた通り、ベースとなるアップロードされた画像の認識（分類）以外に、その分類の結果を利用するサービスもある。分類機能はバックエンドの識別機で実現したが、他の機能は全て Web アプリケーションで実現される。

Web アプリケーションと識別機は、互いに独立性を保った二つのサブシステムであ

る。画像ファイルを分類する際のみ、識別機を呼び出してアップロードされた画像を識別する。

Web アプリケーションの利用シーンは、以下の図 5.2 のユースケース図で示す。

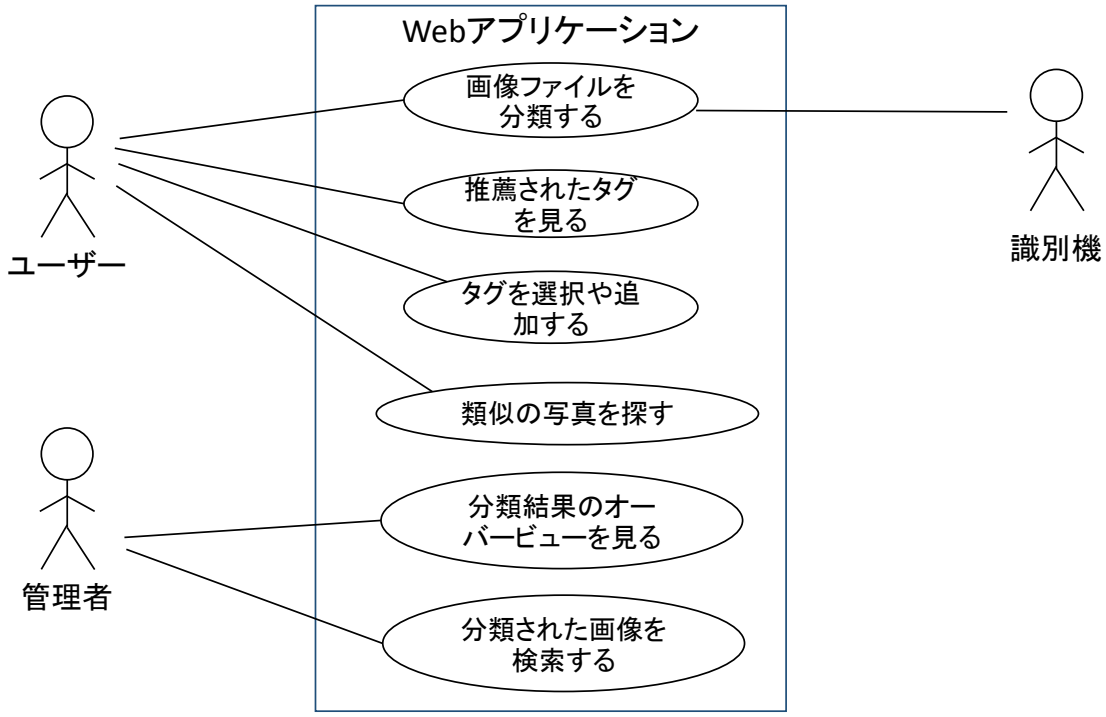

図 5.2　Web アプリケーションのユースケース図

この部分の設計と実装は筆者の担当部分ではないので、その中身の設計と実装の詳細は、本書では割愛する。

5.4　データベース

各分類に対応するスタイル、分類の結果、ユーザーに推薦されたタグを保存するため、本システムは MySQL データベースを利用した。

データベースに記録した情報は、主に画像のファイル名、識別機が出力した画像の分類結果、さらにユーザーが推薦される全タグ、ユーザー自ら追加したタグなどを含める。

データベースの構成は、以下の E-R 図、図 5.3 で示す。

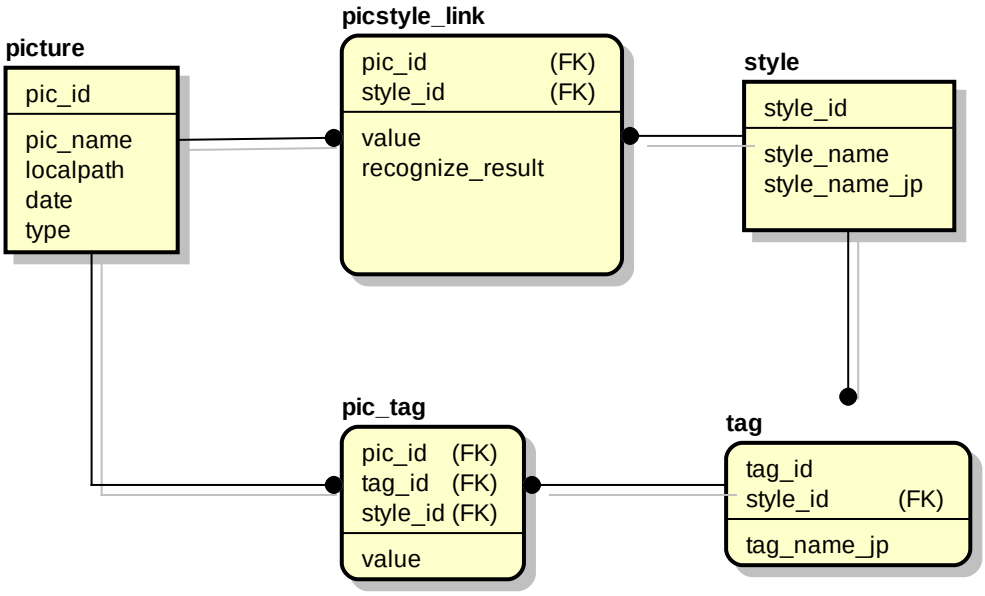

図 5.3　D-ROOM データベースの E-R 図

E-R 図で表示したデータベースにある各テーブルと列の役割は、表 5.1 で説明する。

表 5.1　D-ROOM データベースのテーブルの説明

テーブル名	保存するデータ	列名	役割
picture	写真の情報	pic_id	プライベートキー
		pic_name	ファイル名
		localname	保存するファイル名
		type	写真の種類
		date	写真データ追加日時
style	写真の分類名(スタイル名)	style_id	プライベートキー
		style_name	スタイルの英語名
		style_name_jp	スタイルの日本語名
tag	推薦するタグ	tag_id	プライベートキー
		style_id	style への関連キー
		tag_name_jp	タグの名称
picstyle_link	分類結果	pic_id	picture への関連キー
		style_id	style への関連キー
		recognize_result	識別機が出力した確信度
		value	拡張のための未使用列
pic_tag	写真に追加したタグ	pic_id	picture への関連キー
		tag_id	tag への関連キー
		value	拡張のための未使用列

データベースの作成と管理は筆者の担当範囲外なので、本書では割愛する。

5.5　システムの実行環境

D-ROOM の Web サーバーと学習専用機の実行環境は、以下表 5.2 で説明する。

表 5.2　D-ROOM の実行環境

Web サーバー	CPU	Core i7 3770K
	RAM	16 GB
	ハードドライブ	1 TB (OS) + 6 TB(データ)
	OS	Ubuntu Server 14.04 LTS
	ソフトウェア	MySQL 5.5, PHP 5.5, TensorFlow 0.11, Apache または Nginx
学習専用機	CPU	Core i7 3770K
	RAM	16 GB
	GPU	NVidia GeForce GTX 1070, 8 GB VRAM
	ハードドライブ	1 TB
	OS	Ubuntu 14.04 LTS
	ソフトウェア	TensorFlow 0.11
ユーザーの PC	OS	Windows 8 以上、又は Ubuntu 14.04 以上
	ブラウザー	最新バージョンの Chrome や Firefox

学習専用機は、分類器を作る学習を加速するため、性能が良い独立の GPU を装着する一方、Web サーバーは、膨大な計算量が必要である学習を実行しないため、独立の GPU を使用しない代わりに、写真データと MySQL データベースを保存するため、6 TB の大容量ハードドライブを装着した。

この実行環境は D-ROOM の最終版で決めたことで、その前の開発中に変更があった。本書 7.2 節、8.5 節はこの変更に関する説明が載っている。

第 6 章　学習機と識別機の実装

6.1　Tunnel 社のプロトタイプの勉強

プロジェクトが開始したところ、我々チーム全員がディープラーニングの原理とその応用に関する知識が皆無であった。

2016 年 5 月 27 日、我々が Tunnel 社に実際に訪問し、顧客の平山知宏さんと会議を行った。その会議で、チームが考えたアイデアの討論以外に、プロジェクト第 1 段階（2016 年 7 月まで）の目標を第 1 章に述べた Tunnel 社が開発したプロトタイプ「インテリアスタイル判定器」の再現と決定した。

6 月下旬、我々は Tunnel 社から画像を保存する Amazon S3[1]クラウドの読み込み権限を取得し、それに保存された 6 月時点に RoomClip にある 140 万枚以上の画像を研究室のサーバーにダウンロードした。

機械学習の初心者である筆者は、それをゼロから再現するには時間が足りない。そのため、7 月上旬、我々は Tunnel 社から「インテリアスタイル判定器」の参考ソースコードが入った Microsoft Azure[2]のクラウド PC の管理権を取得し、勉強を開始した。

プロトタイプの構成は、以下 3 部分であった。

◎共用モジュール

◎分類器を作成する学習機

◎RoomClip の画像 URL で分類器を評価する評価スクリプト

それ以外に、サンプルの教師データセットも含めていた。英語のカテゴリ名が付けられた 20 個のカテゴリに、各 1000 枚ぐらいの JPEG ファイルが入っていた。さらに、その教師データの学習で作成したサンプルの分類器データもあった。

我々が手に入れたプロトタイプは、ソースコードのみであり、仕様書などドキュメントが一切なかった。そのため、筆者はそのソースコードを読みながら、それに関する他の参考資料も探していた。

そのソースコードを詳しく読むと、その構成と内容も TensorFlow の CNN チュートリアル[13]（以下、チュートリアルと略称）に使った Apache ライセンスでオープンソースされたサンプルのソースコード cifar10[14]と類似であると分かった。

そのサンプルには、CNN による画像分類に関する基本的な機能が既に実装されている。

◎教師データセットの読み込み

◎共通モジュール（CNN 構成、誤差の計算、パラメータの反映）

---

[1] https://aws.amazon.com/s3/

[2] https://azure.microsoft.com/en-us/overview/what-is-azure/

◎教師データセットを学習する学習機

◎教師データセットに対して識別の平均誤差を評価するプログラム

しかし、このサンプルプログラムは、特定の画像を判定する機能が実装されていないので、それを参考するだけでは、Tunnel 社のプロトタイプを再現できない。

Tunnel 社のプロトタイプは、使用した CNN ネットワークの構成が上記のサンプルのソースコードと同じである。ただし、学習機に入力される画像のサイズは 24×24 から 40×40 に拡大されて、CNN ネットワークに使用したフィルタの一部パラメータも変更された。さらに、識別機もサンプルのバージョンより、RoomClip のサーバーにある画像を認識する機能が追加された。だが、それを実行したところ、識別機に画像を読み込む部分に、リサイズの方法が学習機と異なるバグを発見した。その問題を解決し、認識が可能であることを確認した。

筆者は、サンプルのソースコードと Tunnel 社プロトタイプのソースコード両方を参考にして、我々が開発するシステム D-ROOM に使う学習機と識別機を実装した。

6.2 D-ROOM の学習機と識別機の概要

今回我々チームは、ディープラーニングそのものの研究ではなく、それをバックエンドとしてサービスの実現が目標である。さらに、前述の通り、我々は TensorFlow の cifar10 サンプルプログラムと Tunnel 社のプロトタイプのソースコードを入手した。

そのため、筆者は学習機と識別機をゼロから開発するより、既存のコードを再利用する方針を採った。代わりに、データセットの読み込み、CNN の構成などコアの部分のソースコードは、ほとんど cifar10 サンプルから流用した。

Tunnel 社のプロトタイプの認識機能を再現するため、その学習機と識別機のソースコードの一部を流用した。前述そのプロトタイプを勉強した際、筆者は既にそれにあるバグを修正し、TensorFlow の API バージョンの変化によるエラーも対応した。我々のプロジェクトは RoomClip の写真のみならず、ユーザーがアップロードした写真の認識も必要なので、識別機をローカルの画像ファイルを認識できるように修正し、不要なコードを削除した。

簡単にいうと、筆者が開発を担当した D-ROOM の学習機と識別機プログラムは、前述のサンプルプログラムとプロトタイプに修正と拡張を加えたものである。

完成したプログラムは、主に 4 つのモジュールがある。学習機と識別機共通の I/O モジュール、コアモジュールと学習機と識別機の本体で構成する。それ以外、筆者は自ら第 8 章で紹介する正解率判定のスクリプトを作成した。全てのモジュールは Python 言語で書いた。各モジュールの関連は、以下図 6.1 で示す。

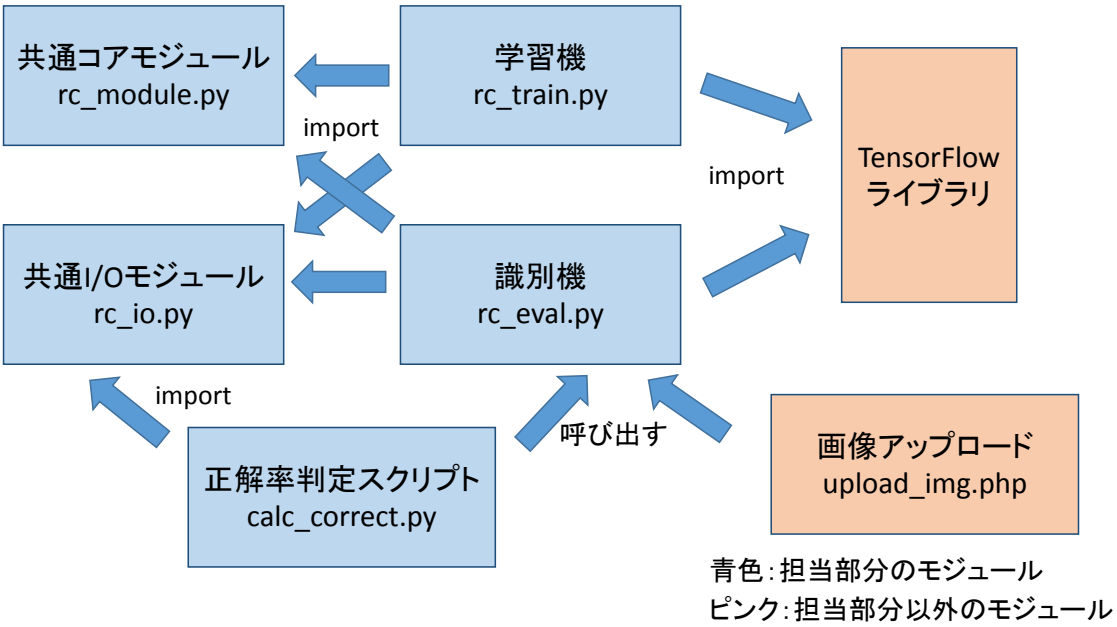

図 6.1　プログラムの各モジュール間の関連

前述の通り、TensorFlow フレームワークには、入力データの解析やディープラーニングに関する基本アルゴリズムなどローレベルなものが実装された。さらに、Python 言語のランタイムには、大量の便利なルーチンを含めている。そのため、プログラムの行数が比較的に少ない。最終版の正解率判定スクリプトを含めたモジュールの行数は合計 695 行であり、コメントや空白行を排除した場合は 454 行である。その中に、最も長い共通コアモジュール rc_module.py の行数は 227 行である。

これから、学習機と識別機の各モジュールの仕様について述べる。各モジュールにある機能に対応する関数や変数の表は、付録 B に参照できる。

6.3　ファイル I/O モジュール(rc_io)

このモジュールは、学習機と識別機、さらにそれらを使うツールが使う共通のモジュールの一つである。

基本となる全てのカテゴリ名をディスクから読み込む、教師データセットのカテゴリに対応する全画像のファイル名を読み込んでカテゴリと紐づけする機能は、このモジュールで実装される。

そして、学習機では、教師データにある一定の数の写真を周期的に読み込む必要がある。その読み込みの効率を向上するため、毎回ディスクから画像ファイルを一々読み込むより、最初から全ての学習用写真のデータを一つ大きな臨時ファイルに一時入れ込んで、必要になった時それからデータをメモリに読み出すという方法を採用した。TensorFlow は、TFRecord という大量の学習データを入れ込めるコンテナ形式をサポ

ートしているので、このモジュールの役割の一つは TFRecord の作成、と TFRecord から写真データの読み出しとなる。

それ以外、学習機と識別機の一部の共通設定(データのディレクトリ、分類の数、CNN ネットワークに送る画像のサイズなど)もこのモジュールで設定する。

6.4　共通コアモジュール(rc_module)

共通コアモジュールは、学習機と識別機両方のコアである。CNN による画像認識に必須の機能を実現したモジュールである。

このモジュールの機能を紹介する前に、まずは画像認識と分類の CNN によく使われる手順（層）について簡単に紹介する。本プロジェクトの目標は、ディープラーニングそのものの研究でないため、ここでの紹介は原理を述べる数学的ものではなく、筆者自身の理解によるもので、正しいとは限らない。

1）畳み込み層(Convolutional Layer)

図 6.2 で示すように、入力の画像データを行列 A とし、他に小さい行列 B を用意する。B を A の上に畳んで、B に含めた全ての数値を各自に覆った A に含めた数値と掛け算し、その掛け算結果全てを足し算する。B の位置を 1 数値ずれて再びその計算を行って、行列 A を完全に走査し、全ての積和の結果を行列 C にする。B はいつも完全に A と重なるので、出力の行列 C は必ず A より小さい。これは、画像処理によく使われる二次元の畳み込み演算である。この場合、行列 B は、フィルタやカーネルだと言われる。

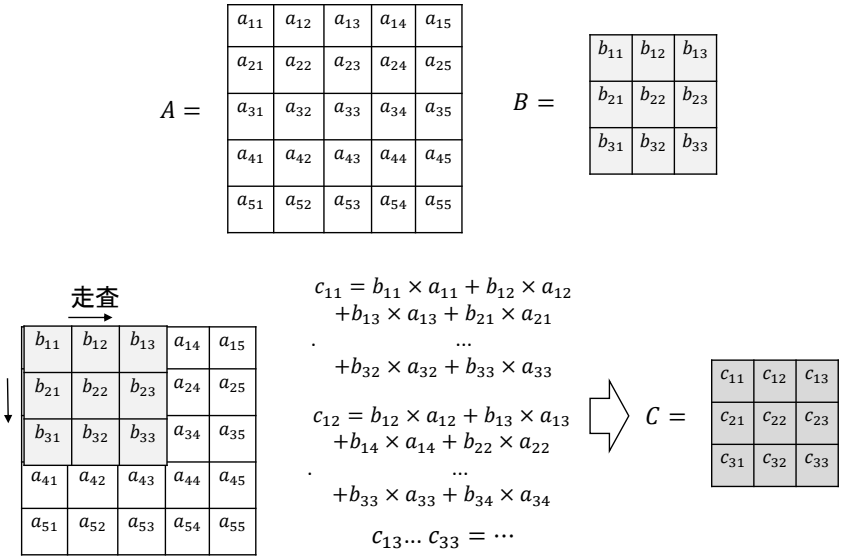

図 6.2　行列の畳み込み演算

CNN の畳み込み層は、この演算を応用した。筆者が実装した識別機の各畳み込み

層(conv 層)では、チュートリアルのソースコードと同じ、各畳み込み層のカーネルを TensorFlow の機能で、ガウス分布を従う値に初期化した 64 個の 5×5 の行列と設定した。

ディープラーニングの「ディープ」は「深い」と意味する。一枚の入力画像からできる限り多くの特徴量(feature)を得るため、1 個のフィルタではなく、数多く異なるものが使用された。

畳み込み運算のみでは、画像の特徴（ニューロンと言われることもあるが、ここでは特徴で呼称する）を抽出することができない。出力行列にさらなる処理が必要となる。

その一つは、バイアスの加算である。バイアスというのは、特徴量の傾向のようなものである。もう一つは、特徴量を活性化関数(Activation Function)で処理する。活性化関数は、動物の神経細胞が特定の信号に反応する確率を抽象化したものであり、ここでは計算量が小さく、学習の速度と最終結果が良い正規化線形関数(rectified linear function)[15]を使う。

畳み込み層で出力したものは、処理を完了した 64 個の特徴を含めた行列である。

2)プーリング層(Pooling Layer)

上記の畳み込み層が増える場合、最終的に特徴の数が膨大になる。プーリングというのは、簡単に言えば、畳み込み層が出力した特徴の行列の縮小である。実際の CNN では、プーリング層を畳み込み層や正規化層の直後に入れることが多い。それにより、計算量が小さくなる以外、入力の微小な変化が特徴の最終的な出力への影響も小さくなる。筆者の CNN では、最大プーリング(Max Pooling)という、特徴行列に有力な値をできる限り残せる方法を使用した。

3)正規化層(Normalization Layer)

正規化は、特徴を画像としてコントラストを一致するというプロセスである。このプロセスは、認識の結果に良い影響があると考えられる。A. Krizhevsky ら[11]によると、局所コントラスト正規化(Local Response Normalization)の運用により、認識の不正解率が少し(1.4%)減少した。

4)全結合層(Fully-connected layer)

全結合層は、畳み込み層とほぼ同じであるが、生成した全ての特徴が繋がるという仕組みがある。全結合層にある各特徴も、活性化関数に通すことができる。

最終的に入力データから得られた特徴を各カテゴリの確信度を付く、つまり分類するのは、最後の全結合層で行った線形の活性化関数である。

筆者の CNN ネットワークは、前述のサンプルプログラムのものを流用した。その構成は以下の図 6.3 で示す。畳み込み層(conv1, conv2)、プーリング層(pool1, pool2)、正規化層(norm1, norm2)、最後に全結合層(local3, local4, softmax_linear)で、合計 9 つ

の層がある。

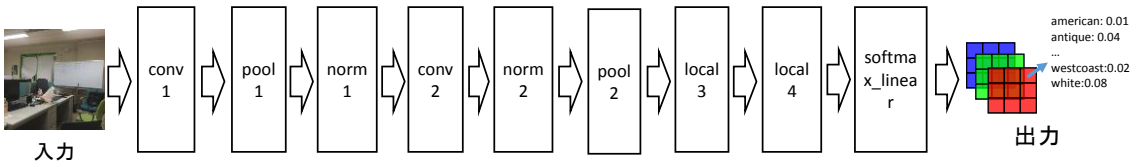

図 6.3　共通コアモジュールで実現した CNN の構成

この構成は、当初にそれの一部を変更する実験を行いたくなったが、毎回の変更では再学習が必要である。CPU のみの学習が遅すぎるので、この実験は行わなかった。

10 月下旬に GPU が使える学習専用機の導入後、精度向上のため、CNN ネットワークの構成を改変する実験を行った。具体の実験内容と精度向上の効果は後ほど 8.7 節で紹介する。

CNN の構成以外に、学習のパラメータの指定、全体の学習誤差の計算、学習に使う画像とカテゴリのバッチの生成、教師データの予備処理など学習に必要な機能、さらに TensorFlow のログファイルを出力する機能もこのモジュールで実現した。

6.5　学習機モジュール(rc_train)

「学習」という言葉は本書に複数回に言及したが、ここでその意味を解釈する。

前の節で簡単に説明した CNN を使えば、画像データを入力し、最後に全結合で CNN から得られた全ての特徴を分類できる。だが、言うまでもなく、重みなど特徴のパラメータが初期値のままであれば、分類の正解が出ないのだろう。いわゆる「教師データ」は、入力の画像、とその画像の分類(画像が所属するカテゴリ)の正解のペアで組み合わせたものである。教師データの画像を CNN に通し、その分類の解と正解の間の誤差を計算し、勾配降下法[16]でその誤差に基づいて、CNN のモデルにある全ての特徴の重みを正解の方向に修正する。この操作を繰り返し、最終的に教師データに対する訓練誤差のみならず、教師データが所属する母集合のデータに対する全体の認識誤差も下がる。これは、機械学習の領域に「学習」と言われるものである。

最も簡単な学習の方法は、1 回に学習機に教師データの 1 サンプルずつ、すなわち画像を 1 枚ずつ通す。だが、これは現在の主流になったマルチコア、マルチスリットの環境を充分に利用できない。そのため、筆者の学習機では、教師データを前述の「バッチ」に分けて、多数の画像を同時に CNN に入力する。TensorFlow では、この形式の入力を自動的に CPU や GPU を使って並列化できる。筆者はこのバッチのサイズのデフォルト値を 128 サンプルに設定した。

さらに、TensorFlow はセッションという概念がある。ユーザーの Python プログラムでセッションに必要な操作を定義して、セッションを初期化すれば、後は設定が要らなく、セッションを実行するだけで良い。

機械学習の領域には、一つの状況を考えなければならない。サンプルの数が限られた教師データを繰り返して学習すると、訓練誤差がほとんど変わらなくなったが、学習してきた CNN が母集合の認識での全体誤差が上がる、すなわち過適合(Overfitting)の状況が発生しやすい。チュートリアルでは、教師データの画像をランダムに切り取るなど措置でサンプルの数を水増し、学習の回数（今後「ステップ数」と呼称）により、学習によって重みの変化率(learning rate)を徐々に下げるなど措置を使用したが、筆者のバージョンはそれを一部のみ実装した。

学習機モジュールは Python で直接実行できるプログラムであり、コマンドラインの引数で最大学習ステップ数を設定できる。

学習を実行する train()の仕組みは、次のフローチャート、図 6.4 で示す。重要なポイントは、1000 ステップごとに、セッションの中に学習してきた CNN の全ての特徴パラメータを含めた「モデル」をディスクに保存することである。保存した「チェックポイント」ファイルは、後で識別機の実行に必要である。

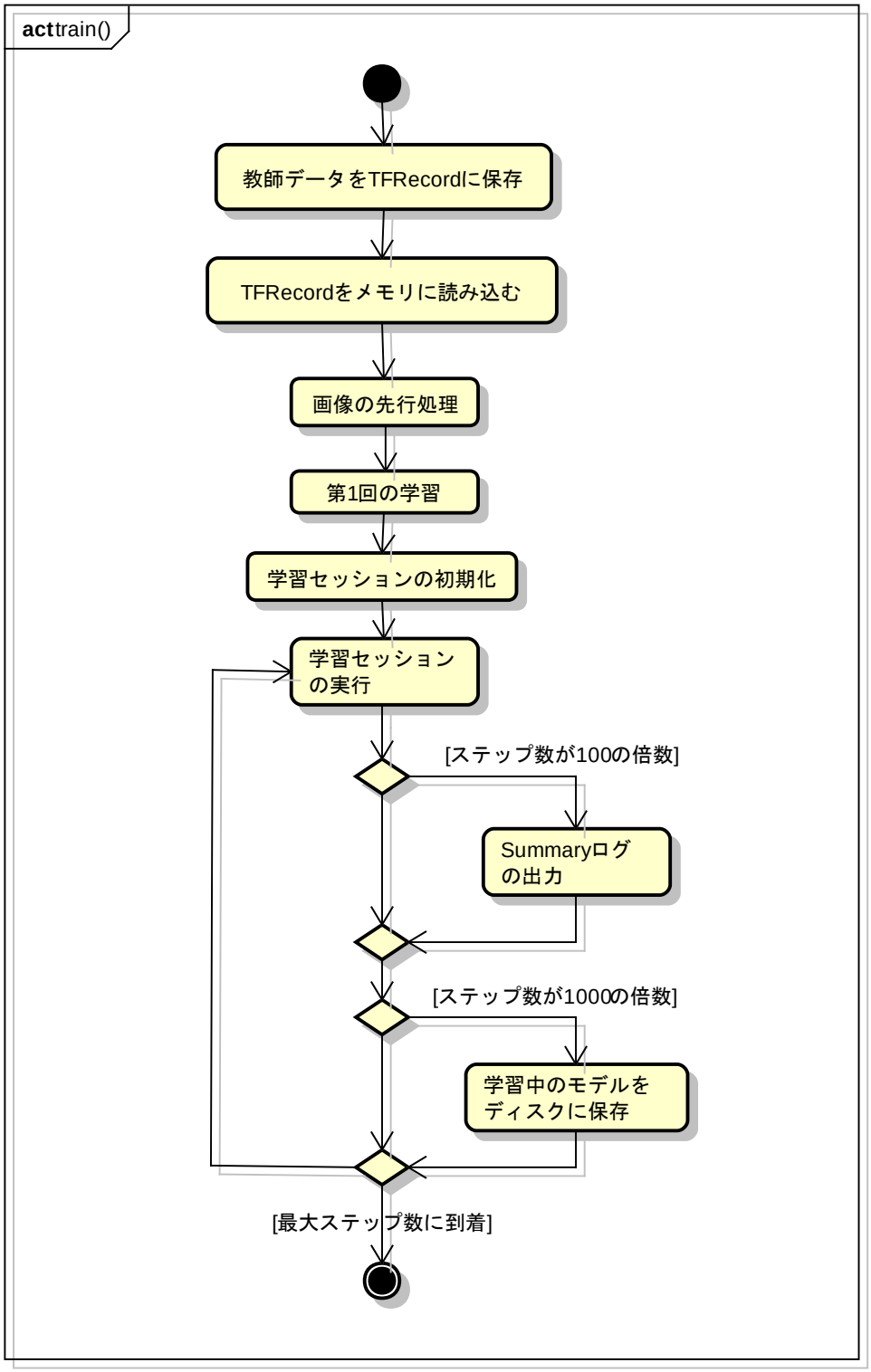

図 6.4　学習機のフローチャート

6.6　識別機モジュール(rc_eval)

サービス班との討論により、筆者が担当する識別機は、サーバーのローカルファイルシステムにある写真ファイルを入力し、その分類結果を JSON 形式で出力する機能が要求された。

筆者は、Tunnel 社のプロトタイプの識別機モジュールに基づいて、Web アプリケーションで簡単に使える API として、D-ROOM で用いられる識別機を実装した。

実行する際、ローカルにある JPEG ファイルをコマンドラインで指定し、認識の結果を JSON で出力する。必要な権限は、画像ファイル、チェックポイントのファイル、とカテゴリ名一覧のディレクトリの読み込み権限のみで、Web サーバーのアカウントからも実行できる。

認識の流れは、前述の学習と似ているが、CNN に入力する画像が一枚のみで、CNN からの出力は誤差を分析せず、直接 Softmax 関数で分類結果を各分類の確信度に変換し、出力する。その出力した JSON は、API の返り値としてフロントエンド班の Web アプリケーションに使われる。

識別機の実行結果の例は以下の図 6.5 で示す。

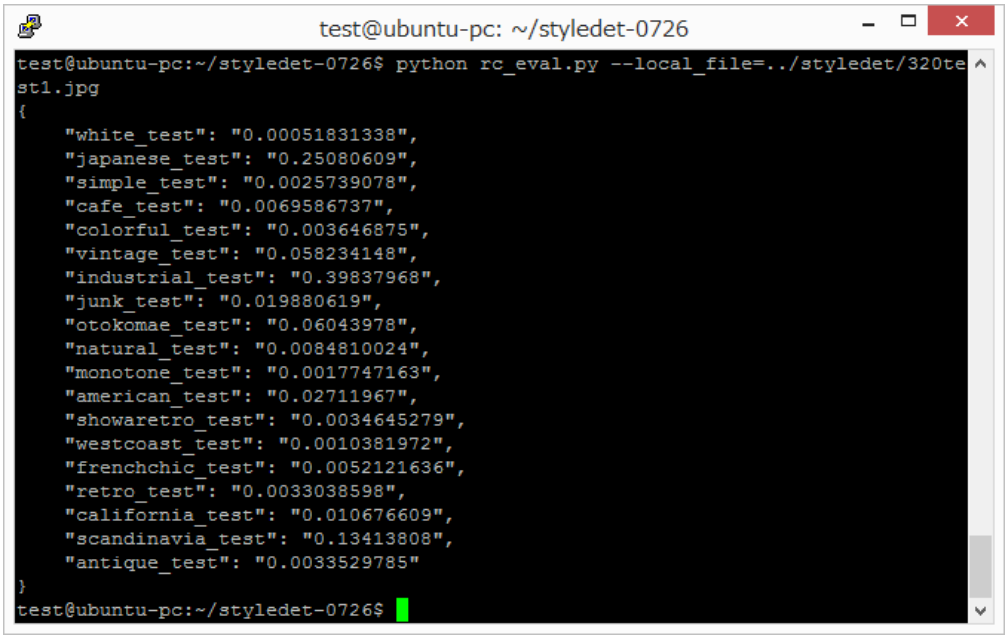

図 6.5　識別機の実行結果のスクリーンショット

識別の流れは、図 6.6 のフローチャートで説明する。

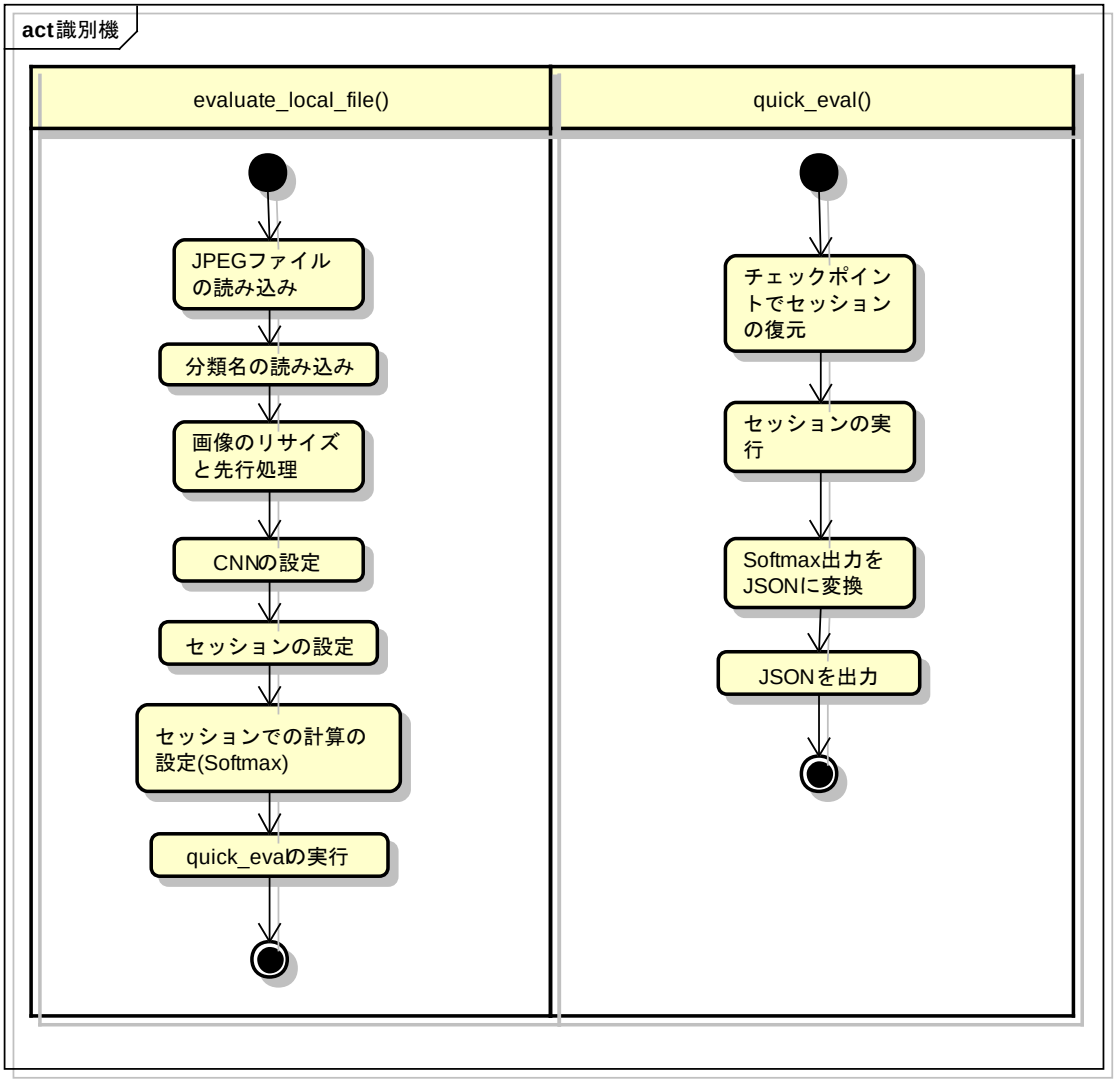

図 6.6　識別機の流れのフローチャート

第 7 章　分類器作成と画像認識の実験

7.1　教師データセット

本書に何度も言及した「認識」や「分類」とは、入力した写真がどのカテゴリに属することを判定することである。だが、そのカテゴリがどのようなものは、ここまで述べていない。本節は、識別機の出力となる写真のカテゴリと、それに対応する教師データセットについて述べる。

1.3 節にも述べたが、Tunnel 社の要望により、学習に使用されるインテリアの写真は全て RoomClip にアップロードされたものから取るようになった。我々は Tunnel 社の平山氏から、RoomClip のデータベースの一部を頂いた。その内容には、RoomClip にアップロードされた全ての写真のファイル名、RoomClip に登録した全てのタグ、写真とタグの関連など重要な情報がある。

我々はそのデータを用いて、特定のタグが付いた全ての画像のファイル名を検索できる。写真のファイル名に重複がないので、ファイル名とファイルそのものとの関係は 1 対 1 である。さらに、我々は Tunnel 社から、2016 年 6 月まで RoomClip にアップロードされた全ての写真をダウンロードする権限を頂き、研究室のサーバーにそれを全てダウンロードしたので、タグを使って写真のデータを選ぶことができる。

チーム全員は、これまでインテリアの専門的な知識が全くなかった。そのため、筆者はインテリアの写真をカテゴリ化する時、RoomClip のユーザーが追加したタグを根拠として使うようになった。

6.1 節に述べた Tunnel 社のプロトタイプで使ったカテゴリを参照しようとしたが、ドキュメントなどがなく、データセットに記載された英語のカテゴリ名のみがあった。そのカテゴリ名の中の二つ（mens と jumble）は、筆者が日本語に正確に翻訳できなかった。そのため、実際に使ったカテゴリはプロトタイプに使ったものと異なり、筆者自身が決めた 19 種類であった。

この 19 種類のカテゴリの教師データセットを作る方法は、前述の通り、カテゴリと RoomClip に登録したタグとの対応関係を立て、RoomClip が提供したデータベースからそのタグが付いた写真を全て検索し、その中からカテゴリごと 1000 枚の写真のファイル名をランダムに選ぶ。さらに、RoomClip の全ての写真ファイルから、選んだファイル名の写真の JPEG ファイルを、教師データセットの各カテゴリに対応するディレクトリにコピーする。Tunnel 社のプロトタイプに参考して、その JPEG ファイルのサイズは 320×320 と設定した。

以下の図 7.1 は、教師データセットのディレクトリ構成を示す。1 カテゴリに 1 ディレクトリがあり、カテゴリに属する画像のファイル名はそれぞれテキストファイルに格納する。

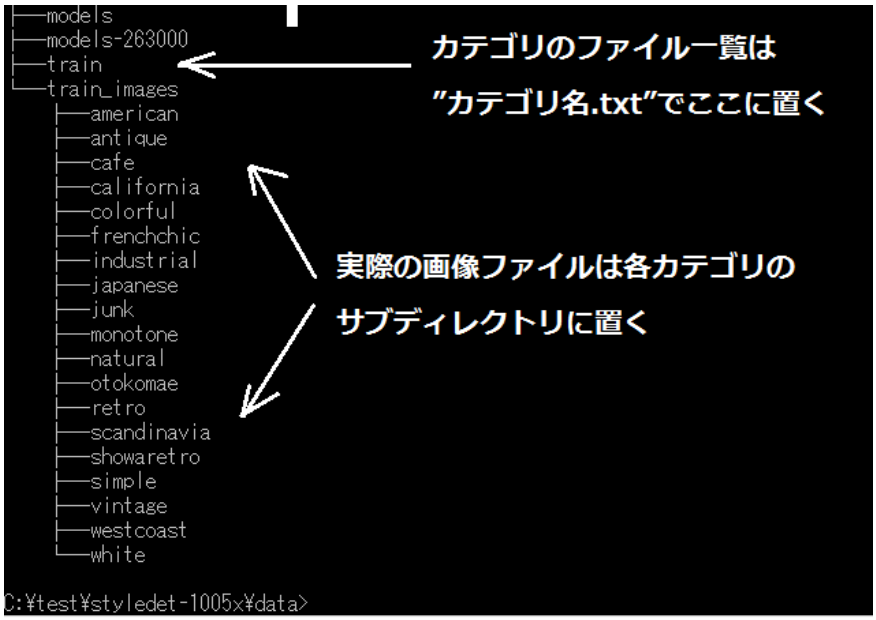

図 7.1　教師データセットのディレクトリ構成

カテゴリとタグの対応関係は、最初は 1 対 1 であった。以下の表 7.1 で、その対応関係を示す。

表 7.1　カテゴリとタグの対応関係（7 月版教師データ）

カテゴリ	対応のタグ	カテゴリ	対応のタグ
american	アメリカン	natural	ナチュラル
antique	アンティーク	otokomae	男前
cafe	カフェ	retro	レトロ
california	カリフォルニア	showaretro	昭和レトロ
colorful	カラフル	scandinavia	北欧
frenchchic	フレンチシック	simple	シンプル
industrial	インダストリアル	vintage	ビンテージ
japanese	和風	westcoast	西海岸
junk	ジャンク	white	ホワイト
monotone	モノトーン		

この方法で作った教師データセット(以下「7 月版」と呼称)は、プロジェクト第 1 段階の分類器の作成に使われた。ただし、プロジェクトの第 1 段階が終わった時、サービス班が行った分類器のテストで、その分類の正確さが疑われたので、その後、認識精度向上の実験で、教師データの選別方法には数度の見直しが行われた。詳しくは次の章で述べる。

付録 A では、写真のファイル名を検索する SQL スクリプトが示される。

7.2　第 1 版分類器の学習

第 3 章の末に述べた通り、本プロジェクトの第 1 段階の目的は、ディープラーニングの基礎知識の勉強と Tunnel 社のプロトタイプの再現となった。そのため、精度や速度など問題を実際に考える前、とりあえず教師データセットを学習して一つの分類器を作る必要があった。

TensorFlow のチュートリアルでの CNN へ入力する画像のサイズは 24×24、Tunnel 社のプロトタイプでは 40×40 であった。筆者は当時、学習機に入力する画像をより大きなサイズにすると、認識の精度は上げられると考えた。そのため、7 月版のデータセットを学習する時、学習機と識別機に入力する画像を 100×100 と指定したが、その時点では、精度が向上できる根拠はなかった。学習に使われる他のパラメータ（バッチのサイズなど）は、デフォルト値を使用した。

本プロジェクトでは、CNN へ入力する画像のサイズと教師データの画像のサイズとは異なる。教師データは RoomClip のサーバーに保存した写真ファイルで、解像度が最低でも 320×320 である。CNN へ入力する画像のサイズが大きくなると、ステップごとの計算量が大幅に増えるので、筆者は RoomClip の写真ファイルをそのまま使わなく、教師データの予備処理の段階で写真を特定のサイズ（I/O モジュールのグローバル変数 fixed_image_size）に縮小するように実装した。

その時点で、学習に使われたのは研究室にある開発用サーバーであった。そのサーバーは CUDA をサポートする GPU が装着されていないため、学習は CPU で行った。第 5 章最後で述べた通り、CPU の型番は Intel の Core i7-3770K で、4 つの物理コアを持つ i7 シリーズのハイエンドタイプである。それにもかかわらず、Tunnel 社のプロトタイプと似ていて 280000 ステップまで学習するには、16 日 20 時間がかかった。この学習で生成した分類器は、本書では「第 1 版」と呼称する。

6.5 節の学習機のフローチャートで示した通り、学習機は学習の途中に 100 ステップ 1 回の頻度で TensowFlow のクラス SummaryWriter を使い、各手順で出力した数値などを summaries ファイルに保存する。その summaries ファイルは、TensorFlow の Web ツール TensorBoard[1]でブラウザー上に可視化できる。

筆者は、以下図 7.2 で示した第 1 回の学習の全体誤差の曲線を観察した。

[1]https://www.tensorflow.org/versions/r0.11/how_tos/summaries_and_tensorboard/index.html

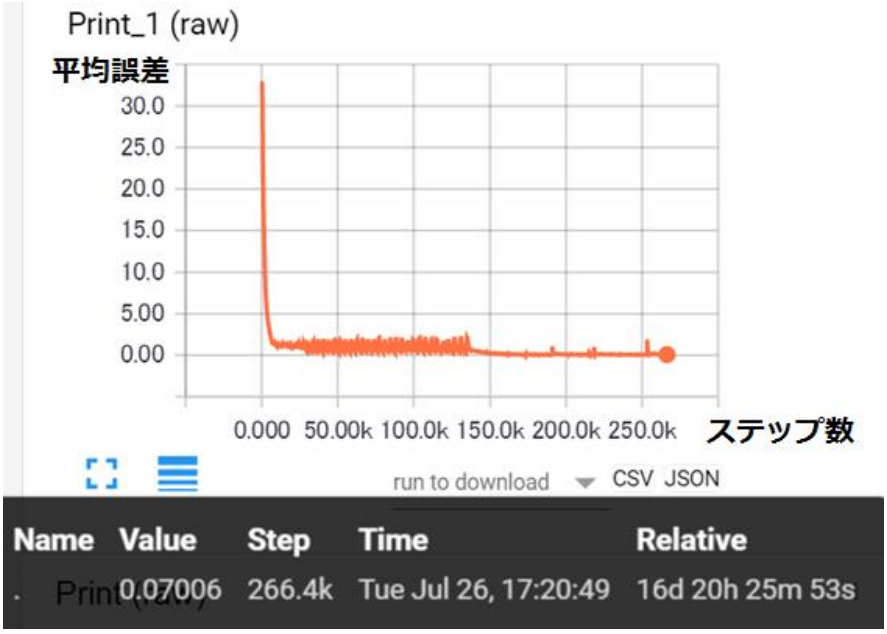

図 7.2　第 1 回の学習で生成した全体の誤差の曲線

この図から見ると、130k ステップまで誤差の値が激しく上下したが、130～160k 位から下落し、最後に 0.05 位まで達した。その現象の原因は分からなかったが、TensorFlow のチュートリアルで見えた現象[13]と似ていた。

7.3　第 1 版分類器を使用した画像認識

前述の通り、本プロジェクト第 1 段階の目標は、Tunnel 社のプロトタイプを再現し、任意の画像を認識できる簡単なシステムを作ることになった。前述の第 1 回学習で生成した分類器と識別機を、当時に簡単なフロントエンドを作成した ZHANG 君に提供し、アップロードされた画像を認識することができた。

図 7.3 で示すように、このフロントエンドは、前述の図 6.5 で示した識別機の出力をユーザーフレンドリーな形式で示すものである。

図 7.3　プロジェクト第 1 段階の成果物のスクリーンショット

だが、この段階で使用した分類器と識別機には、問題が少なくない。まず、当時認識精度を測る方法がまだ確立していなかったが、チームメンバーの主観的な評価は良くなかった。そして、分類器の生成は 2 週間以上かかるので、精度向上の実験に必須となる学習の回数が限られる。さらに、一回の識別の所要時間が 10 秒ぐらいで、実際の運用では遅すぎる。

第 2 段階でプロジェクトの目標を決定したところ、筆者は既に認識精度向上の実験に着手した。最終目標であるサービスを作るため、認識の精度と速度の向上という任務は筆者が所属するバックエンド班に分担するようになった。色々試行錯誤を行った末、筆者が最終的にサービスに使われる分類器を作成した。次の章は、この認識精度向上の実験を詳述する。

第 8 章　認識精度向上の実験

プロジェクトの第 1 段階は、7 月末に問題なく完了した。といったが、その段階の目的は、とりあえず Tunnel 社のプロトタイプの再現なので、認識精度の計測は行わなかった。更に、サービス班が識別機のテストを行ったところ、分類の結果が良くないというフィードバックがあった。第 2~3 段階に筆者が主に担当したのは、識別機の評価に当たる認識精度の測定と、精度向上の実験での試行錯誤である。

8.1　認識精度の定義と評価方法

第 2 段階以降に筆者が所属したバックエンド班の担当内容は、サービス班に提供する識別機の認識精度と速度の向上であると決まった。そのため、各バージョンの分類器の認識精度を測る必要があった。

先行研究では、分類器の精度を評価する際、評価用データセットでの平均誤差を使った。筆者の場合は、その平均誤差が非常に高いという可能性は十分あると考えた。

RoomClip にアップロードされた写真には、デリゲートなものが非常に多い。第 2 章にも言及したが、白い犬が写真に入っただけでその写真を「ホワイト」というスタイルを表すタグを付くなど状況が多い。そして、スタイルを正しく表す画像としても、部屋全体の画像とオブジェクトの画像など、全く雰囲気が異なるものが混在する。さらに、写真の内容がそれぞれ複雑で、特徴が人目でも取りづらいのに、そのスタイルが同じだというケースが非常に多い。

先行研究でよく言及した、TensorFlow のチュートリアルにも使われた CIFAR-10[1]など画像分類によく使われたデータセットは、カテゴリそれぞれに明確な定義がある。例えば、「cat」のカテゴリは猫の写真、「truck」のカテゴリはトラックの写真である。それと全く異なって、RoomClip の写真に追加したタグが表すスタイルは、このような明確の定義がなく、抽象的なもので、人の目でも分からないこともある。

そのため、誤差の低さではなく、その正解率の高さで認識の精度を評価したいと考えた。正解率が高いほど、認識の精度が高い。

筆者は、評価用データセットにある写真ファイルを認識し、その分類結果に確信度が一番高いカテゴリがその写真ファイルに属するカテゴリ、すなわち予想の分類結果に合致すれば、正解とする。そして、各カテゴリに対する正解の回数と写真ファイルの数を記録し、最終的に評価用データセット全体の平均正解率を計算できる。

---

[1] http://www.cs.toronto.edu/~kriz/cifar.html

8.2　評価用データセット

正解率を計測するため、評価用データセットを作らなければならない。Tunnel 社のプロトタイプには評価用データセットがあるので、作り方はそちらに参考した。

Tunnel 社のプロトタイプには、二つのデータセット、教師データセットと評価用データセットが既に含めている。その二つを比較すると、教師データと評価用データの間に、重複したファイルがない。第 6 章にも述べたが、ディープラーニングでの学習の本質は CNN のパラメータを教師データに適合する過程である。そのため、学習が終わった CNN でその教師データの写真を分類すると、正解になるのは当然であろう。ゆえに、評価用データに学習に使用したものを排除しないと、平均正解率に影響がある。

そして、分類の正解率の公平性を保つため、教師データと評価用データのカテゴリの分け方も同じにしなければならない。最も簡単な評価データを選ぶ方法は、図 8.1 で示したように、同じカテゴリの写真データの集合から、二つの共通部分がない真部分集合を選び、その一つを教師データにし、もう一つを評価用データにする。

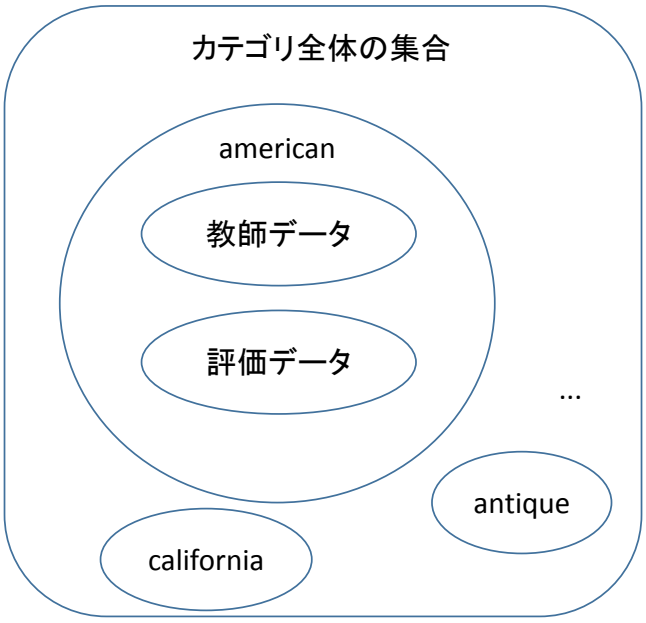

図 8.1　教師データと評価データの関係

正解率を計測するため、筆者は前述の方法で、第 7 章に紹介した 7 月版の教師データセット、後述のチューニングで作った新しい教師データセットそれぞれに対応する評価用データセットを作った。それらのデータセットには、19 個のカテゴリそれぞれに 130~1000 枚の写真がある。

なぜ教師データセットと同じくカテゴリそれぞれ 1000 枚ぐらいにしなかったのは、一部のカテゴリ(特に frenchchic)にある画像は少ない（1500 枚以下）ので、筆者の方法では二つの部分集合を同じサイズにするのは不可能だということである。

8.3　正解率計測スクリプト

評価用データセットの一括判定を自動化するため、筆者は Python で正解率計測のスクリプトを作成した。

その仕組みは、評価用データセットの指定したカテゴリにある全ての写真ファイルの一覧を作り、それを一枚一枚識別機に通し、出力された結果の JSON から確信度が一番高いカテゴリを検索し、それと正解のカテゴリ（予想のカテゴリ）の名前と比較して正解を判定する。そして、全てのファイルに対する正解の回数とカテゴリに対する正解率を計算する。その流れは、次の図 8.2 のフローチャートで説明する。

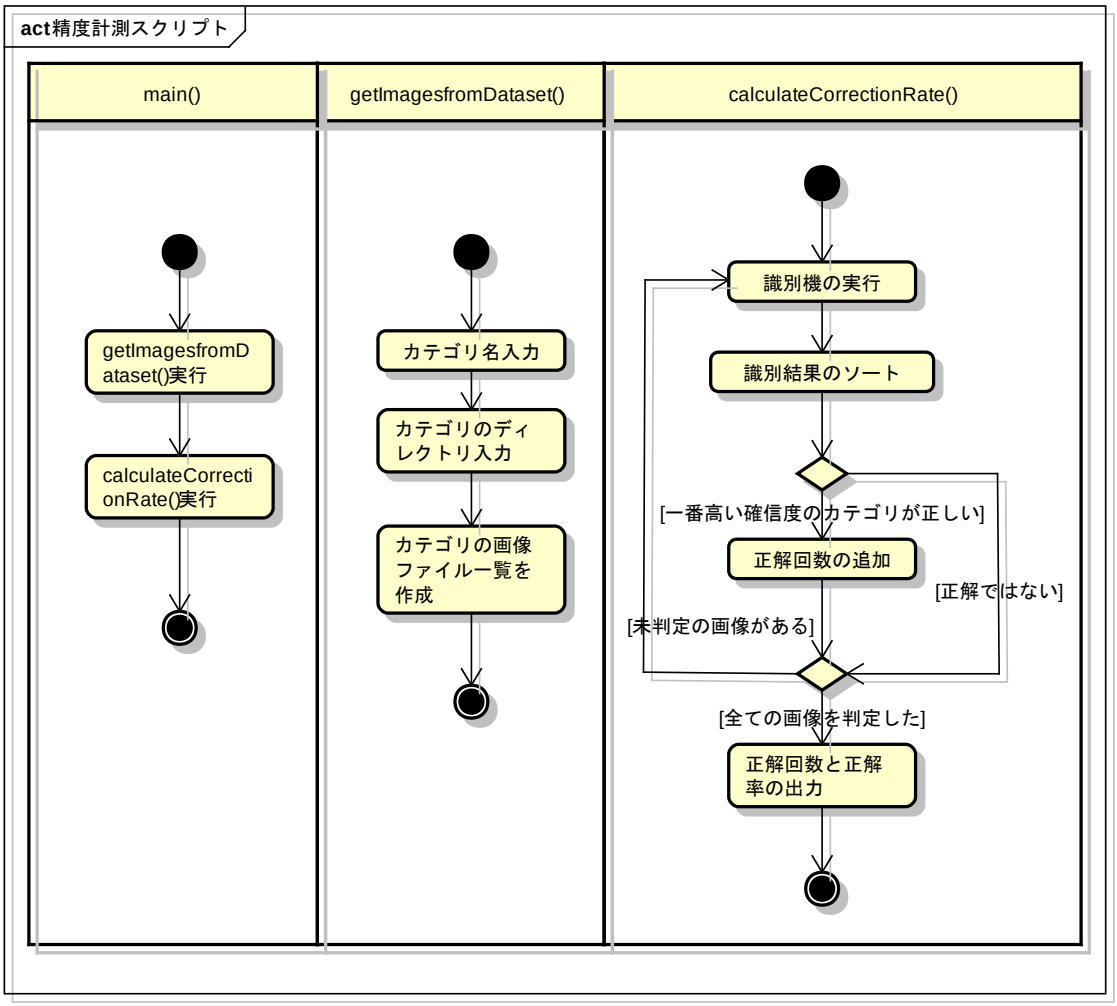

図 8.2　正解率計測スクリプトのフローチャート

このフローチャートには書いてないが、必要の場合は毎回の分類結果と正解をログに全て出力することもできる。

データセット全体の正解率を計算するため、全てのカテゴリに対し、このスクリプトを一々実行する必要がある。筆者は、Shell スクリプトで 19 カテゴリの連続計測を実

現した。正解率計測スクリプトの使用方法と Shell スクリプトは、本書付録 D でさらに紹介する。

正解率計測スクリプトの準備により、認識精度向上の実験の条件が備えた。

8.4　カテゴリと教師データセットのチューニング

本章の初頭にも述べた通り、プロジェクトの第 2 段階に入ると、第 1 段階で作成した 7 月版のデータセットと、それを学習して生成した第 1 版分類器に色々問題点が発見された。そのため、最終成果物の実装まで、筆者の目標はチューニングにより認識精度の向上と決定された。

本節で述べるチューニングは、写真ファイルのカテゴリ化方法の調整と選んだ教師データセットに一部の写真の手動選別によるものである。

まず、筆者は 7 月版の教師データセットの内容をチェックした。一部のカテゴリの間に、重複した画像があるという問題が発見された。そのため、筆者は Tunnel 社から頂いた RoomClip のデータベースから写真を検索する方法を変えて、カテゴリに対応するタグが一つだけ付けられる写真を選ぶようにした。この方法で、「9.22 版」の教師データセットを作った。

だが、このような方法でカテゴリの写真を選ぶと、一部のカテゴリの画像の数が大幅に減少した。筆者はチームメンバーとの討論により、ここまで使ったカテゴリに対応する一部のタグは、RoomClip には主流ではなく、それと意味が似ている他のタグの方がよく使われると分かった。

カテゴリとタグの対応関係を 1 対 1 から 1 対多にすれば、すなわち、同じカテゴリに複数の意味が近いタグを含めば、この状況に対応できると考えた。この新しいカテゴリ化の方法で、「10.5 版」の教師データセットとそれに対応する評価用データセットを作った。

図 8.3 は、最初のカテゴリ化方法とそれに改善を加えたものとの区別を示す。

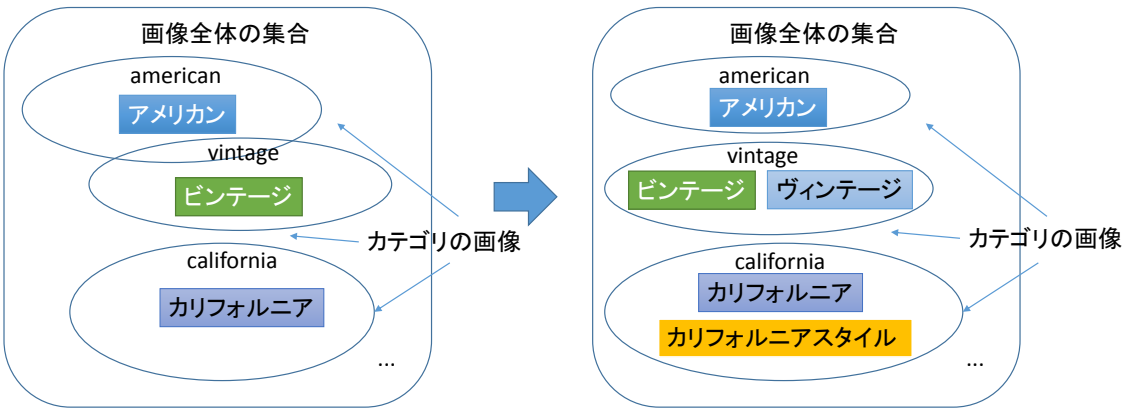

図 8.3　カテゴリの画像の選択方法の改善策

後述の正解率計算では、10.5 版のカテゴリ化方法での正解率の向上が限られると分かった。原因を分析すると、その一つは、「japanese」のカテゴリの中には、「和室」というタグが含めたと考える。「和室」はインテリアのスタイルではなく、部屋のタイプを表す言葉である。「和室」の写真であっても、「和風」のスタイルでないことが多い。そのような写真を含めた場合、教師データの質に悪影響が出る可能性がある。もう一つは、どの方法でデータベースから写真を検索しても、他の写真と明らかに雰囲気が異なる代表的でない写真が出る。

この問題の解決策として、まずはタグ「和室」を「和風」と意味がより近い他のタグに変える。さらに、データベースからランダムに選んだ 1000 枚の写真にある不適切なものを手動で排除する。筆者はチームメンバーの張墾君の協力をもらって、最終版の「11.3 版」教師データセットを作った。

表 8.1 は最終版のカテゴリとタグの関連を示す。

表 8.1　カテゴリとタグの対応関係（最終版）

カテゴリ	日本語名	対応のタグ
american	アメリカン	アメリカン
antique	アンティーク	アンティーク
cafe	カフェ	カフェ, カフェ風
california	カリフォルニア	カリフォルニア, カリフォルニアスタイル
colorful	カラフル	カラフル
frenchchic	フレンチシック	フレンチシック
industrial	インダストリアル	インダストリアル
japanese	和風	和風,和,和モダン
junk	ジャンク	ジャンク
monotone	モノトーン	モノトーン
natural	ナチュラル	ナチュラル
otokomae	男前	男前
retro	レトロ	レトロ
showaretro	昭和レトロ	昭和レトロ
scandinavia	北欧	北欧
simple	シンプル	シンプル
vintage	ビンテージ	ビンテージ, ヴィンテージ
westcoast	西海岸	西海岸, 西海岸インテリア
white	ホワイト	ホワイト, ホワイトインテリア

付録 A では、RoomClip データベースから写真のファイル名を検索する SQL スクリプトの最終版が示される。

8.5　新しい分類器の学習

教師データセットのチューニングの効果を評価するため、学習機でそれを学習させて新しい分類器を生成する必要がある。

9.22 版と 10.5 版教師データセットが作られた時点、チームはまだ GPU が装着した学習専用機がなく、研究室の開発用サーバーの CPU で学習を行った。7 月で学習した分類器の第 1 版と同じ、この二つデータセットで分類器の第 2、3 版の学習は長い時間がかかった。

10 月中旬から、ハードウェア系を担当した藤田君は研究室の一台の PC に nVidia の最新の GPU、GTX 1070 を装着し、学習専用機として使えるようになった。その学習速度の向上で、試行錯誤ができる回数が増え、7.2 節で述べた CNN に入力する画像のサイズを 100×100 にする効果も、プロトタイプと同じ 40×40 で分類器の生成により、検証できるようになった。

表 8.2 は、ここまで同じ学習機で生成した各バージョンの分類器のパラメータと学習の所要時間を示す。

表 8.2　主要の分類器データの学習の所要時間

Ver.	教師データ	入力サイズ	ステップ数	GPU	所要時間
1	7 月版	100×100	266,000	無	16 日 20 時間 26 分
2	9.22 版	100×100	229,000	無	15 日 18 時間 25 分
3	10.5 版	100×100	190,000	無	12 日 1 時間 12 分
4	10.5 版	100×100	473,000	有	1 日 21 時間 38 分
5	10.5 版	40×40	283,000	有	11 時間 21 分
6	10.5 版	40×40	689,000	有	1 日 47 分
7	11.3 版	40×40	300,000	有	21 時間 27 分
8	11.3 版	40×40	500,000	有	2 日 3 時間 32 分

11.3 版の教師データセットを学習したところ、学習機は他のチームメンバーが開発に使われたので、第 7、8 バージョンの学習の速度（時間内の学習ステップ数）が 10.5 版より遅くなった。

それらの学習で、7.2 節にも示した全体の誤差の曲線が突然大幅に下落する現象が見えた。特に 40×40 の場合は、初期の全体誤差が比較的に低いので、この全体誤差のカーブが鮮明になった。

図 8.4 はバージョン 5 の分類器の学習で記録した全体誤差の曲線である。約 13 万ステップのところから、その前に 1.00 位に浮動していた誤差の値が一気に 0.1 以下に減少した。

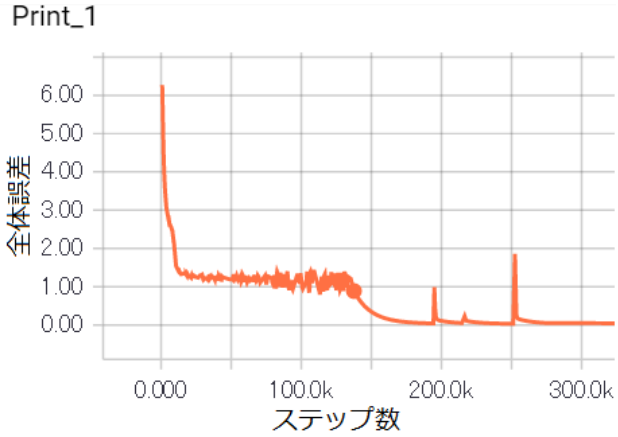

図 8.4　バージョン 5 の分類器の学習で生成した誤差の曲線

その一方、表 8.2 で示すように、学習機は同じであるが、教師データ以外に入力画像サイズとステップ数という二つの変数がある。もともと筆者は正解率に影響がある 3 つの変数を独立にしてそれぞれに対応する分類器を作り、正解率の計測実験を行いたいが、時間が限られたので、そのような厳密な実験はできなかった。

8.6　正解率計測の結果

筆者は、Tunnel 社のプロトタイプにある分類器、そして 8.5 節で述べたバージョン 1、4、5、7 と 8 の分類器に①、②、③、④、⑤、⑥の番号を付け、教師データに対応する評価用データを使い、分類の平均正解率を計測した。前述の通り、時間の関係で、精度に影響ある要素を独立変数にした厳密な実験ではなかったが、その結果は示せる。

6 組の実験用分類器の特徴は以下の表 8.3 で示す。

表 8.3　正解率計測に使用する 6 組の分類器

番号	教師データ	ステップ	CNN の入力画像サイズ
①	Tunnel 社プロトタイプ	281,000	40×40
②	7 月版	266,000	100×100
③	10.5 版	473,000	100×100
④	10.5 版	283,000	40×40
⑤	11.3 版	300,000	40×40
⑥	11.3 版	500,000	40×40

正解率の基準を設けるため、我々の学習機で生成した分類器以外、6.1 節で述べた Tunnel 社のプロトタイプに含めたサンプルの分類器（分類器①）の正解率も計算した。評価用データは、プロトタイプに含めたものを使用する。CNN の構成が同じため、我々の識別機は、プロトタイプの分類器データを直接に使うことができる。ただし、プロトタイプに使ったカテゴリの数が違うので、識別機のパラメータの調整が行われた。

正解率計測の結果は、以下図 8.5 で示す。

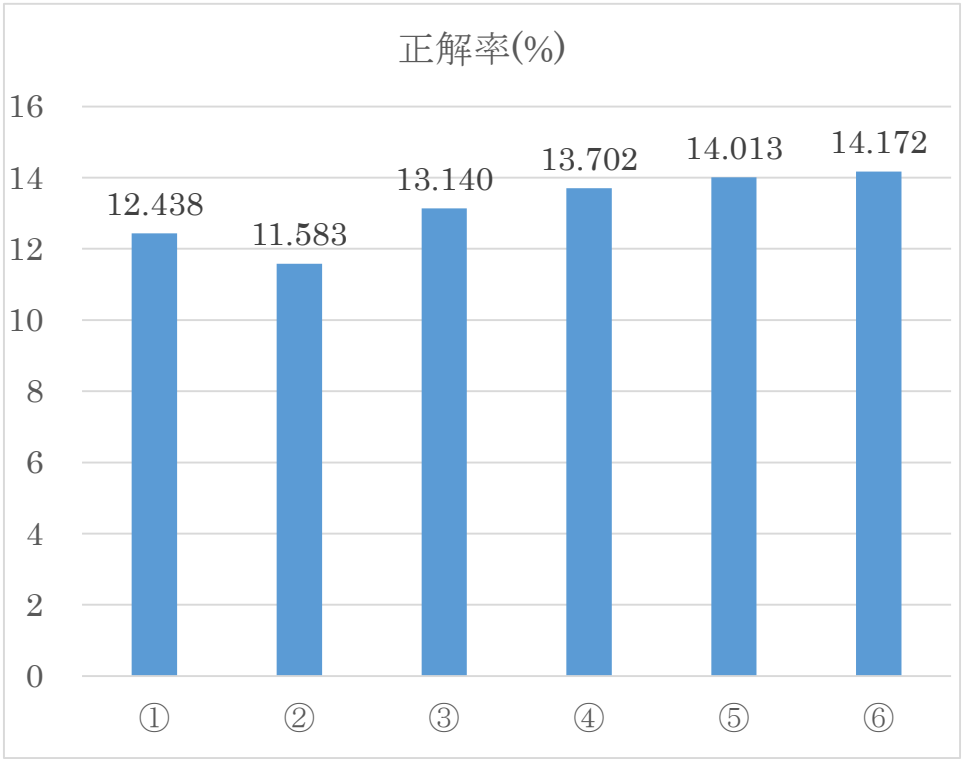

図 8.5　6 組の分類器の平均正解率

6 組の分類器のどちらも、正解率の値が低い。その原因は、第 1 節で述べた通りに、カテゴリ化する根拠が RoomClip のユーザーが付けたスタイルのタグであって、そのスタイルが抽象的なので定量化やモデル化が難しいと推測する。

Tunnel 社プロトタイプの分類器①を基準にすれば、我々の分類器②は正解率が基準より低いが、③～⑥の正解率が基準より高い。これは、教師データセットのチューニングが有効であると説明した。

入力画像のサイズが同じ、ステップ数も近い④と⑤は、明らかに⑤の方の正解率が高い。これは、第 4 節に述べたカテゴリ「japanese」に含めるタグの改変と不適切な画像の排除が有効であると説明した。

同じ教師データで生成され、入力画像のサイズも同じ⑤と⑥は、ステップ数が多い⑥の方は正解率が高い。ただし、その差が 0.16%で、影響が大きいとは言えない。

同じ教師データで生成された③と④は、③の方は学習ステップ数が多いが、入力画像のサイズが小さい分類器④の正解率が高い。これは 7.2 節に述べた「入力する画像をより大きなサイズにすると、認識の精度は上がる」という考えを否定した。

以上で分類器の間の平均正解率を比較した。各カテゴリの精度に関するのは、以下の混同行列(confusion matrix)で説明する。旧バージョンの分類器(①～④)の正解率を計算する時、詳しいログは記録されてなかったので、ここで示すのは、分類器⑥の正解率計測のログにより生成した混同行列である。

	americ	antiq	cafe	califo	colorf	frenc	indus	japan	junk	mono	natura	otoko	retro	scand	show	simpl	vinta	west	white			正解率(%)	変動係数
american	139	37	36	38	139	16	53	41	61	21	33	53	69	54	81	17	38	45	15	986		14.097	0.68295
antique	50	71	73	35	33	51	69	40	62	32	47	78	56	58	59	46	43	32	49	984		7.215	0.277477
cafe	39	60	83	55	36	44	64	37	71	34	60	71	39	60	63	40	50	27	39	972		8.539	0.30158
california	46	48	38	96	53	49	55	31	52	32	36	56	53	72	47	44	43	84	45	980		9.796	0.321725
colorful	104	17	22	42	384	7	10	31	25	5	24	16	55	60	67	25	22	41	20	977		39.304	1.636519
frenchchic	4	7	5	6	6	27	6	3	10	6	6	6	4	11	3	8	3	5	13	139		19.424	0.748758
industrial	23	41	37	36	11	34	90	42	52	55	22	82	40	31	30	26	36	32	16	736		12.228	0.514713
japanese	14	21	33	29	32	22	34	81	36	23	25	37	34	38	45	31	37	26	11	609		13.300	0.455861
junk	30	62	56	24	36	25	61	46	110	17	43	70	40	39	50	18	38	31	14	810		13.580	0.535081
monotone	16	30	21	41	6	77	58	29	27	342	30	51	15	49	17	63	18	25	63	978		34.969	1.419536
natural	32	48	67	58	44	46	34	57	62	41	91	36	43	78	37	55	32	45	75	981		9.276	0.325706
otokomae	37	41	60	55	16	31	94	44	86	69	35	120	39	55	39	27	51	52	21	972		12.346	0.510201
retro	69	47	43	41	100	24	41	66	48	26	32	52	88	64	88	35	44	43	23	974		9.035	0.436525
scandinavia	27	45	61	61	50	36	34	45	29	41	87	45	47	107	47	65	40	63	54	984		10.874	0.378189
showaretro	72	29	47	38	126	29	45	69	65	24	25	48	69	64	105	27	36	31	22	971		10.814	0.55804
simple	11	39	39	43	25	77	48	43	20	104	47	40	33	86	25	122	30	48	84	964		12.656	0.589233
vintage	50	49	52	29	62	42	70	51	57	38	38	73	55	68	59	38	68	47	27	973		6.989	0.265566
westcoast	36	48	43	95	41	51	45	33	51	43	57	55	47	79	41	41	33	100	34	973		10.277	0.380187
white	15	44	48	58	28	115	24	22	27	93	59	30	29	66	20	67	21	42	178	986		18.053	0.779623
	814	784	864	880	1228	803	935	811	951	1046	797	1019	855	1139	923	795	683	819	803				
																			合計	16949	全体正解率	14.17193	%
	縦軸：予想の分類結果							青色：正解															
	横軸：実際の分類結果							赤色：不正解の第1位															
	数字：回数							ピンク：不正解の第2位															

図 8.6　分類器⑥の精度計測で生成した混同行列

この混同行列から見ると、評価用データセットにあるカテゴリに対して、正解率が高いのは colorful と monotone で、低いのは antique、cafe と vintage である。一つの予想の分類結果のカテゴリにある全ての解の回数に対する変動係数も、同じ傾向で、正解率が高いカテゴリほど、変動係数が高い。

一部のカテゴリの写真に対して、回数が他と圧倒的な差がある不正解がある。例えば、american のカテゴリの写真に colorful と分類されたものが多数あり、逆に colorful カテゴリの写真に american と分類されたものも多い。white と frenchchic、monotone と simple、westcoast と california などカテゴリのペアにも同じ現象が見える。それから見ると、一部のカテゴリ（スタイル）の間に、共通の特徴があって混同しやすいと考えられる。この問題は、全体の正解率に大きな影響があるかもしれない。

次は、識別機の性能に関して簡単に述べる。

識別機の一括実行は、GPU と CPU を利用した場合、性能の差がほとんど出なかった。筆者が実装した識別機が GPU を充分に利用できないと考えられる。その一方、認識する画像が同じであっても、CNN に入力する画像のサイズが 100×100 の分類器②、③を使う場合、認識の所要時間は CNN に入力する画像のサイズが 40×40 の分類器④～⑥より、1 枚は 4 秒ほど長い。②と③のチェックポイントのファイルが 120 MB のサイズで、④～⑥の方が 20 MB しかない。筆者は識別機の Python スクリプトのステップごとの実行時間を測る命令を追加して、ボトルネックを探そうとしたが、ほとんどの時間が TensorFlow セッションの実行に使われたと表示したので、どちらのステップがボトルネックであることはまだ不明であった。

どちらにしても、1 枚の画像の認識の時間は CPU での学習の際 1 ステップで 1 バッチ（128 枚の画像）を学習した時間（分類器②の場合は、平均 5.47 秒）より長い。6.6 節にも述べたが、筆者はその性能のボトルネックは、チェックポイントの読み込みとメモリ上に CNN パラメータの構築にあると考えた。

8.7　CNN 構成のチューニング

第 3 段階の中盤に、サービス班からのフィードバックにより、分類の精度を更に上げられるかという要望があった。筆者は、データセットをさらにチューニングするには時間が足りないと考えて、CNN の知識をより深く勉強するため、その前 TensorFlow のサンプルと Tunnel 社のプロトタイプに基づいた、我々の学習機と識別機のコアである CNN の構成を改変し、最終版「11.3 版」教師データを学習し、効果を評価した。

第 6 章の図 6.3 で示した通り、我々の CNN は元々畳み込み層(conv)二つ、プーリング層(pool)二つ、正規化層(norm)二つ、さらに 3 つの全結合層で組み合わせたものである。筆者は、全結合層以外の層の有無と数を着手し、CNN の構成に調整を加えた。層それぞれのパラメータの改変は、筆者はその根拠がよく分からなく、時間も少ないので諦めた。

表 8.4 は、行った 4 回の CNN の改変の内容と、その改変が行われた学習機で生成した分類器を示す。改変する前のものの正解率と比較するため、すべての学習は、教師データが 11.3 版を使って、CNN の入力画像のサイズを 40×40、ステップ数を 300,000 に設定した。

表 8.4　CNN 構成のチューニングの内容

回数	改変の内容	学習の結果	生成した分類器
第 1 回	conv3, pool3 と norm3 の追加	学習が成功（300,000 ステップで誤差 0.05）	⑦
第 2 回	第 1 回の結果に基づいて、norm1, norm2 を削除	学習が成功（300,000 ステップで誤差 0.05）	⑧
第 3 回	第 1 回の結果に基づいて、更に conv4, pool4 と norm4 を追加	誤差が収束しない（300,000 ステップでも誤差 2 以上）	—
第 4 回	第 1 回の結果に基づいて、pool 層の全削除	学習の速度が大幅に下がった（改変前の 1/3 に）	—

第 6 章と付録 B の通り、CNN の構成は学習機と識別機の共通コアモジュール rc_module.py の関数 inference()に定義される。それにチューニングを与えると、学習機と識別機も共に新しいバージョンになる。

チューニング後で学習した分類器データは、改変された CNN の構成を定義した新しい共通コアモジュールを使用しなければならない。そのため、新しい学習機で生成した分類器データを使用する際、それに対応する新しい識別機も必要である。

第 3 回、第 4 回のチューニングを加えた学習機での学習では、誤差が収束しない、学習速度の大幅の下落など問題が発生したので、生成した分類器は破棄された。

第 1 回と第 2 回チューニングした学習機で生成した分類器⑦、⑧は、筆者が第 6 節

で述べた方法で、それぞれに対応する新識別機で正解率を計測した。そして、その二つの正解率を、第 6 節で述べた Tunnel 社のプロトタイプ（分類器①）の正解率、旧学習機で⑦、⑧と同じ条件（教師データが 11.3 版、入力画像のサイズが 40×40、ステップ数が 300,000）で生成した分類器⑤の正解率と比較した。その結果は、図 8.7 で示す。

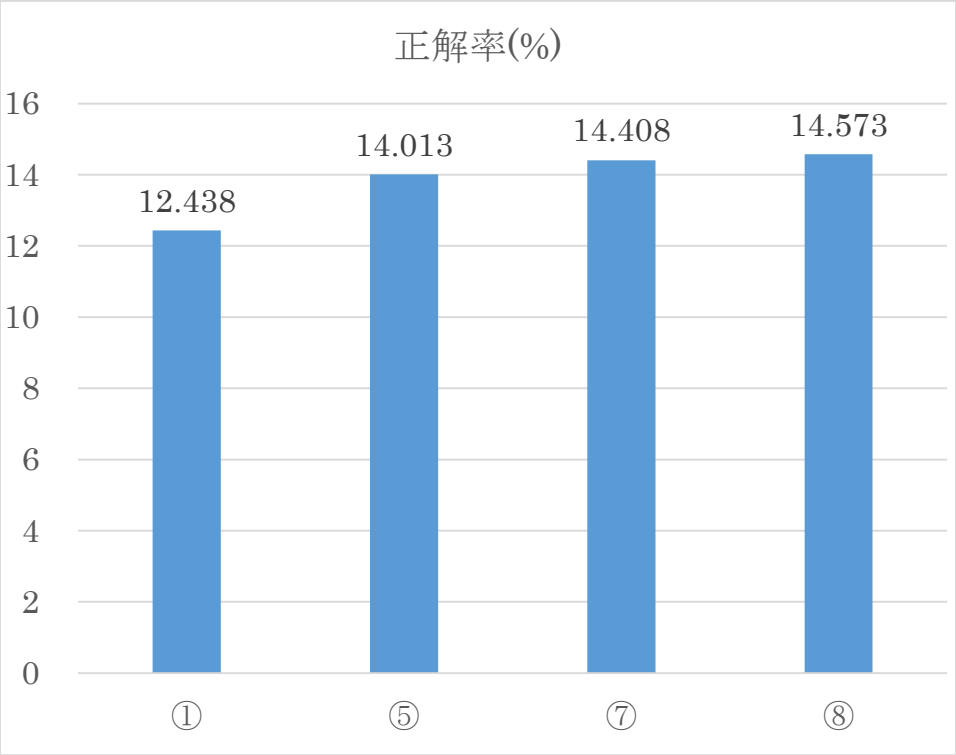

図 8.7　CNN 構成のチューニングにより正解率の向上

畳み込み層が 3 つになった⑦は、同じ条件で二つの畳み込み層がある⑤より、0.4%の正解率の向上が見えた。そして、⑦と比べて正規化層が減少した⑧は、更に 0.17%の正解率の向上が見えた。

CNN 構成チューニング後の各カテゴリの正解率については、以下図 8.8 で示した分類器⑧の計測で得られた混同行列で説明する。

	americ	antiq	cafe	califo	colorf	frenc	indus	japan	junk	mono	natura	otoko	retro	scand	show	simpl	vinta	west	white			正解率(%)	変動係数
american	175	26	49	47	114	19	39	49	60	28	20	51	57	52	77	22	55	34	12	986		17.748	0.73374
antique	47	47	57	47	29	34	74	59	76	36	56	66	46	55	69	34	57	38	57	984		4.776	0.266667
cafe	42	53	88	52	22	42	55	47	73	25	56	67	51	56	54	42	56	47	44	972		9.053	0.293374
california	41	26	39	117	53	41	50	47	51	44	44	44	50	62	48	38	40	93	52	980		11.939	0.40028
colorful	110	19	26	47	359	11	13	39	25	5	23	13	48	46	75	21	33	33	31	977		36.745	1.525594
frenchchic	4	4	12	2	1	30	14	0	4	9	8	4	4	9	1	6	3	5	19	139		21.583	1.001417
industrial	23	34	41	34	10	36	93	49	41	40	20	76	31	29	39	42	54	27	17	736		12.636	0.506739
japanese	22	26	23	20	32	24	40	87	38	18	30	35	38	34	39	44	34	13	12	609		14.286	0.507182
junk	40	39	54	31	30	30	52	44	102	21	46	62	51	33	42	27	48	36	22	810		12.593	0.42749
monotone	21	21	28	33	11	74	62	22	12	349	26	54	17	35	12	62	17	23	99	978		35.685	1.476687
natural	22	45	71	42	37	47	40	37	56	42	106	31	43	64	40	75	33	50	100	981		10.805	0.434677
otokomae	56	41	67	43	16	29	94	63	64	64	35	116	44	33	43	23	66	54	21	972		11.934	0.490213
retro	78	47	43	41	115	26	45	66	52	17	46	53	65	44	95	28	48	42	23	974		6.674	0.474712
scandinavia	55	34	44	57	46	47	39	59	36	57	68	37	53	101	40	72	38	50	51	984		10.264	0.308982
showaretro	103	39	57	49	122	21	38	72	42	19	27	43	63	41	113	25	48	30	19	971		11.637	0.608758
simple	13	30	37	53	11	70	57	62	22	78	66	25	30	74	21	115	39	48	113	964		11.929	0.599185
vintage	69	46	53	38	51	34	60	70	54	49	35	63	52	56	59	35	77	43	29	973		7.914	0.26179
westcoast	47	24	52	98	43	41	39	42	43	40	37	44	50	57	36	55	44	114	67	973		11.716	0.419953
white	12	45	29	45	20	109	19	27	22	90	63	27	17	72	17	86	24	46	216	986		21.907	0.942117
	980	646	870	896	1122	765	923	941	873	1031	812	911	810	953	920	852	814	826	1004				
																			合計	16949	全体正解率	14.57313	%
	縦軸：予想の分類結果							青色：正解															
	横軸：実際の分類結果							赤色：不正解の第1位															
	数字：回数							ピンク：不正解の第2位															

図 8.8　CNN 構成改変後の分類器⑧の精度計測で生成した混同行列

図 8.6 の分類器⑥の計測で生成した混同行列と比較すれば、正解率が上がったカテゴリは 12 個で、正解率が下がったカテゴリは 7 個である。正解率が最高のカテゴリは分類器⑥の場合と同じ、colorful と monotone である。各カテゴリに対する不正解を見ると、分類器⑥の場合とも似ていて、american/colorful、westcoast/california など混同しやすいカテゴリのペアがある。

結論としては、筆者が行った CNN 構成のチューニングは正解率の向上に有効であると説明された。

8.8　実験結果の分析と最終版の識別機の確定

筆者は、教師データセットと CNN の構成に色々試行錯誤を行った。と言っても、最終的に得られた 15%を下回る平均正解率は、筆者の予想より低い。この現象の原因は、筆者は二つの考えがある。

1)スタイルによる分類の難しさ

本章最初の部分にも述べたが、CNN による画像認識の先行研究に扱うデータは、CIFAR-10 や ImageNet などタグが付けられた画像のセットである。そのタグの付け方が明確で（「犬」「猫」「トラック」など物体)、客観的なものである。

それと異なって、今回我々の分類システムの目的は、「この写真は何？」ではなく、「この写真の雰囲気から、どのようなスタイルが感じた？」のようなことである。インテリアのスタイルというものは、抽象的、かつ主観的で、明確な定義がなく、人の目でも分からないことがある。さらに、前述の混同行列で説明したが、インテリアのスタイルの間に、明確に定義されなかった類似点もあるかもしれない。先行研究で成功した方法をそのままスタイルの分類に運用すると、想定した結果が出ない可能性が

高いと考えられる。

2)カテゴリ化方法の問題

RoomClip にアップロードされた画像のカテゴリ化は、教師データセットの作成の必須条件である。前述の我々のカテゴリ化方法は、RoomClip のユーザーらが追加したスタイルを表すタグが唯一の根拠である。

この方法のメリットとデメリットは共に明らかである。そのメリットは、インテリアの知識が全くない筆者のチームより、インテリアに関心がある RoomClip のユーザーによるタグの追加は信憑性が高い。一方、そのデメリットは、先述の通り、インテリアのスタイルは明確な定義がないので、人により特定のスタイルの定義、言い換えると特定のスタイルに感じた「雰囲気」が異なる。そのため、一つのカテゴリに対して、より代表的である写真を選ぶことでは、更なる措置が必要だと考えられる。

今まで我々の分類方法は、以上の問題により認識の精度が全体的に低い。といっても、筆者の識別機を運用するサービス班は、100%正確な認識を求めていなく、その認識の結果にある確信度の上位のカテゴリを使って、それに対応するタグをユーザーに推薦することが目的である。

筆者は、前述の正解率の計測で使用した、比較的に高い正解率を持つ⑤、⑥、⑦と⑧の四つの分類器とそれに対応する識別機をサービス班に提供した。サービス班のフィードバックは、以下 2 点である。学習のステップ数が多い⑥は、分類結果に確信度が 90%以上になる頻度が高くて極端的である。全体的の正解率が最も高い⑧は、逆に分類結果の確信度が高い写真が他の三つよりも少なかった。

そのため、最終的にシステムに使われた画像分類バックエンドは、出力が極端的でなく、正解率も比較的に高い⑦番の分類器とそれに対応する識別機のコンビネーションになった。

第 9 章　プロジェクトの技術サポート

9.1　サーバーの管理

プロジェクトが開始した時点、チームは研究室に余っている一台の PC を開発用サーバーとして準備した。

そのサーバーのスペックは、5.5 節で既に紹介した。元々Windows 8 がインストールされたが、筆者はその 1 TB のハードドライブを初期化し、Ubuntu Server 14.04 LTS と当時の開発で使った TensorFlow r0.9 をインストールし、環境を確保した。

6 月中に、チームリーダーとハードウェア系担当の藤田君に協力し、RoomClip の写真データ保存用の 6 TB のハードドライブを初期化し、Linux の/media/datadrive にマップした。その同時に、MySQL 環境を準備し、1.3 節で述べた Tunnel 社から頂いた RoomClip のデータベースの一部の内容を新しい MySQL データベースに導入した。

その後、Tunnel 社のプロトタイプの再現の際、Web フロントエンドの初期開発のため、開発用サーバーに更に PHP、Apache の実行環境を整えた。

Tunnel 社は、我々プロジェクトに 8 コア CPU、8 GB のメモリのある Azure のクラウドサーバーを提供した。外部から我々が作ったプログラムを実行するため、8 月、筆者はそのクラウドサーバーの環境を整備し、Web サーバー(Nginx)、PHP、MySQL と TensorFlow をインストールし､第 1 段階で作った全てのプログラムをアップロードし、実行に成功した。

10 月下旬、藤田君が行った GPU が装着した学習専用機の準備により、TensorFlow も GPU を対応する TensorFlow r0.11 にアップグレードされた。そのため、筆者は今まで TensorFlow を使用したプログラムを修正し、新しい TensorFlow に対応した。

その後の開発で､サーバーのトラブルシューティング(ソフトウェアの故障など)は、筆者がチームメンバーに積極的に協力した｡特に開発用サーバーにできたものを Azure クラウドサーバーへの配置は、ほぼ全て筆者が行った。

付録 G と H では､開発用サーバーと Azure サーバーにあるシステムの最終版のファイルの一覧が示されている。

9.2　チーム内でのファイル共有

筆者は、以前の開発での経験で、チーム全員がファイルを便利に共有できる、ドキュメントを共同に編集できる環境が開発に対して重要だと考えた。そのため、筆者はチーム内にファイル共有の手段として、Google Drive に共有フォルダーを作った。

共有フォルダーに権限があるのは、顧客である平山氏、課題担当教員である岡先生と橋本先生、そして我々チームの全員である。共同作業の時、チームメンバーが各自作っ

たファイルを共有フォルダーにアップロードし、バージョン管理できる。そして、ミーティングの時、チームメンバー全員共同に議事録を作ることもできる。

9.3　サービス班への支援

サービス班のメンバー二人は、Web 系やモバイルアプリの開発経験がある一方、Linux と Python の知識は全くなかった。そのため、筆者はサービス班の一部の開発にも協力した。

第 1 段階の最後で ZHANG SHUNAN 君は、PHP で識別機の簡単なフロントエンドを実装した。その際、重要な部分である識別機の呼び出し、返された JSON の解析のルーチン、すなわち識別機の API の実装とデバッグは、識別機を担当した筆者の協力で完成した。

さらに、最終評価の手段として、サービス班は 12 月から Tunnel 社から頂いた 2016 年 6 月までに RoomClip にアップロードされた、合計 140 万枚以上の写真を全て我々の識別機に通し、オーバービューを作る予定であった。筆者が実装した識別機の認識速度が遅く、最終版にしても一枚の写真の判定が 5 秒ぐらいかかる。認識の高速化を同時に試してみたが、1 台の PC で全ての判定を行うには時間が不足であった。そこで筆者は、チーム全員の研究室のデスクトップ PC、開発用サーバーを加えて合計 5 台の PC が同時にその 140 万枚の写真を判定する方法を設計した。

この方法は非常に直観的である。まずは、本書第 1 章と第 7 章に述べた RoomClip から頂いたデータベースから、140 万枚の写真ファイルの一覧をテキストファイルに出力する。そして、このテキストファイルを 5 つ部分に分けて、開発用の Web サーバーとチームメンバー4 人の PC に置く。前述の通り、そのデータベースにある写真ファイル名は写真の内容と 1 対 1 の関連があり、唯一性が保証されたので、同一の写真が複数の PC で認識され、重複の分類結果がサービス班のデータベースに書き込まれてしまうことはないであろう。

判定する全て 140 万枚の写真ファイルは、既に RoomClip から開発サーバーにダウンロードされて、研究室の LAN に HTTP でアクセスできる。さらに、その写真ファイルのサイズは、今までの学習と正解率計測で使った識別が一番速い、細かい要素もある程度残せる 320×320 にリサイズした。

筆者は、最終版の識別機と分類器をチームメンバー全員の 4 台の PC に配置し、実行可能であることを確認した。それに加え、サービス班の張墨君が開発した一括判定スクリプトを実行して、開発用 Web サーバーを含める 5 台の PC それぞれにある写真ファイルの一覧を読み込んで、画像を 1 枚ずつダウンロードして判定する。その判定結果は、Web サーバーにある MySQL データベースに記録される。

図 9.1 は、一括判定のアーキテクチャを示す図である。

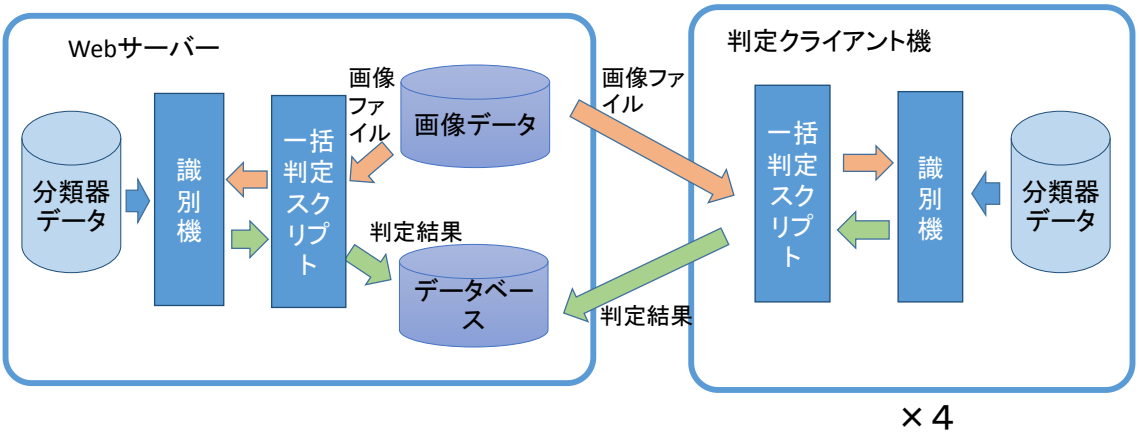

図 9.1　140 万枚の写真の一括判定システム

12 月 3 日に、140 万枚の写真の一括判定が開始した。当初 12 月末でその判定が完成できると予想したが、色々問題が発生した。例えば、Apache、MySQL を実行されている Web サーバーで CPU に高い負荷をかける識別機を連続に実行すると、不安定になる場合がある。その原因で、12 月に Web サーバーで行った一括判定は数回に Python の Error -11 で予想外に中止した以外、Apache のクラッシュにより全てのクライアント機が画像を取得できなく、判定が中止したことも 1 回発生した。

そのような不具合を乗り越えて、一括判定は 2017 年 1 月 6 日午前 2 時ぐらいにようやく完成した。判定した写真は、合計 1,437,664 枚であった。所要時間は 34 日で予定より長くなったが、プロジェクト終了前にオーバービューを作成することができた。

第 10 章　振り返りと今後の展望

10.1　プロジェクトの振り返り

筆者は、本プロジェクトの運営と開発で、色々経験と教訓を得た。

経験として、まず、筆者はこれまで勉強しなかった機械学習の知識を少し身につけた。参考資料を読み始めた時、筆者は、最近に囲碁などこれまで AI ができない領域で莫大な成功を得たディープラーニングの技術が魔法のようだと思った。勉強の進みにより、その仕組みは未だに完全に理解できていないが、それほど難しいものではないと分かった。

そして、それまで経験したものと異なり、仕様が明確ではない開発に関する経験を身につけた。Tunnel 社がプロジェクト最初に我々に提出した要望は、明確ではなく、単にディープラーニングを用いた面白いサービスを作ることであった。我々開発チームは、プロジェクトの第 1 と第 2 段階で、具体的に何を作るというアイデア提出に力を入れ、最終にチームの能力に適合するアプローチを決めた。

さらに、最終目標が決定された 10 月後半から報告書を書く 12 月後半まで、残っている開発時間が 2 か月という緊迫な状況を乗り越えて、チームがシステムに実現すべき機能を基本的に完成した。

その一方、どの側面から見ても、今回のプロジェクトは大成功とは言えない。進捗の遅さ、識別機の正解率の低さ、テストや評価方法の不完全など問題があり、最終的に作ったシステムは完成度が低く、製品として言えない。その原因は、主に我々開発チーム自身にある。

第 7 章で説明したが、分類器の学習は CPU のみでは非常に遅い。今回我々は、プロジェクトの方向性が確定された 10 月中旬という遅い時点で学習専用機と GPU を導入した。筆者が作った教師データセットの学習には、GPU を利用して 10 倍以上の速度向上が見えたが、もし第 1 段階から GPU を導入すれば、学習時間の短縮で、分類器の精度向上のためのチューニングではより多い回数の試行錯誤ができ、より良い分類器を作れるであろう。

筆者自身は、教師データセットを作ったところ、始めから Tunnel 社のプロトタイプのアプローチ、つまり、RoomClip のユーザーが付けたスタイルを表すタグで画像をカテゴリ化するということが正しいと信じた。しかし、その方法で作ったデータセットで学習してきた分類器の分類精度は、予想外に低かった。確かに我々のプロジェクトは、インテリアの専門家が選んだインテリアのスタイルの代表的な写真でデータセットを作るのは非現実的であったが、もしプロジェクトの始めから、余った時間を利用して手動で RoomClip の写真をスタイルによりカテゴリ化できたら、分類器の精度がそこまで低くないであろう。

プロジェクトの運営面にも、筆者は色々問題を発見した。

一番大きな問題は、ドキュメントを作る意識の不足である。最近世の中には、伝統的な「ウォーターフォール」など製品開発のモデルがよく批判され、「アジャイル」などドキュメントへの依頼が少なく、仕様の変更に反応が速い開発モデルが流行しているが、我々のチームが残したドキュメントが少なさすぎて、その形式も不規則である。本報告書を作成した際、参照できるドキュメントがほとんどなく、機能面の詳しいことはソースコードを読みながら書いた。

そして、開発時間の利用が甘く、浪費が非常に多い。特に、分類器の作成と精度向上を担当した筆者は、2 か月の夏休みという貴重な時間を充分に利用できなく、一時帰国でその半分以上の時間を浪費した。

さらに、開発チームの団結力が低いということも問題になった。毎週に行うべき共同作業が、中間発表の直前など以外に行った回数が少ない。そのため、考え方の統一はおろか、メンバーの意見の交換すら少ない。長期休暇以外に原則的に毎週で行う会議も、10 月に入ってから欠席者が出る状況が出た。筆者は、これが開発全体の効率に大きな影響があると考えた。

以上の問題で、本プロジェクトで得られた成果物は、もう一つのプロトタイプのようなものになった。だが、この開発で、筆者自身を含めた開発チームのメンバーの成長が見えた。本プロジェクトで得られた経験と教訓は、今後の仕事と生活の糧になる。

10.2　今後の展望

本プロジェクトは、決めた 19 種類のインテリアのスタイルによりインテリア写真を分類し、そのスタイルに相応しいタグをユーザーに推薦するシステムを開発した。Tunnel 社はこの技術をどのような方式で運用するか、筆者は少し考えた。

前ほど何回も述べた通り、インテリアのスタイルは主観的なものであり、人により感じたものが異なる。より精度が高い分類を実現するのは、やはりインテリアに詳しい人による写真の再カテゴリ化が必要になると考えた。さらに、インテリアのスタイルは決して 19 種類だけではないので、マイナーなスタイルを含めた、更に細かいカテゴリ化方法があれば、分類の精度が更に高い実用的な識別機が作れるであろう。

そして、大規模な Web サービスに当たって、識別機の効率と安定性への要求は非常に高い。我々が作ったサービスで実現した 5 秒ほどかかる認識は、RoomClip のような規模のサービスには決して足りない。このサービスを RoomClip に実装するため、ハードウェアとソフトウェアの両方面から、効率と安定性を考慮し再設計する必要がある。

第 11 章　おわりに

インテリア写真に特化した写真共有サービス RoomClip では、写真の検索や分類は、主に写真に追加した情報（名称、タグ、説明文）など、特にタグを用いて行う。だが、そのタグ機能を有効に利用していない利用者が多く、検索機能の有効性に影響がある。さらに、他の写真共有サービスと更に差別化するため、新技術を用いる独特な新機能が必要となった。その問題を解決するため、本プロジェクトでは、ディープラーニングの技術でインテリア写真を分類し、その分類結果でユーザーにタグを推薦する機能、更に全ての分類結果の一覧を表示し、分析するシステム「D-ROOM」を開発した。

筆者はこの開発に当たって、TensorFlow のディープラーニングのチュートリアル、および Tunnel 社がその前に開発したプロトタイプ「インテリアのスタイル判定」を参考とし、畳み込みニューラルネットワーク（CNN）の技術で D-ROOM に使われた識別機と分類器を生成する学習機を実装した。そして、ユーザーが写真に追加したスタイルを表すタグでの検索により、RoomClip にアップロードされた画像をカテゴリ化し、学習に使われる教師データセット、正解率で定義された分類の精度を計測するための評価用データセットの作り方を確立した。

しかし、計測の結果から見ると、その分類の精度は非常に低く、実用的なものとは言えない。カテゴリ化方法と CNN の構成の両方面から精度向上のチューニングを行い、基準となる Tunnel 社のプロトタイプより 2%ほど精度の向上ができたが、まだ実用的なレベルに達してない。

結論として、筆者を含めた開発チームが開発してきたシステム「D-ROOM」は、Tunnel 社の要望を一部達成したものの、RoomClip に実際に運用するのは、更に力を入れる必要がある。大成功とは言い難いが、筆者はこの開発で、機械学習の基礎知識を身につき、Linux と Python でのシステム開発の腕を磨いた。筆者が本プロジェクトで得た経験と教訓を、技術者として将来の生活と仕事で活かせると考えている。

謝辞

本プロジェクトを進めるにあたり、Tunnel 株式会社担当執行役員の平山知宏様は、ご多忙の中に、我々のチームに RoomClip のデータとプロトタイプ「インテリアスタイル判定器」のソースコードの提供など、プロジェクトの成敗に関わる重要なご支援を頂きました。筆者はこの場を借りて、深く御礼を申し上げます。

課題担当教員である岡瑞起准教授、橋本康弘助教は、数多くの会議に参加頂き、ディープラーニングの専門的な知識とプロジェクトの方向性の決定から、特定課題研究報告書の執筆まで、日ごろから熱心にご指導を頂きました。筆者はここで深く感謝を申し上げます。

筆者の指導教員である高橋伸准教授は、このプロジェクトの遂行と並びに、中間報告書の検査と、この特定課題研究報告書の執筆において、多くのご指導を頂きました。誠にありがとうございました。

本プロジェクトのメンバーである、藤田卓也君、張曌君と ZHANG SHUNAN 君と、この 1 年間にお互いに支えて、とても有意義なチーム開発ができました。お世話になりました。

最後に、これまで大学院生活を一緒に過ごした高度 IT コースのみんなに、心より感謝致します。

参考文献

[1]総務省， ICT の進化によるライフスタイル・ワークスタイルの変化, 平成 26 年度版情報通信白書, 第 1 部, 第 4 章, 第 1 節, 第 1 項, 2014.
[2]ICT 総研, “2016 年度 SNS 利用動向に関する調査”, < http://ictr.co.jp/report/20160816.html>(2016/12/9 アクセス).
[3]Tunnel 社, “企業様お問い合わせ ｜ RoomClip（ルームクリップ）”,<https://roomclip.jp/doc/official>,(2016/6/29 アクセス).
[4]平山知宏, “Deep Learning (TensorFlow) で インテリアスタイル判定器を 作ってみるテスト // Speaker Deck: ”, SpeakerDeck, <https://speakerdeck.com/hirayama/deep-learning-tensorflow-te-interiasutairupan-ding-qi-wo-zuo-tutemirutesuto>, (2016/12/9 アクセス).
[5]岡谷貴之，深層学習，杉山将（編），まえがき，講談社，東京，2015.
[6]岡谷貴之，深層学習，杉山将（編），pp.79-82，講談社，東京，2015.
[7]麻生英樹， 安田宗樹， 前田新一， 岡野原大輔， 岡谷貴之， 久保陽太郎， ボレガラ ダヌシカ（著），“階層型ニューラルネットワークによる深層学習，” 深層学習－Deep Learning－，神嶌敏弘（編），第 1 章，近代科学社，東京，2015
[8] LeCun, Yann, et al. "Learning algorithms for classification: A comparison on handwritten digit recognition." *Neural networks: the statistical mechanics perspective* 261 (1995): 276.
[9] LeCun, Yann, et al. "Handwritten Digit Recognition with a Back-Propagation Network." *Advances in Neural Information Processing Systems*. 1990.
[10] Schapire, Robert E. "The strength of weak learnability." *Machine learning* 5.2 (1990): 197-227.
[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
[12] TensorFlow, “About TensorFlow”, < https://www.tensorflow.org/about.html> (2016/6/22 アクセス).
[13] TensorFlow, “Convolutional Neural Networks”, <https://www.tensorflow.org/versions/r0.11/tutorials/deep_cnn/index.html>(2016/11/5 アクセス).
[14] GitHub, “tensorflow/tensorflow/models/image/cifar10 at r0.11 · tensorflow/tensorflow:”<https://github.com/tensorflow/tensorflow/tree/r0.11/tensorflow/models/image/cifar10>(2016/12/15 アクセス).
[15] 岡谷貴之，深層学習，杉山将（編），pp.10-11，講談社，東京，2015.
[16] 麻生英樹，安田宗樹，前田新一，岡野原大輔，岡谷貴之，久保陽太郎，ボレガラ ダ

ヌシカ（著），“大規模深層学習の実現技術，”　深層学習－Deep Learning－，神嶌敏弘（編），第 4 章，近代科学社，東京，2015

付録 A　教師データの写真のファイル名を選ぶ SQL スクリプト

ここでは、我々が Tunnel 社から頂いた RoomClip データベースの一部から、教師データセットに入れる写真のファイル名を検索する SQL スクリプトである。

1)7 月版

7 月版では、指定のタグがあれば選ぶという簡単な仕組みである。

```
# getname.sql
use roomclip_droom;

select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='アメリカン') order by rand() limit 1000 into outf
ile '/tmp/american.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='アンティーク') order by rand() limit 1000 into ou
tfile '/tmp/antique.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='カフェ') order by rand() limit 1000 into outfile
 '/tmp/cafe.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='カリフォルニア') order by rand() limit 1000 into
outfile '/tmp/california.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='カラフル') order by rand() limit 1000 into outfi
le '/tmp/colorful.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='フレンチシック') order by rand() limit 1000 into
outfile '/tmp/frenchchic.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='インダストリアル') order by rand() limit 1000 int
o outfile '/tmp/industrial.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='ジャンク') order by rand() limit 1000 into outfi
le '/tmp/junk.txt' lines terminated by '¥n' ;
```

```
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='男前') order by rand() limit 1000 into outfile '
/tmp/otokomae.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='モノトーン') order by rand() limit 1000 into outf
ile '/tmp/monotone.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='ナチュラル') order by rand() limit 1000 into outf
ile '/tmp/natural.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='レトロ') order by rand() limit 1000 into outfile
 '/tmp/retro.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='昭和レトロ') order by rand() limit 1000 into outf
ile '/tmp/showaretro.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='北欧') order by rand() limit 1000 into outfile '
/tmp/scandinavia.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='ビンテージ') order by rand() limit 1000 into outf
ile '/tmp/vintage.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='シンプル') order by rand() limit 1000 into outfi
le '/tmp/simple.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='ホワイト') order by rand() limit 1000 into outfi
le '/tmp/white.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='西海岸') order by rand() limit 1000 into outfile
 '/tmp/westcoast.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, photos p  where p.id=t.photo_id and  t.tag
_id in (select id from tag where name='和風') order by rand() limit 1000 into outfile '
/tmp/japanese.txt' lines terminated by '¥n' ;
```

2)最終版(11.3 版)

7 月版と異なり、全てのタグの一覧に一つだけタグがある写真を選んで、その中から

カテゴリを表すタグの集合に所属するものを選ぶ。

```
# getname-1031_hand.sql
use roomclip_droom;

#  カテゴリを表すタグを一つだけ付けられた写真の ID とファイル名を臨時テーブル nphotos に
create temporary table single_category_tag_photo
select photo_id from
(select photo_id, count(*)cnt from tag_photo_relation where tag_id in
(select id from tag where name in ('アメリカン',  'ビンテージ', 'ヴィンテージ', 'アンティーク','カフェ','カフェ風','カリフォルニアスタイル','カリフォルニア', 'カラフル', 'フレンチシック', 'インダストリアル', 'ジャンク', '男前', 'モノトーン', 'ナチュラル', 'レトロ', '昭和レトロ', '北欧', 'シンプル', 'ホワイト', 'ホワイトインテリア', '西海岸', '西海岸インテリア', '和風', '和', '和モダン'))
group by photo_id)a
where a.cnt=1;

create temporary table nphotos
select id, filename from photos p, single_category_tag_photo s
where p.id=s.photo_id;

# ここで意味が同じタグは一つのカテゴリに合併される
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.tag_id in (select id from tag where name='アメリカン') order by rand() limit 1200 into outfile '/tmp/american.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.tag_id in (select id from tag where name='アンティーク') order by rand() limit 1200 into outfile '/tmp/antique.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.tag_id in (select id from tag where name='カフェ' or name='カフェ風') order by rand() limit 1200 into outfile '/tmp/cafe.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.tag_id in (select id from tag where name='カリフォルニア' or name='カリフォルニアスタイル') order by rand() limit 1200 into outfile '/tmp/california.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.tag_id in (select id from tag where name='カラフル') order by rand() limit 1200 into outf
```

```
ile '/tmp/colorful.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='フレンチシック') order by rand() limit 1200 into
 outfile '/tmp/frenchchic.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='インダストリアル') order by rand() limit 1200 in
to outfile '/tmp/industrial.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='和風' or name='和' or name='和モダン') order by
rand() limit 1200 into outfile '/tmp/japanese.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='ジャンク') order by rand() limit 1200 into outf
ile '/tmp/junk.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='モノトーン') order by rand() limit 1200 into out
file '/tmp/monotone.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='ナチュラル') order by rand() limit 1200 into out
file '/tmp/natural.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='男前') order by rand() limit 1200 into outfile
'/tmp/otokomae.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='レトロ') order by rand() limit 1200 into outfil
e '/tmp/retro.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='北欧') order by rand() limit 1200 into outfile
'/tmp/scandinavia.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='昭和レトロ') order by rand() limit 1200 into out
file '/tmp/showaretro.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='シンプル') order by rand() limit 1200 into outf
ile '/tmp/simple.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='ビンテージ' or name='ヴィンテージ') order by rand
```

```
() limit 1200 into outfile '/tmp/vintage.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='西海岸' or name='西海岸インテリア') order by rand
() limit 1200 into outfile '/tmp/westcoast.txt' lines terminated by '¥n' ;
select p.filename from tag_photo_relation t, nphotos p  where p.id=t.photo_id and  t.ta
g_id in (select id from tag where name='ホワイト' or name='ホワイトインテリア') order by
rand() limit 1200 into outfile '/tmp/white.txt' lines terminated by '¥n' ;
```

付録 B　学習機と識別機の各モジュールの機能の詳細情報

表 B.1　ファイル I/O モジュールの機能

機能	機能の説明	関数や変数の名称
共通設定	データのディレクトリ、分類の数、CNN ネットワークに送る画像のサイズなどの設定	num_classes fixed_image_size data_dir_path
分類の名称の読み込み	ディレクトリ構成から各分類の名称を読み込む	get_label_names()
分類と写真の紐づけ	各分類に所属する教師データの写真を使って、(分類,写真)のペアをメモリ上に作る	_get_images_and_labels()
写真データを TFRecord に入れ込み	分類情報を含めた写真データを全て TFRecord ファイルに一時保存する	_images_and_labels_to_TFRecord()
TFRecord の読み出し	TFRecord ファイルに保存した写真データをメモリに読み出す	read_data(tfrecords_path)

表 B.2　共通コアモジュールの機能

機能	機能の説明	関数名や変数名
学習のパラメータ設定	一部ユーザーが設定する学習のパラメータ(バッチサイズ、減衰、学習の速度など)	batch_size num_epochs_per_decay initial_learning_rate learning_rate_decay_factor moving_average_decay
画像の予備処理	教師データの画像のリサイズ、ランダム性の追加、コントラストの標準化	distort()
データのバッチの生成	教師データからランダムにバッチを作って、同時にそれに対する正解と紐づけ	generate_image_and_label_batch()
データを CNN に通す	CNN の構成を定義し、データをそれに通す	inference()
誤差の計算	バッチに対して、カテゴリの正解との誤差を計算する	loss()
全体の誤差の計算	計算した全ての誤差の指数移動平均値を計算する	_add_loss_summaries()
学習セッションのパラメータの初期化	全ての初期パラメータを反映し、学習のデータ構成を準備	create_train_op()

表 B.3　学習機モジュールの機能

機能	機能の説明	変数名や関数名
学習機のパラメータ	最大ステップ数、教師データのディレクトリ名設定など	train_dir train_tfrecords max_steps
学習の実行	フローチャートに参照	train()
メイン関数	TensorFlow 全体の初期化、学習の実行	main()

表 B.4　識別機モジュールの機能

機能	機能の説明	変数名や関数名
メイン関数	TensorFlow を初期化、識別の実行	`main()`
認識の準備	認識するファイルの読み込みと予備処理、分類名の読み込み、セッション設定	`evaluate_local_file()`
認識の実行	CNN セッションの再現、セッションの実行、JSON の出力	`eval_once()`

付録 C　学習機と識別機のコマンドライン説明

ここでは、学習機と識別機のコマンドラインについて、表で示す。

表 C.1 学習機のコマンドライン説明

引数	必須	使用方法
最大のステップ数	いいえ	`--max_steps=[`最大の学習ステップ数`]` デフォルトは `1000000`
バッチのサイズ	いいえ	`--batch_size=[`バッチに含めるサンプル数`]` デフォルトは `128`
データのディレクトリ	いいえ	`--data_dir_path=[`ディレクトリのパス`]` デフォルトは `rc_io.py` に設定済

表 C.2 識別機のコマンドライン説明

引数	必須	使用方法
識別する画像のファイルパス	はい	`--local_file=[`画像ファイルのフルパス`]`
データのディレクトリ	いいえ	`--data_dir_path=[`ディレクトリのパス`]` デフォルトは `rc_io.py` に設定済

付録 D　正解率計測スクリプトのコマンドライン説明と一括計測 Shell スクリプト

筆者の正解率判定スクリプトは、カテゴリ一つ一つ判定する必要がある。19 個のカテゴリを全て判定するため、別途 Shell スクリプトを書く必要がある。

表 D.2 正解率判定スクリプトのコマンドライン説明

引数	必須?	使用方法
カテゴリの名称	はい	`--tag=[`カテゴリの名称`]`
データセットの位置	はい	`--imagepath=[`データセットのパス`]`
詳しいログの出力	いいえ	`--nodebug=[0` や `1]` デフォルトは `0`、詳しいログ（毎回の認識結果）は出力されない

筆者が指定した 19 種類のカテゴリを一括に判定する Shell スクリプトは、そのソースコードをここで説明する。

```
# calc_correct.sh
# Temporally disable GPU for now..
export CUDA_VISIBLE_DEVICES=
DATASET=$1

python calc_correct.py --imagepath="$DATASET" --tag='american' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='antique' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='cafe' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='california' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='colorful' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='frenchchic' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='industrial' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='japanese' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='junk' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='monotone' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='natural' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='otokomae' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='retro' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='scandinavia' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='showaretro' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='simple' --nodebug='1'
```

```
python calc_correct.py --imagepath="$DATASET" --tag='vintage' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='westcoast' --nodebug='1'
python calc_correct.py --imagepath="$DATASET" --tag='white' --nodebug='1'
```

ここで calc_correct.py は正解率計測スクリプトのファイル名で、引数$1 はデータセットの位置となる。

付録 E　識別機の配置方法

筆者が実装した識別機には、他の Linux PC（Python 2.7 と TensorFlow r0.11 がインストールされた環境）に移行しても使用できる。

必要のファイルは、付録 G で紹介したファイルと eval_images、train_images の下の全てのサブディレクトリ（画像ファイルは不要）である。

ただし、識別機のソースファイルと、学習で生成した分類器のチェックポイントファイルに両方ファイルのパスが書き込まれたので、それらを修正し、新しい環境のパスに直す必要がある。

1) I/O モジュール rc_io.py

以下の行を探して、更新する。

```
#Location of Data
tf.app.flags.DEFINE_string('data_dir_path',              '/home/droom/test-8/data',
'data_dir_path')
```

data_dir_path の後ろのパスを、実際にデータ（models、eval_images、train_images）が置かれるディレクトリのパスに修正する必要がある。

2) チェックポイントファイル models/checkpoint

第 1 行の

```
model_checkpoint_path: "/home/droom/test-8/data/models/model.ckpt-299999"
```

このパスを実際のディレクトリのパスに直す必要がある。

付録 F　本システム各部分のファイル一覧

本文第 5 章で割愛したが、ここで本システム各部分のファイルの一覧と簡単な紹介を、表 F.1 で示す。

表 F.1　システム各部分のファイル一覧

部分	ファイル名	役割
識別機、学習機共通	rc_module.py	共通モジュール
	rc_io.py	I/O モジュール
識別機	rc_eval.py	識別機本体
	data/models/*	学習機が生成された CNN モデル
	data/eval_images/*, data/train_images/* (サブディレクトリのみ、画像ファイルが要らない)	各分類の名前
学習機	rc_train.py	学習機本体
	data/train/*.txt	教師データの写真ファイル一覧
	data/train_images/*/*.jpg	教師データの写真ファイル
Web アプリケーション	index.php	ホームページ
	upload_img.php	写真のアップロード、識別結果の表示
	tag.php	タグの推薦
	imgSave.php	識別結果をデータベースに保存
	db_config.php	データベースの設定
	db_class.php	データベースの SQL ルーチン
	similarity.php	類似の写真の表示
	view_all.php	識別した写真の一覧表示
	statistics.php	統計情報の表示
	search.php	検索条件の入力
	search_rs.php	検索結果の表示
	css/*	Web ページの様式
	js/*	Web ページに必要な Javascript

付録 G　開発サーバーのディレクトリ構成

以下の表 G.1 は、12 月末時点、我々の研究室にある開発サーバーにある本システムのファイルの構成である。

表 G.1 開発サーバー上にあるディレクトリ構成

部分	パス	ファイル	役割
識別機	/media/datadrive/ser_styledet_last	rc_io.py	共通 I/O モジュール
		rc_module.py	共通コアモジュール
		rc_eval.py	識別機
		data/models/*	最新の分類器モデル
		data/eval_images/*, data/train_images/*	カテゴリ名の一覧
アップロードされた画像ファイル	/media/datadrive/get-service-data/tdata_images/test	*.jpg	Web アプリケーションにアップロードされた全画像。全ユーザーに書き込み権限あり
RoomClip の画像	/media/datadrive/roomclip-bucket/img_640/	*.jpg	最初に RoomClip からダウンロードした全画像
	/media/datadrive/roomclip-bucket/img320/	*.jpg	140 万枚の判定のためリサイズした画像
Web アプリケーション	/var/www/html/shunan	*.php	Web アプリケーションのソースコード
		js/*	クライアントで実行する JavaScript
		css/*	HTML の様式
		resize/*	アップロード写真の予備処理の臨時ディレクトリ。apache ユーザーに全権限がある

付録 H　Azure サーバーのディレクトリ構成

表 H.1 は、現時点 Tunnel 社が我々に提供した Azure サーバーにシステムのファイル構成である。ファイルがほとんど開発サーバーと同じので、ここで紹介するのは各ディレクトリのパスである。ストレージスペースが足りないので、RoomClip の画像はここで保存しない。

表 H.1 Azure サーバー上にあるディレクトリ構成

部分	パス	コメント
識別機	/home/azureuser/ser-styledet-last	
アップロードされた画像ファイル	/home/azureuser/droompic	全ユーザーが書き込み可能
Web アプリケーション	/var/www/shunan	サブディレクトリ resize は、全ユーザーが書き込み可能