FUJITSU AI ソリューション

Zinrai ディープラーニング システム
機能説明書

Version 3.0

P3KD-1042-03

はじめに

本書は、FUJITSU AI ソリューションの Zinrai ディープラーニング システムを利用して、学習／推論を行うための機能、および運用管理方法を説明します。

本書の読者

本書は Zinrai ディープラーニング システム上で、学習／推論される方を対象として説明します。
本書を読むにあたって、以下の知識が必要です。

- ディープラーニングに関する基本的な知識
- Jupyter および Python コードに関する基本的な知識

本書の構成

本書の構成は以下のとおりです。

● **第 1 章 概要**

Zinrai ディープラーニング システムの概要、およびシステム構成を説明します。

● **第 2 章 バッチ型学習**

バッチ型学習で提供するディープラーニングフレームワーク、学習ユーティリティの機能を説明します。

● **第 3 章 対話型学習**

対話型学習で提供するディープラーニングフレームワーク、学習（推論）の機能を説明します。

● **第 4 章 対話型へのネットワークインターフェース追加**

対話型環境の Docker コンテナに、外部から直接接続する機能を説明します。

● **第 5 章 エッジ連携**

エッジ端末およびエッジモデルの管理方法など、提供する機能を説明します。

● **第 6 章 知識ライブラリ**

知識ライブラリの機能、知識ライブラリへのデータ登録方法を説明します。

● **用語集**

Zinrai ディープラーニング システム固有の用語、および関連する用語を説明します。

関連ドキュメント

以下の関連ドキュメントがあります。必要に応じて参照してください。

ドキュメント名称	概要
FUJITSU AI ソリューション Zinrai ディープラーニング システム ユーザーズガイド	Zinrai ディープラーニング システムを使用した学習・推論、および各種データの管理方法を知りたい場合にお読みください
FUJITSU AI ソリューション Zinrai ディープラーニング システム 機能説明書（本書）	Zinrai ディープラーニング システムの機能を知りたい場合にお読みください
FUJITSU AI ソリューション Zinrai ディープラーニング システム ソフトウェアライセンス条件について	Zinrai ディープラーニング システムで使用している、OSS（オープンソースソフトウェア）の使用許諾条件を記載しています

本書の表記について

本書では、略称および記号を以下のように使用しています。

製品名／技術名の略称について

本書では、製品名／技術名を以下のように表記しています。

製品名／技術名	略称
Deep Neural Network	DNN
Google Chrome™ ブラウザ	Chrome
Graphics Processing Unit	GPU
Network Attached Storage	NAS
Neural Network	NN

記号について

本書では、参照先、キー、メニューなどを表記するために、以下のように記号を使用します。

記号	意味
「 」	本書内の参照先のタイトル、画面での設定値を「 」で囲んでいます
『 』	他マニュアル参照のマニュアル名を『 』で囲んでいます
[ ]	画面のボタン名、タブ名、ドロップダウンメニュー、およびキーボードのキー名を示します 例：[設定] ダイアログボックス、[ファイル] メニュー、[項目名]、 [OK] ボタン、[Enter] キー
[ ] - [ ]	画面のメニューとメニューの階層を示します 例：[New] メニューの [Terminal] の場合 [New] - [Terminal]

記号	意味
[ ] + [ ]	同時に押すキーを示します 例：[Alt] キーを押しながら、[Tab] キーを押す場合 [Alt] + [Tab]
[|]	コマンド入力で、パラメーターの選択肢を示します 例：[A|B]

コマンドインターフェースの説明では、以下のような記号を使用します。

記号	意味
XXXX	値や文字列が可変であることを表す場合、斜体（イタリック体）の文字を使用、または < > で囲みます
<XXXX>	

また、以下のアイコン表記を使用します。

注　意　操作や設定を行ううえで制限される内容や注意が必要な内容が書いてあります。

備 考　操作や設定を行ううえで知っておくと便利な機能や使い方など、本文を補足する内容が書いてあります。

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

高度な安全性が要求される用途への使用について

本サービスは、一般事務用、パーソナル用、家庭用、通常の産業等の一般的用途を想定して開発・設計・製造されているものであり、原子力施設における核反応制御、航空機自動飛行制御、航空交通管制、大量輸送システムにおける運行制御、生命維持のための医療用機器、兵器システムにおけるミサイル発射制御など、極めて高度な安全性が要求され、仮に当該安全性が確保されない場合、直接生命・身体に対する重大な危険性を伴う用途（以下「ハイセイフティ用途」という）に使用されるよう開発・設計・製造されたものではありません。
お客様は本サービスを必要な安全性を確保する措置を施すことなくハイセイフティ用途に使用しないでください。また、お客様がハイセイフティ用途に本サービスを使用したことにより発生する、お客様または第三者からのいかなる請求または損害賠償に対しても富士通株式会社およびその関連会社は一切責任を負いかねます。

商標

- Linux® は米国及びその他の国における Linus Torvalds の登録商標です。
- Ubuntu は、Canonical Ltd. の登録商標です。
- Docker is a trademark or a registered trademark of Docker, Inc. in the United States and/or other countries.
- Apache は、Apache Software Foundation の商標または登録商標です。
- NVIDIA、CUDA は、米国および / または他国の NVIDIA Corporation の商標および / または登録商標です。
- Apple、Apple のロゴ、Mac OS は、米国および他の国々で登録された Apple Inc. の商標です。
- IOS は、Cisco の米国およびその他の国における商標または登録商標であり、ライセンスに基づき使用されています。
- Google、Android は、Google Inc. の商標または登録商標です。
- その他の会社名、各製品名などの固有名詞は、各社の商号、登録商標または商標です。
- その他、会社名、システム名、製品名などには必ずしも商標表示を付記しておりません。

2018 年 10 月 第 3 版

改版履歴表

（1/1）

版数	日付	変更箇所（変更種別）（注）	変更内容
初版	2017 年 5 月	全体	新規作成
02	2017 年 11 月	全体	機能追加
03	2018 年 10 月	1.2（追加）	MXNet、Keras を追加
		2.2.2（変更）	学習の中断の説明を変更
		2.2.2（追加）	学習の停止の説明を追加
		第 3 章（追加）	• 対話型種別の種類を追加 • 対話型環境の中断を追加 • 各フレームワークの情報を最新化 • 学習モデル形式変換について追加 • 注意事項を最新化
		6.3.1（削除）	「知識ライブラリからのデータダウンロード」を削除
		6.3.1.1（追加）	「モデルセット、Edge 用データ、その他」のデータ構成を追加

注）変更箇所は最新版の項番を示しています。ただし、アスタリスク（*）の付いている項番は旧版の項番を示します。

目次

第 1 章　概要

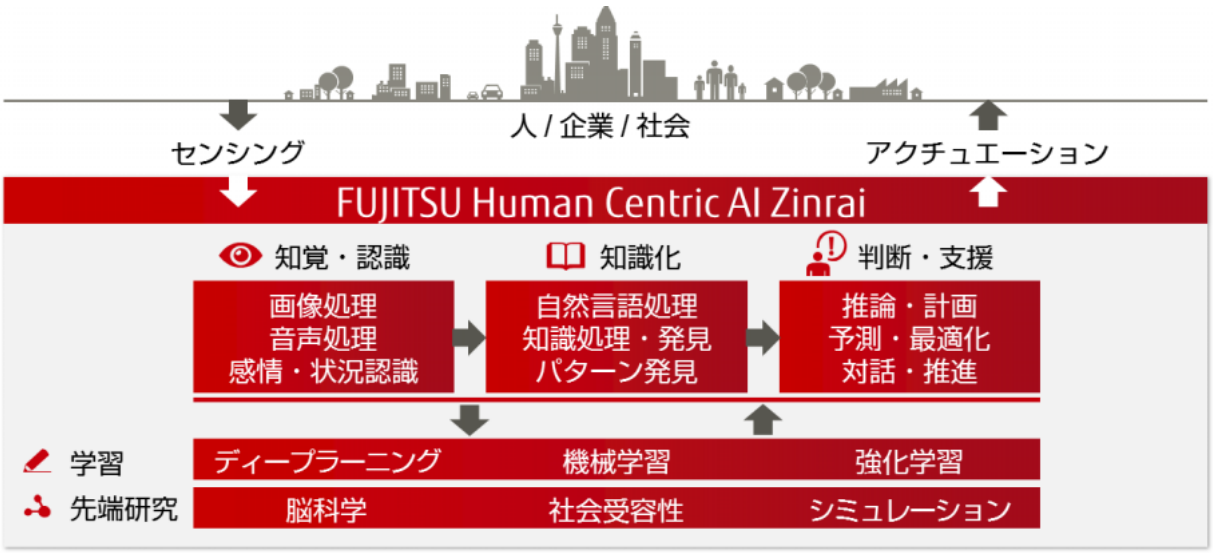
トピック：

この章では、Zinrai ディープラーニング システムの概要について説明します。

1.1 FUJITSU Human Centric AI Zinrai とは

「FUJITSU Human Centric AI Zinrai（ジンライ）」（以下、Zinrai）は、「知覚・認識」、「知識化」、「判断・支援」、およびそれらを高度化し成長させる「学習」などの AI に関する知見や技術を結集し、体系化したものです。

図 1：Zinrai の構成要素



Zinrai は以下の要素で構成されています。

- 知覚・認識
  人のように五感を駆使し、人の感情、気付き、気配りまで処理する感性メディア技術です。
  例えば、振り込め詐欺検知、「人の気持ち理解」による顧客対応サービスの向上と自動化などに活用できます。
- 知識化
  人が理解する知識に加え、さらに機械処理できる知識を創り出す技術です。
  例えば、診療時の意思決定支援、金融監督業務の改善などを進めます。
- 判断・支援
  スーパーコンピュータも活用し、社会やビジネス上の課題を数理的に解決する技術です。
  例えば、空港における混雑緩和、津波浸水シミュレーターなどに利用できます。

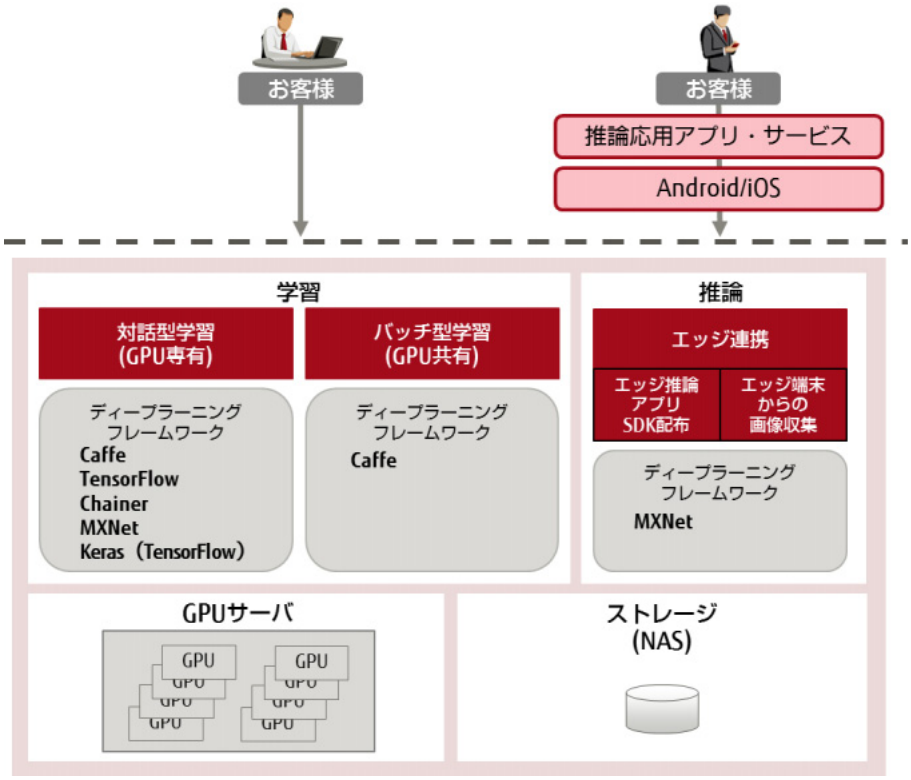
1.2 Zinrai ディープラーニング システムとは

Zinrai ディープラーニング システムは、最新 GPU（Graphics Processing Unit）を実装することで、世界最速クラスの学習処理能力を実現したディープラーニング基盤です。
Zinrai ディープラーニング システムには、以下のような特長があります。

- GPU を使用したディープラーニング専用設計となっており、マルチ GPU 環境で高速学習を行います。
- 操作には、Web ブラウザから簡単に操作できる GUI（Graphical User Interface）を使用するため、ディープラーニング特有のスキルが不要です。

Zinrai ディープラーニング システムは、学習、エッジ端末による推論、およびストレージ（NAS）の機能を提供します。

図 2：Zinrai ディープラーニング システムの機能



学習

ストレージ（NAS）に格納した学習データを使用して、ディープラーニング基盤で学習を行います。

学習の環境には、バッチ型学習および対話型学習があります。

- バッチ型学習
  学習処理をジョブスケジューリングして実行するための環境です。必要なときに GPU を共有して学習を実行します。
  バッチ型学習では、以下の機能を提供します。
  - 学習ウィザード
    Web UI から簡単な操作で学習を実行できます。
  - ディープラーニングフレームワーク
    画像処理のフレームワークで代表的な Caffe を使用します。
  - 学習ユーティリティ
    学習を支援するツールを提供します。
- 対話型学習
  GPU を専有して学習処理を実行するための環境です。
  対話型学習では、以下の機能を提供します。
  - Jupyter を使用した実行環境
  - ディープラーニングフレームワーク
    - Caffe
    - TensorFlow
    - Chainer
    - MXNet
    - Keras（TensorFlow）
  - Time Series Data Analysis（時系列データ解析）
    富士通が開発した、時系列として連続したデータ（以降、時系列データ）のディープラーニング技術を用いて、畳み込みニューラルネットワークを使用した学習と分類を行います。

機能の詳細は、「第 2 章 バッチ型学習」(P.13) および 「第 3 章 対話型学習」(P.32) を参照してください。

推論

学習済みモデルを利用して推論するための環境を、構築および運用します。また、エッジ端末で推論に使用した画像データを、ストレージ（NAS）にアップロードして保存できます。

推論環境として以下を提供します。

- エッジ連携
  Android または iOS を搭載するモバイルデバイスに対応しています。エッジ端末では、オフラインで推論機能を利用できます。
- エッジ端末用の推論アプリケーション開発環境（Software Development Kit（以降、SDK））の配布
  エッジ端末用アプリケーションを作成するための SDK を提供します。
  お客様は SDK を利用し、エッジ推論用アプリケーションを開発できます。
- エッジ端末からの画像収集
  エッジ端末上の推論で使用した画像データを、ストレージ（NAS）へアップロードできます。

エッジ連携機能の詳細は、「第 5 章 エッジ連携」(P.60) を参照してください。

ストレージ（NAS）

ストレージ（NAS）に、Zinrai ディープラーニング システムを利用する際に使用する各種データを集約して、管理します。

ストレージ（NAS）は、全テナント（お客様の管理単位）で共有し、テナントごとのディレクトリで分けて使用します。各種データは、以下の 3 つの領域（ディレクトリ）で管理します。

- 学習用ストレージ（/learning）
  バッチ型学習環境の学習用データ、対話型学習環境の学習用データ、学習済みデータなどを保存／管理するディレクトリです。
- 知識ライブラリ（/knowledge）
  学習用データ、Edge 用データなどを保存／管理するディレクトリです。
- 対話型リポジトリ（/jupyter）
  対話型学習環境で使用するデータ、Docker イメージなどを保存／管理するディレクトリです。

以下の機能を提供します。

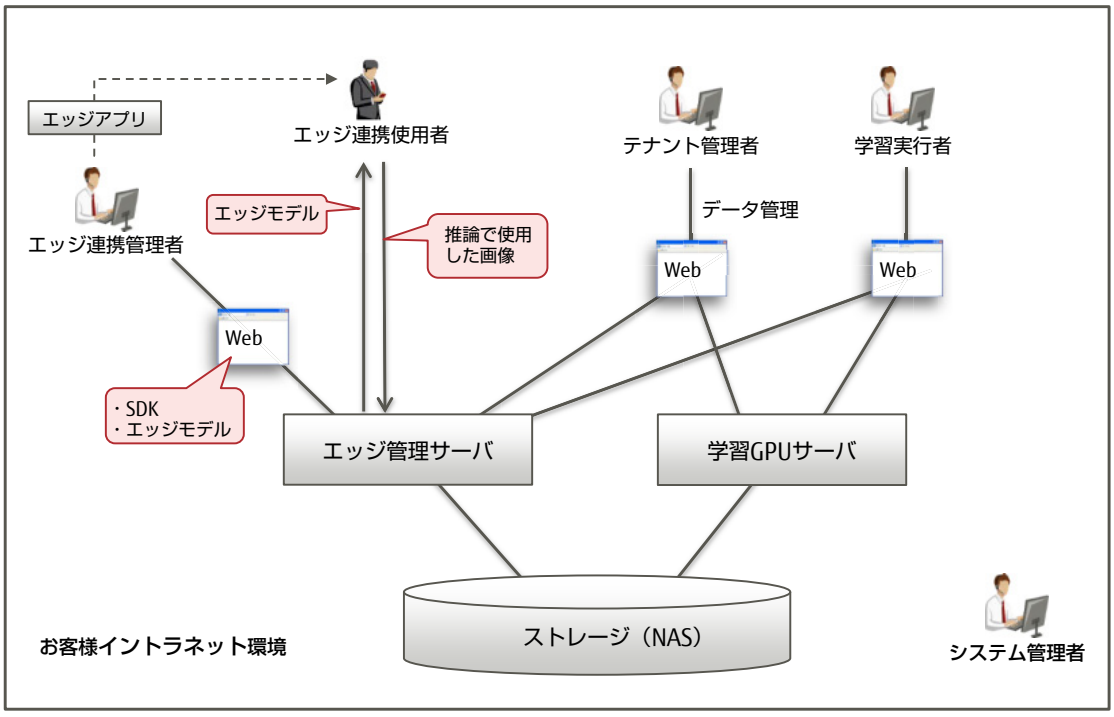
- 画像データ、ニューラルネットワーク、学習済みモデルなどの学習／推論に必要なデータを格納
- ストレージ（NAS）とエッジ端末／学習コンポーネント間のデータ転送機能
- 知識ライブラリに格納したデータの一覧、詳細情報の取得／更新／削除などのデータ管理機能

知識ライブラリの機能の詳細は、「第 6 章 知識ライブラリ」(P.71) を参照してください。

1.2.1　Zinrai ディープラーニング システムの利用形態

以下に、Zinrai ディープラーニング システムの利用形態を担当者別に示します。

図 3：Zinrai ディープラーニング システムの利用形態



第 2 章　バッチ型学習

トピック：

この章では、バッチ型学習を使用した学習の機能概要、機能詳細、注意事項などについて説明します。

2.1 機能概要

バッチ型学習では、Web UI を使用した簡単な操作で学習を実行できます。
以下の機能を提供します。

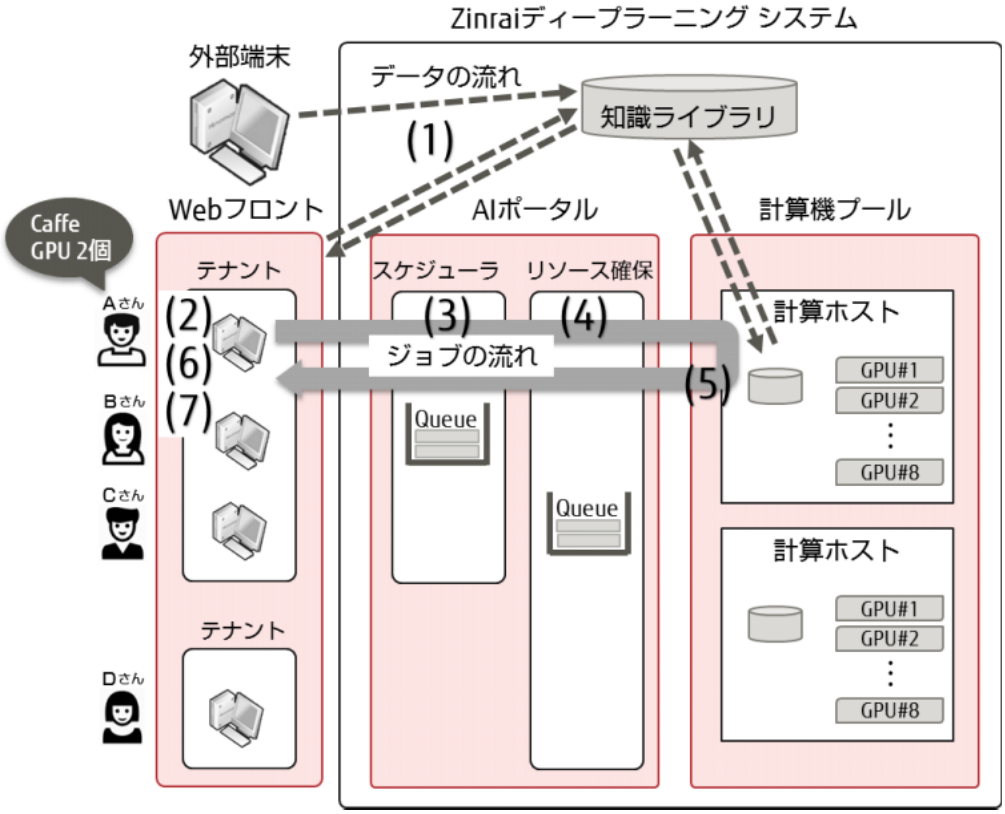
- ディープラーニングフレームワーク
  Caffe を使用しています。
- 学習、認識処理
  学習を実行し、認識結果を確認できます。また、学習中は進捗状況を確認できます。
- 学習ユーティリティ
  学習を支援するユーティリティを提供しています。
  - NN 最適化
    学習済みモデルに NN 最適化を実行すると、システムがハイパーパラメーターのチューニングを開始します。完了すると最適化された学習済みモデルが一覧表示されます。必要な学習済みモデルをダウンロードして使用します。

2.2　機能詳細

2.2.1　操作と処理の流れ

バッチ型環境でのユーザーの操作と処理の流れを以下に示します。

図 4：バッチ型環境の操作と処理の流れ



(1)データ登録	Web UI のデータアップロード機能またはファイル転送機能を使用して、データを登録します。
(2)サブミット	Zinrai ディープラーニング システムにログインし、ジョブを実行します。
(3)ジョブ管理	サブミットされたジョブを FIFO 方式でスケジュール管理します。
(4)リソース確保	ジョブの処理に必要な計算資源を確保します。 計算資源が不足するジョブはキューイングされ、実行待ち状態となります。
(5)ジョブ実行	知識ライブラリ上のデータを入力します。 学習ノード上で処理を実行し、出力データを知識ライブラリに保存します。
(6)ジョブ状態確認	ジョブの状況を確認します（ジョブ実行待ち、ジョブ実行中、ジョブ完了）。
(7)ジョブ結果表示	ジョブの終了コード、解析結果を表示します。

2.2.2　学習

ワーキングセット（ニューラルネットワーク、データセット、学習モデル）を指定して学習を行い、推論を行うための学習済みモデルを作成します。

学習の状況（epoch 回数、accuracy 値（正解率）、loss 値など）を適宜確認できます。

学習で使用するデータ、情報

Zinrai ディープラーニング システムの学習で使用する、以下のデータや設定情報について説明します。

- ユーザーデータ
- データセット
- ワーキングセット
- 学習モデル
- 学習済みモデル

■ ユーザーデータ

ユーザーデータは、Zinrai ディープラーニング システムの学習ユーティリティで作成したデータです。

- 教師データ
- prototxt ファイル（ニューラルネットワーク、学習パラメーターが記述された定義ファイル）

教師データは、学習データ（画像ファイル）を解析するための基準となるデータです。データセット用に指定されたディレクトリに従って、知識ライブラリに格納します。
Solver を定義している solver.prototxt は、Web UI で指定したソルバー情報に従って自動生成されます。直接 solver.prototxt を変更できません。

■ データセット

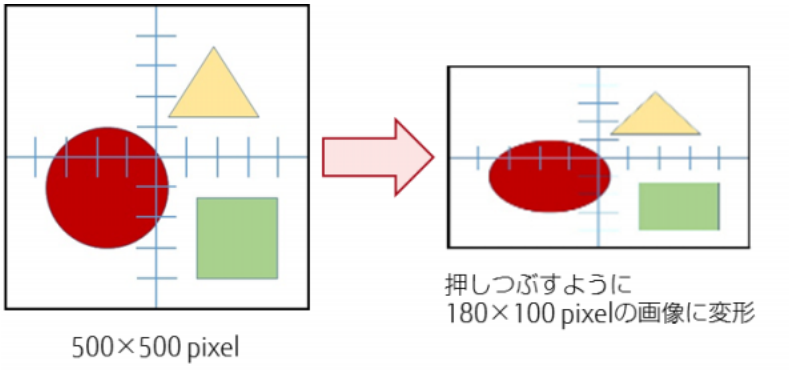
データセットは、指定された構成のディレクトリに以下のデータを格納したものです。

- Caffe に入力するために、教師データをデータベース（LMDB）形式に変換したデータ
- 平均画像データ（Zinrai ディープラーニング システムで学習の入力となる画像の正規化に使用する、全画像の平均を取った画像）

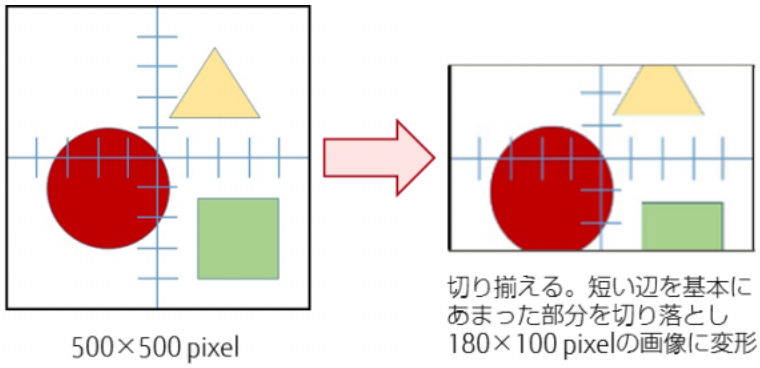
データセットに使用する画像データのサイズは、使用するニューラルネットワーク（prototxt）の種類によって最小サイズに制限があるため、適切なサイズを指定してください。また、元画像の縦横比により変形が必要な場合は、適切なリサイズ（変形）形式を指定してください。

以下に、500×500 pixel の元画像を 180×100 pixel の画像に変形する場合を例に、指定できる変形形式と変形後の画像を示します。

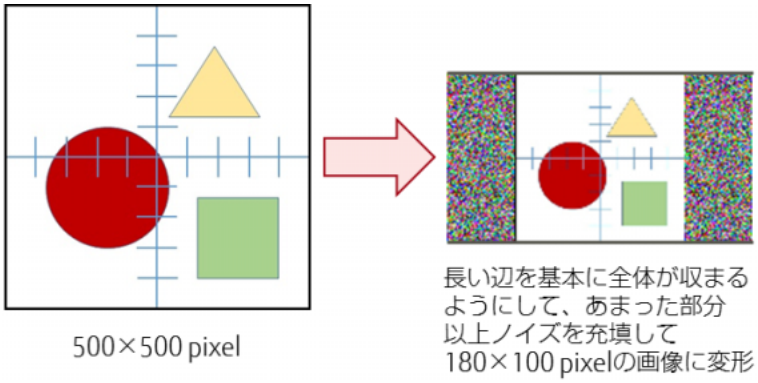
- Squash
  短い辺に合わせて、画像を押しつぶすようにして変形されます。
  縦横比が変わっても元画像の情報すべてを認識させたい場合に、この形式を選択します。



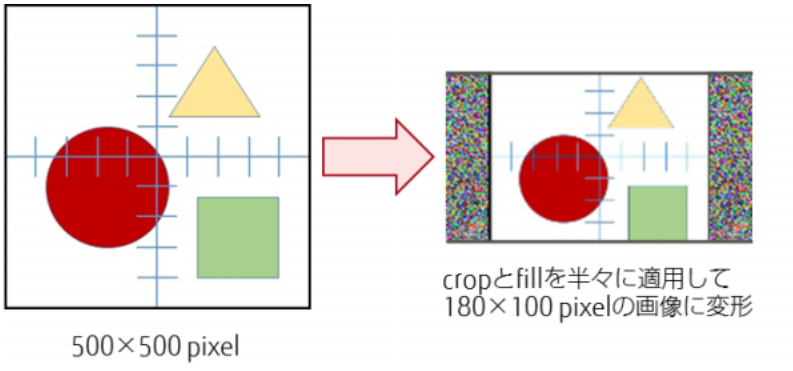
- Crop
  短い辺に合わせて、長い辺が切り落とされます。
  画像の中心付近を重点的に認識させたい場合に、この形式を選択します。



- Fill
  長い辺に合わせて、短い辺の足りない部分がノイズで充填されます。
  元画像の縦横比を保持したまま画像全体を認識させたい場合に、この形式を選択します。



- HalfCropHalfFill
  Crop と Fill が半々に適用された形に変形されます。
  画像の中心付近を重点的に認識させたいが、元画像の縦横比は変えたくない場合に、この形式を選択します。



■ ワーキングセット

学習で使用するニューラルネットワーク、データセット、および学習モデルを組み合わせたものです。学習開始時に必ず指定します。

■ 学習モデル

Zinrai ディープラーニング システムで作成する以下のデータです。

- prototxt ファイル（ニューラルネットワーク、学習パラメーターが記述された定義ファイル）
  Solver を定義している solver.prototxt は、Web UI で指定したソルバー情報に従って自動生成されます。直接 solver.prototxt 上で定義を変更できません。

■ 学習済みモデル

学習が終わったときに出力される、以下の生成物です。

- deploy.prototxt（推論用のニューラルネットワーク）
- caffemodel ファイル（Caffe で作成される、学習パラメーターファイルのスナップショット）

学習の種類

学習には、新規学習、学習再開、および追加学習があります。

■ 新規学習

ワーキングセットを新規に作成して、学習を行います。
学習の結果によっては、accuracy 値（正解率）を向上させるためにニューラルネットワークの編集やハイパーパラメーターの変更を行い、新たに学習します。

■ 学習再開

学習を中断した場合に、スナップショットファイルを使用してニューラルネットワークの状態を復元し、新しい学習ジョブとして学習を再開できます。ハイパーパラメーターは変更せず、中断したときの epoch 回数から再開します。

■ 追加学習

追加学習は、学習済みモデルを追加の画像データで微調整し、学習済みモデルの汎用性を高める場合に行います。

学習済みモデルで使用した画像データと新たな画像データを合わせてデータセットを作成します。
次に、ニューラルネットワークの各レイヤーの学習パラメーターの値をスナップショットファイルの内容で初期化して、新たに学習を始めます。epoch 回数は 0 から開始します。

追加する画像データ（入力イメージデータ）は、事前に 1 つのデータにマージしておく必要があります。

注　意

NN 最適化モデルでは、追加学習はできません。

■ 学習の中断

以下の場合に、学習を中断します。

- モデルを指定して学習を中断したとき
- 指定した最大学習時間内に Training epochs で指定した回数の学習が完了せず、時間切れになったとき

学習を中断すると、中断したときのスナップショットファイルが保存されます。スナップショットファイルを使用して学習を再開できます。

スナップショットファイルについては、「2.2.5 スナップショットファイル」(P.20)を参照してください。

■ 学習の停止

以下の場合に、学習を停止します。

- ジョブを選択して学習を停止したとき
- データセットを指定して、データセットの作成を停止したとき
- 学習中にエラーが発生したとき
- システムの異常などで学習が停止したとき

学習を停止した場合は、停止したときのスナップショットファイルが保存されます。スナップショットファイルを使用して学習を再開できます。

エラー発生、システムの異常などで学習が停止された場合は、停止したときのスナップショットファイルは保存されません。学習を再開するときは、停止する前までのスナップショットファイルを使用してください。

スナップショットファイルについては、「2.2.5 スナップショットファイル」(P.20)を参照してください。

2.2.3　認識

学習済みモデル（caffemodel ファイル）を使用して、学習モデルの accuracy 値（正解率）を確認します。

認識の結果、カテゴリーごとの値（認識率）、各中間層の出力結果（統計情報（分布グラフ）、可視化情報（画像イメージ））が表示されます。

カテゴリーごとの値（認識率）が想定どおりでなかった場合は、ハイパーパラメーターのチューニング、または画像データを追加して、新規学習または追加学習を行ってください。

注　意

NN 最適化モデルでは、認識はできません。

2.2.4　実行ジョブ待合せ時の最大キュー数

バッチ型学習で実行ジョブを待ち合わせる場合の最大キュー数は、1テナント当たり最大6ジョブです。

2.2.5　スナップショットファイル

学習中に作成されるスナップショットファイルについて説明します。

スナップショットファイルとは、Snapshot interval で設定した間隔ごと、または学習が終了するタイミングで作成される学習パラメーターファイルのことで、以下の 2 種類があります。

- solverstate ファイル
  Snapshot interval で設定した間隔ごと、または学習が中断されたタイミングで作成される、Binary proto（Protocol Buffers）形式のスナップショットファイルです。
  学習再開を行う場合に使用します。

  solverstate ファイルのファイル名は、以下です。

  snapshot_iter_*n*.solverstate

  n：スナップショットを作成した時点のイテレーション回数

- caffemodel ファイル
  Snapshot interval で設定した間隔ごと、または学習が終了するタイミングで作成される、Binary proto（Protocol Buffers）形式のスナップショットファイルです。
  推論、追加学習、または学習再開を行う場合に使用します。

  caffemodel ファイルのファイル名は、以下です。

  snapshot_iter_*n*.caffemodel

  n：スナップショットを作成した時点のイテレーション回数

2.3　ディープラーニングフレームワーク

バッチ型学習では、ディープラーニングのフレームワークとして、以下を使用しています。

- Caffe

Caffe の詳細については、Caffe の公式サイトを参照してください。

http://caffe.berkeleyvision.org/

2.4　学習ユーティリティ

バッチ型学習では、以下の学習ユーティリティを提供しています。必要に応じて御利用ください。

- NN 最適化

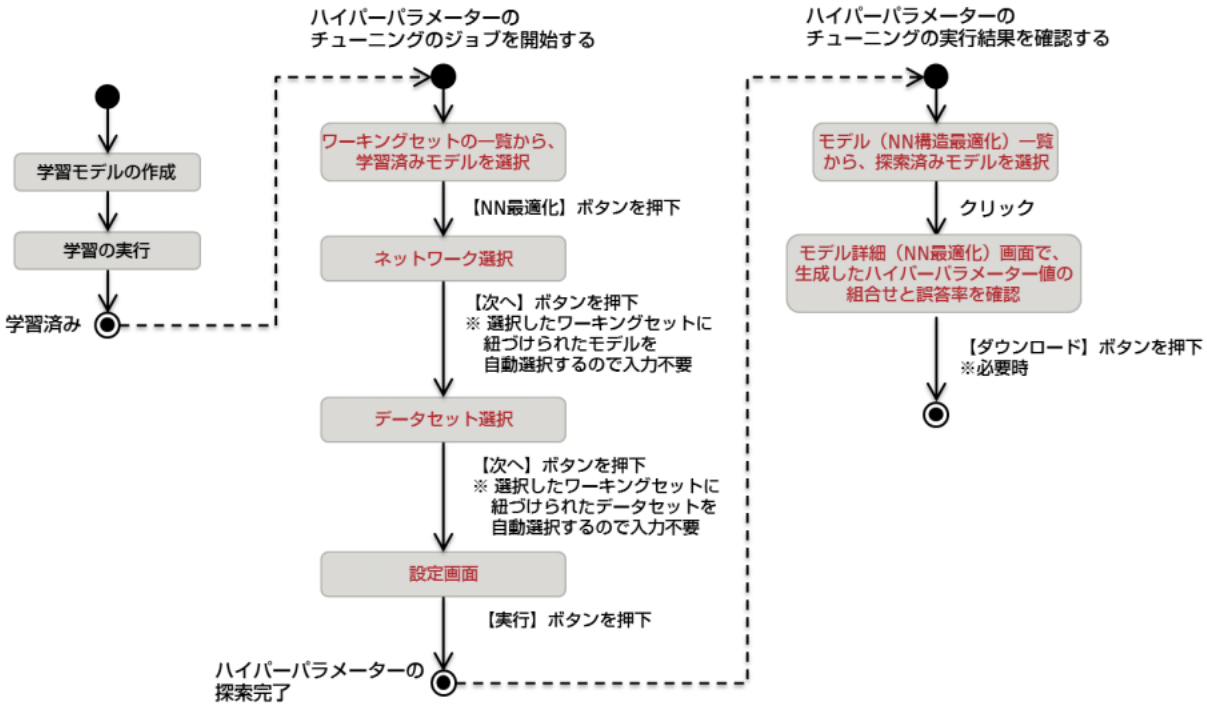
2.4.1　NN 最適化

学習結果の accuracy 値（正解率）や学習速度は、ニューラルネットワーク（NN）で定義したハイパーパラメーターの値やその組合せによって変動します。従来は、学習結果から学習精度に影響を与えた可能性のあるハイパーパラメーターを選び出して値を変更後、学習モデルの作成、学習の実行、学習結果の確認を行い、最適な結果が出るまでこれらの作業を繰り返す必要がありました。

バッチ型学習では、NN 最適化機能を使用して、これらの作業を自動化できます。

以下に、NN 最適化の動作フローを示します。

図 5：NN 最適化の動作フロー



NN 最適化は、以下の手順で行います。
Web UI を使用した操作の詳細については、『ユーザーズガイド』の「ニューラルネットワークを最適化する」を参照してください。

(1)NN 最適化用の学習モデルの作成

最適化したい学習済みモデルのワーキングセットを流用して、NN 最適化用の学習モデルを作成します。

NN 最適化の学習ウィザードに従って、「ワーキングセット情報の入力」、「学習の設定」を設定します。「学習の設定」で、ハイパーパラメーターやジョブキューなどを設定します。

- NN 最適化パラメーター入力
  チューニング候補のレイヤー名およびハイパーパラメーター名を選択して、最小値と最大値を変更します。最大 3 個のハイパーパラメーターをチューニングできます。

- ジョブキュー
  NN 最適化で使用する合計 GPU 数や最大学習試行回数を考慮し、ジョブキューを選択します。

(2)NN 最適化の実行

NN 最適化処理を実行します。
システムで、ハイパーパラメーターの値の組合せを自動生成し、学習を 2 多重で試行します。

注　意

NN 最適化では、手順 (1) で指定したハイパーパラメーターの最小値～最大値の範囲から、任意の数を選択して、ハイパーパラメーターを自動生成します。そのため、学習済みモデルの Caffe の定義内容（train_val.prototxt など）によっては、Caffe がエラー終了する組合せのハイパーパラメーターが生成されることがあります。
このような原因で NN 最適化がエラー終了する場合は、正常終了したものの中から、誤答率の低い組合せ結果を確認してください。
例えば、AlexNet の conv*N*（*N*：数字）のレイヤーは、入力が偶数以外の場合に定義エラーになります。

(3)NN 最適化の実行結果

モデル詳細（NN 最適化）画面の「NN 最適化サマリ」に、実行結果の一覧が表示されます。学習が正常終了した場合は、誤答率およびテストロスが表示されます。
任意の実行結果を選択して、以下のデータを含む tgz ファイル（*1）をダウンロードできます。

*1： tar でアーカイブし、gzip 形式に圧縮したファイル

- mean.binaryproto
  流用元の学習モデルに含まれるファイル
- labels.txt
  流用元の学習モデルに含まれるファイル
- solver.prototxt
  NN 最適化で Caffe 実行時に使用した定義ファイル
  流用元の学習モデルと同じ定義になっています。
- train_val.prototxt
  NN 最適化で Caffe 実行時に使用した定義ファイル
  NN 最適化で動的に生成したハイパーパラメーターに置き換わっています。
- deploy.prototxt
  NN 最適化で Caffe 実行時に使用した定義ファイル
  NN 最適化で動的に生成したハイパーパラメーターに置き換わっています。
- caffelog.log
  NN 最適化で Caffe を実行したときのログ
  Caffe の実行がエラーになった場合も含まれます。
- snapshot_iter_*n*.caffemodel
  NN 最適化で生成したスナップショットファイル
- snapshot_iter_*n*.solverstate
  NN 最適化で生成したスナップショットファイル
- job.log
  NN 最適化のログファイル
  Caffe とは別に NN 最適化の制御で通知する事象（容量不足など）があった場合に含まれます。

手順 (3) でダウンロードしたデータを再度 NN 最適化で使用して、手順 (1) ～手順 (3) を繰り返すことで、テストロスの低いハイパーパラメーターの組合せに近づいていき、学習の精度を高めることができます。

ただし、ある程度の回数を実行すると誤答率は頭打ちになります。このような場合は、一定の epoch 数で NN 最適化を終了させ、新しい学習モデルで NN 最適化をやり直してください。

2.5 制限事項

バッチ型学習で学習する場合の制限事項について説明します。

2.5.1 データベース種別

データセット（教師データのデータベース）は、知識ライブラリに登録されている教師データ群を使用して、利用者が Web UI 上から作成できます。

データベース種別は、LMDB（Lightning Memory-Mapped Database）が指定されます。

2.5.2 Solver 種別、Layer 種別

学習で使用するニューラルネットワーク（prototxt ファイル）には、各層の Layer 種別、各層のサイズや層の配置、および層の数などの情報が含まれています。ハイパーパラメーターは Solver 種別などを設定して変更できます。

2.5.2.1 Solver 種別、Solver パラメーター

バッチ型学習でサポートしている Solver 種別、および設定可能な Solver パラメーターとその他パラメーターを以下に示します。

Solver 種別

サポートしている Solver 種別を以下に示します。

Solver 種別の詳細については、以下の Web サイトを参照してください。

http://caffe.berkeleyvision.org/tutorial/solver.html

表 1：Solver 種別一覧

Solver 種別	GPU 数が 1、2、または 4 の場合
SGD	○
Nesterov (NAG)	○
AdaGrad	○
RMSProp	○
AdaDelta	○
Adam	○

○： サポート

Solver パラメーター、その他パラメーター

設定可能な Solver パラメーター、およびそのほかのパラメーターを以下に示します。

表 2：Solver パラメーター、その他パラメーター一覧

Web UI 上の設定項目	対応する Caffe パラメーター	GPU 数が 1、2、または 4 の場合	備考
Training epochs	max_iter	○	epoch 数を指定してください
Snapshot interval	snapshot	○	epoch 数を指定してください
Validation interval	test_interval	○	epoch 数を指定してください
Random seed	random_seed	○	
Batch size	batch_size	○	1GPU 当たりのバッチサイズを指定してください
Batch Accumulation	iter_size	○	
Solver type	solver_type	○	表 1 のサポートしている Solver 種別を参照してください
Base Learning Rate	base_lr	○	
Policy	lr_policy	○	
Step Size (%)	stepsize	○	
Step Values (%)	stepvalue	○	
Gamma	gamma	○	
Power	power	○	
Crop Size	crop_size	○	
Subtract Mean	mean_file mean_value	○	image を設定：mean_file により Subtract Mean が設定されます pixel を設定：mean_value により Subtract Mean が設定されます

○： 設定可能

○ 備 考

momentum は固定値（0.9）が設定されます。表 2 に記載のない学習パラメーターは自動設定されます。

2.5.2.2 Layer 種別、Layer パラメーター

バッチ型学習でサポートしている Layer 種別、および Layer パラメーターの制限を、以下に示します。

Layer 種別

バッチ型学習でサポートしている Layer 種別を以下に示します。
Layer 種別の詳細については、以下の Web サイトを参照してください。

http://caffe.berkeleyvision.org/tutorial/layers.html

表 3：Layer 種別一覧

Layer 種別	GPU 数が 1、2、または 4 の場合
AbsVal	○
Accuracy	○
ArgMax	○
BNLL	○
BatchNorm	○
BatchReindex	○
Bias	○
Concat	○
ContrastiveLoss	○
Convolution	○
Data	○
Deconvolution	○
Dropout	○
DummyData	×
ELU	○
Eltwise	○
Embed	○
EuclideanLoss	○
Exp	○
Filter	○
Flatten	○
HDF5Data	×
HDF5Output	×
HingeLoss	○
Im2col	○
ImageData	×
InfogainLoss	○

Layer 種別	GPU 数が 1、2、または 4 の場合
InnerProduct	○
LRN	○
Log	○
MVN	○
MemoryData	×
MultinomialLogisticLoss	○
PReLU	○
Pooling	○
Power	○
ReLU	○
Reduction	○
Reshape	○
SPP	○
Scale	○
Sigmoid	○
SigmoidCrossEntropyLoss	○
Silence	○
Slice	○
Softmax	○
SoftmaxWithLoss	○
Split	○
TanH	○
Threshold	○
Tile	○
WindowData	×

○： サポート、×：未サポート

注　意

GPU 数が 1、2、または 4 の場合、サポートしていない以下の Layer 種別が設定されたときは、エラーが表示されて学習を実行できません。サポートされている Layer 種別に指定し直してください。

- DummyData
- HDF5Data
- HDF5Output
- ImageData
- MemoryData
- WindowData

Layer パラメーターの制限

- バッチ型学習では、LayerParameter のニューラルネットワーク定義をサポートします。V0LayerParameter および V1LayerParameter のニューラルネットワーク定義はサポートしていません。

  以下にニューラルネットワーク定義の例を示します。

  - サポートするニューラルネットワーク定義例（LayerParameter）

```
layer {
  name: "innerproduct1"
  type: "InnerProduct"
  inner_product_param {
    num_output: 10
    bias_term: false
    weight_filler {
      type: "gaussian"
      std: 10
    }
  }
  bottom: "data"
  top: "innerproduct1"
}
```

  - サポートしないニューラルネットワーク定義例（V1LayerParameter の場合）

```
layers {                ← layer を使用してください
  name: 'innerproduct1'
  type: INNER_PRODUCT   ← layer type は、enum ではなく文字列を使用してください
  inner_product_param {
    num_output: 10
    bias_term: false
    weight_filler {
      type: 'gaussian'
      std: 10
    }
  }
  bottom: 'data'
  top: 'innerproduct1'
}
```

2.6 注意事項

バッチ型学習を使用する場合の注意事項について説明します。

- 学習ログについて
  学習中は、学習処理の履歴やその情報が学習ログに記録されます。
  学習ログは、Web UI から参照および取得できます。学習ログのファイル名を以下に示します。
  - GPU 数が 1、2、または 4 の場合
    caffelog.log

  学習中に異常が発生した場合は、異常の内容が学習ログに記録されます。担当保守員が調査の目的で学習ログを取得することがあります。
- スナップショットファイルについて
  Snapshot interval を指定する場合は、生成されるスナップショットファイルの容量や格納先の知識ライブラリの空き容量を考慮してください。
  学習中に知識ライブラリの空き容量がなくなると、エラーが発生し、学習が停止します。
  例えば、ニューラルネットワークが AlexNet の場合、1 回のスナップショットファイル（solverstate ファイルおよび caffemodel ファイル）の容量は、500MB 程度です。

  知識ライブラリの容量不足で学習が停止してしまった場合は、知識ライブラリ内の不要なファイルを削除し必要な空き容量を確保して、学習を再開してください。
- ニューラルネットワークの prototxt について
  バッチ型学習で使用するニューラルネットワークの prototxt は、お客様が直接編集できます。
  編集する場合の注意点を、以下に示します。
  - train_val.prototxt
    - Type: “InnerProduct”

注意点	最も出力側にある InnerProduct は、カテゴリー数と紐付く num_output を自動で設定するため、inner_product_param {} にある num_output: nn をコメント化しておくこと

例	`type: "InnerProduct"`
	`…`
	`inner_product_param {`
	`#`
	`# num_output will be set by system automatically`
	`# from category_count of train dataset`
	`#`
	`# num_output: 10`

- Type: "Accuracy"

注意点	• top: には、"accuracy" の文字列を含めること • 最も出力側の type:"Accuracy" だけ、top:"accuracy" とする • 複数の type:"Accuracy" がある場合は、top: は異なる文字列にすること

例	• 最も出力側の Accuracy
	`layer{`
	`  type:"Accuracy"`
	`  …`
	`  **top:"accuracy"**`
	`}`
	
	• それ以外の Accuracy
	`layer{`
	`  type:"Accuracy"`
	`  …`
	`  **top:"loss1/accuracy"**`
	`}`
	
	`layer{`
	`  type:"Accuracy"`
	`  …`
	`  **top:"loss2/accuracy"**`
	`}`

- type: "SoftmaxWithLoss"

注意点	• top: には "loss" の文字列を含めること • top: には "accuracy" の文字列を含んではならない • 複数の type: "SoftmaxWith" がある場合は、top: は異なる文字列にすること

例	`layer {`
	`name: "loss"`
	`type: "SoftmaxWithLoss"`
	`bottom: "ip2"`
	`bottom: "label"`
	`**top: "loss1"**`
	`}`
	
	`layer {`
	`name: "loss"`
	`type: "SoftmaxWithLoss"`
	`bottom: "ip2"`
	`bottom: "label"`
	`**top: "loss2"**`
	`}`

○ 備 考

NN 最適化の場合、Loss レイヤー（SoftmaxWithLoss など）では以下のどちらかを指定する必要があります。

- "loss"（固定）を指定
- "loss1/loss1"（固定）、"loss2/loss1"（固定）、および "loss3/loss3"（固定）を指定

- deploy.prototxt
  - input: 入力層

注意点	Input_shape を用いること（caffe RC3 では、input_param は未サポート）

例	• OK
	input_shape {
	  dim: 1
	  dim: 3
	  dim: 28
	  dim: 28
	}
	
	• NG
	input_param { shape: { dim: 1 dim: 3 dim: 28 dim: 28 }}

  - Type: “InnerProduct”

注意点	最も出力側にある InnerProduct は、カテゴリー数と紐付く num_output を自動で設定するため、inner_product_param {} にある num_output: nn をコメント化しておくこと
例	type: "InnerProduct" … inner_product_param { # # num_output will be set by system automatically # from category_count of train dataset # **# num_output: 10**

第 3 章　対話型学習

トピック：

この章では、対話型学習を使用した学習（推論）の機能概要、機能、注意事項などについて説明します。

3.1 機能概要

対話型環境では、Jupyter 上で Python コードや Python スクリプトを実行して、対話的な学習（推論）ができます。
以下の機能を提供します。

- Docker コンテナ上の Jupyter で Python コードや Python スクリプトを実行して、実行結果を確認しながら学習（推論）のユーザープログラムを作成できます。
  また、ディープラーニング向けの様々なライブラリやツールをインストールして、実行環境を自由にカスタマイズできます。
  万一の実行環境の復旧に備えて、カスタマイズした内容や設定手順は別途記録してください。
- 以下の対話型環境を提供します。

表 4：対話型種別と Python、CUDA、cuDNN のバージョン

対話型種別	Python	CUDA	cuDNN	備考
Base_Python2_cuda9.0_cudnn7.0	2.7.12	9.0	7.0	Jupyter だけの環境に、任意のディープラーニングフレームワークやライブラリをインストールして、学習（推論）を行う
Base_Python3_cuda9.0_cudnn7.0	3.6.5	9.0	7.0	
Base_Python2_cuda9.1_cudnn7.1	2.7.12	9.1	7.1	
Base_Python3_cuda9.1_cudnn7.1	3.6.5	9.1	7.1	
Caffe_Python2_cuda9.0_cudnn7.0	2.7.12	9.0	7.0	Jupyter の ターミナル上で、Caffe を使用して学習（推論）を行う
Caffe_Python3_cuda9.0_cudnn7.0	3.6.5	9.0	7.0	
TensorFlow_Python2_cuda9.0_cudnn7.0	2.7.12	9.0	7.0	Jupyter Notebook 上で、TensorFlow、Keras を使用して学習（推論）を行う また、ONNX を使用して学習モデル形式の変換ができる
TensorFlow_Python3_cuda9.0_cudnn7.0	3.6.5	9.0	7.0	
Chainer_Python2_cuda9.1_cudnn7.1	2.7.12	9.1	7.1	Jupyter Notebook 上で、Chainer を使用して学習（推論）を行う また、ONNX を使用して学習モデル形式の変換ができる
Chainer_Python3_cuda9.1_cudnn7.1	3.6.5	9.1	7.1	
MXNet_Python2_cuda9.1_cudnn7.1	2.7.12	9.1	7.1	Jupyter Notebook 上で、MXNet を使用して学習（推論）を行う また、ONNX を使用して学習モデル形式の変換ができる
MXNet_Python3_cuda9.1_cudnn7.1	3.6.5	9.1	7.1	
TimeSeriesDataAnalysis_Python2	2.7.12	9.1	7.1	Jupyter のターミナル上で、時系列データ（Time Series Data）を解析するための学習（推論）を行う

対話型環境で説明している機能は、OSS（オープンソースソフトウェア）を使用しています。
これらのソフトウェアにはライセンス使用許諾条件があります。『ソフトウェアライセンス使用許諾条件について』を参照してください。

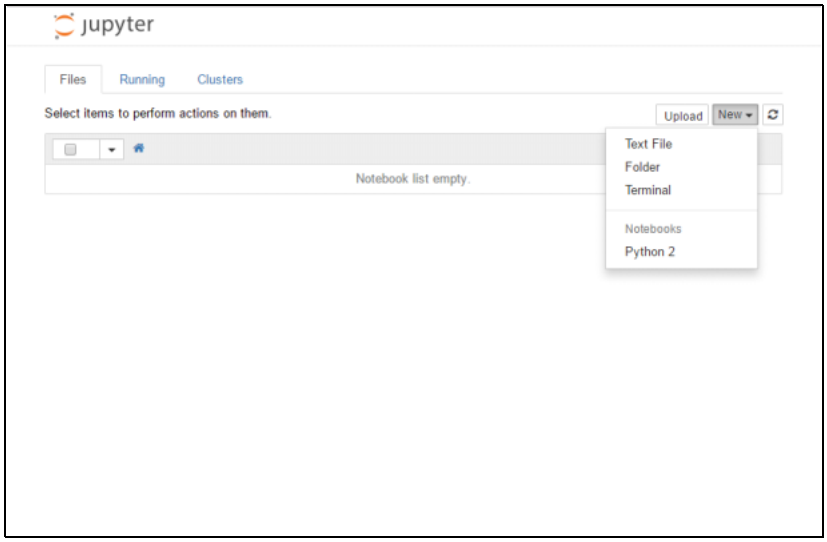
3.2　機能詳細

3.2.1　動作環境

Jupyter

対話型環境で動作する Jupyter のバージョンは、以下の手順で確認できます。

1. 対話型環境（Jupyter）でターミナルを起動します。
   ［New］-［Terminal］を選択します。
   画面例：Python2 系



備 考

対話型環境で使用している Python のバージョンによって、［New］のセレクトボックスの表示が以下のように変わります。
　Python2 の場合：［New］-［Python2］
　Python3 の場合：［New］-［Python3］

2. 以下のコマンドを実行します。

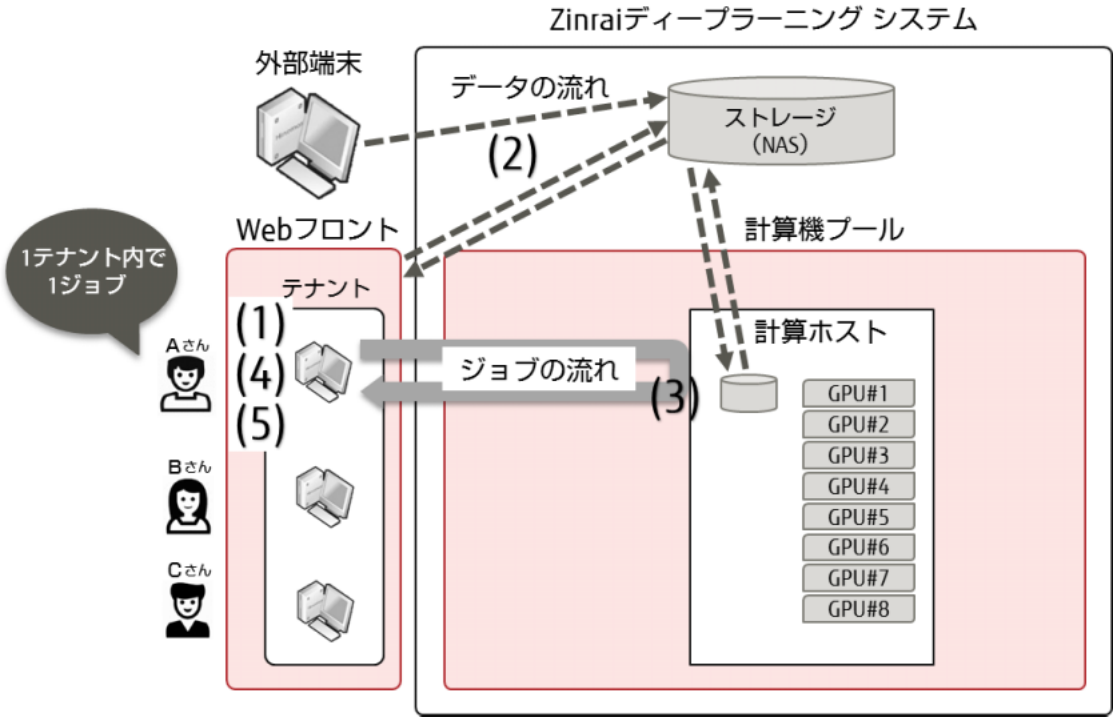
```
# pip list
```

対話型環境にインストールされている Jupyter の情報が表示されます。

3.2.2　操作と処理の流れ

対話型環境でのユーザーの操作と処理の流れを以下に示します。

図 6：対話型環境の操作と処理の流れ



(1)対話型環境起動	Zinrai ディープラーニング システムにログインし、対話型環境を起動します。 対話型環境は、1 テナント内で 1 つだけ起動できます。
(2)データ登録	Web UI のデータアップロード機能およびファイル転送機能を使用して、 データをストレージ（NAS）に登録します。
(3)ジョブ実行	Jupyter 上で実行した Python コードまたは Python スクリプトに従って、 ジョブ（学習（推論））を実行します。
(4)ジョブ結果	解析結果を確認します。
(5)データ保存	出力データを workspace に保存します。

3.2.3　対話型学習の管理

Web UI から対話型学習の管理を行います。
対話型環境の一覧表示、各環境の詳細情報表示、処理の中断／再開、対話型環境の新規作成などができます。

Web UI を使用した操作の詳細については、『ユーザーズガイド』の「対話型学習の管理」を参照してください。

● **対話型一覧**

対話型環境の一覧を表示します。各実行環境のステータスを確認できます。また、名前をクリックすると、詳細情報が表示されます。

● **新規作成**

対話型環境を新規に作成して、起動します。

● **中断**

起動中の対話型環境を中断（停止）します。中断したときの状態が Docker イメージに保存され、中断したときの状態から処理を再開できます。

● **再開**

停止中の対話型環境を再開（再起動）します。

● **削除**

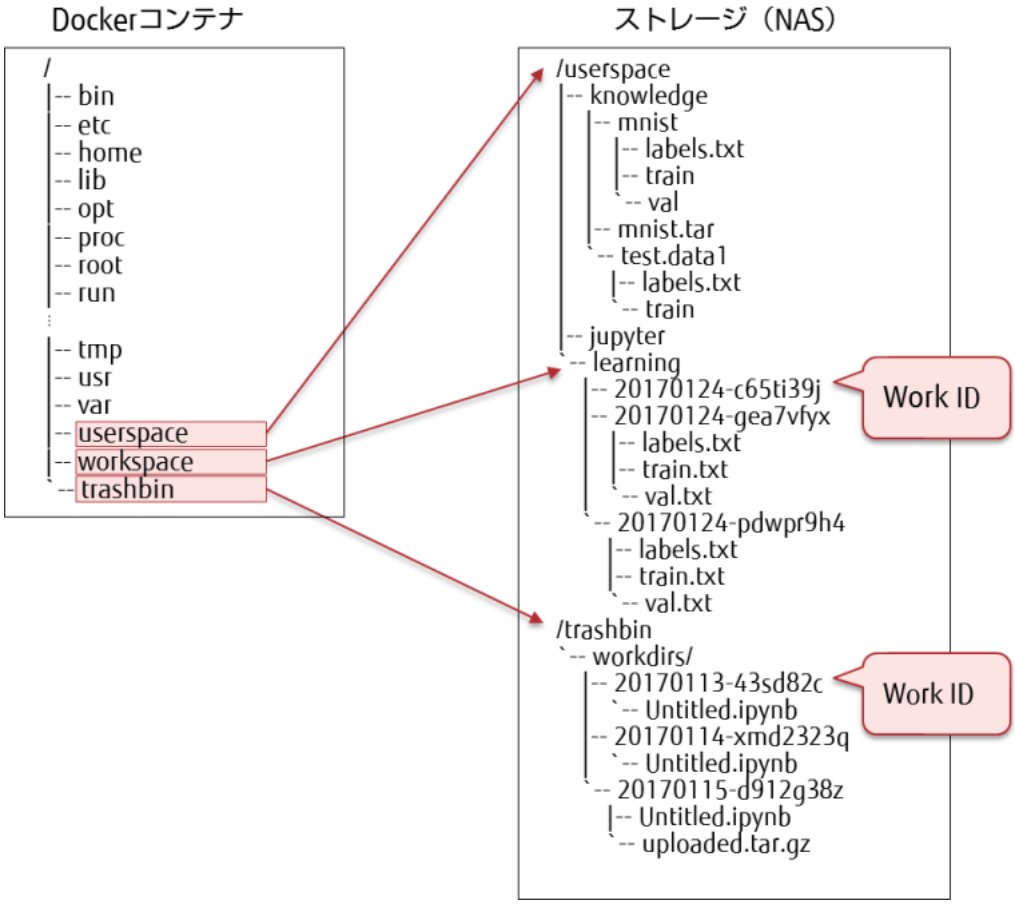
停止中の対話型環境を削除します。削除すると Docker イメージも削除されます。

3.2.4　対話型学習のディレクトリ構成

対話型環境を起動すると、対話型学習で使用するストレージ（NAS）の領域が Docker コンテナ上にマウントされます。
Docker コンテナ上にマウントされる領域とディレクトリ構成について説明します。

- userspace
  読込み専用のユーザー領域で、お客様に割り当てられたストレージ（NAS）のすべてのデータを参照できます。マウントは、処理の中断または再開後も維持されます。
  userspace/learning/*Work ID* は、Work ID に対応する、ほかのジョブや対話型学習が使用している workspace に相当します。
- workspace
  対話型学習で使用する作業領域です。
  workspace に書き込んだデータは、ストレージ（NAS）の学習用ストレージ（learning/*Work ID*）に、保存されます。
- trashbin
  以前に削除した対話型環境がある場合、削除した環境の学習で使用していた workspace を参照および更新できる領域です。
  trashbin/workdirs/*Work ID* の Work ID は、削除した対話型環境の Work ID です。

図 7：対話型学習のディレクトリ構成

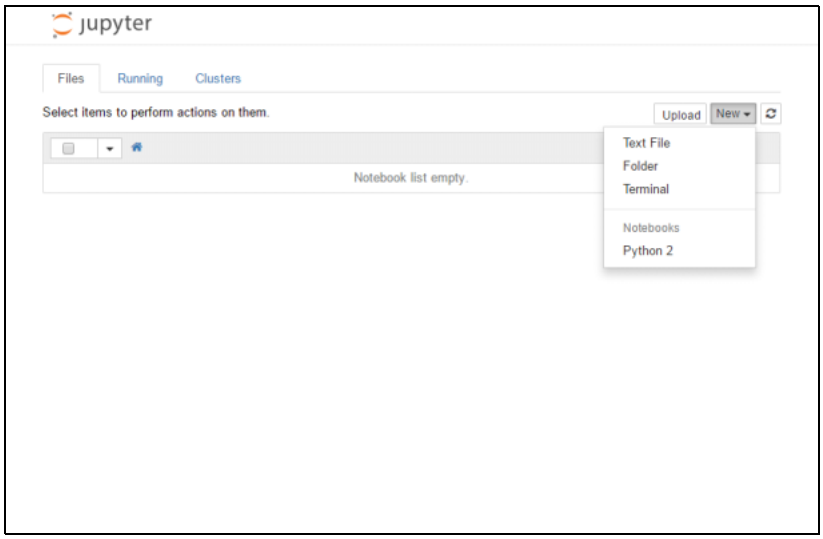


3.2.5　学習で使用するフレームワーク

3.2.5.1 Base

インストール済みの OS パッケージおよび Python パッケージは、以下の手順で確認できます。

1. 対話型環境（Jupyter）でターミナルを起動します。
   ［New］-［Terminal］を選択します。
   画面例：Python2 系



2. インストール済みの OS パッケージを確認するには、dpkg -l コマンドを実行します。

```
# dpkg -l
```

   インストール済みの OS パッケージが一覧表示されます。

3. インストール済みの Python パッケージを確認するには、pip list コマンドを実行します。

```
# pip list
```

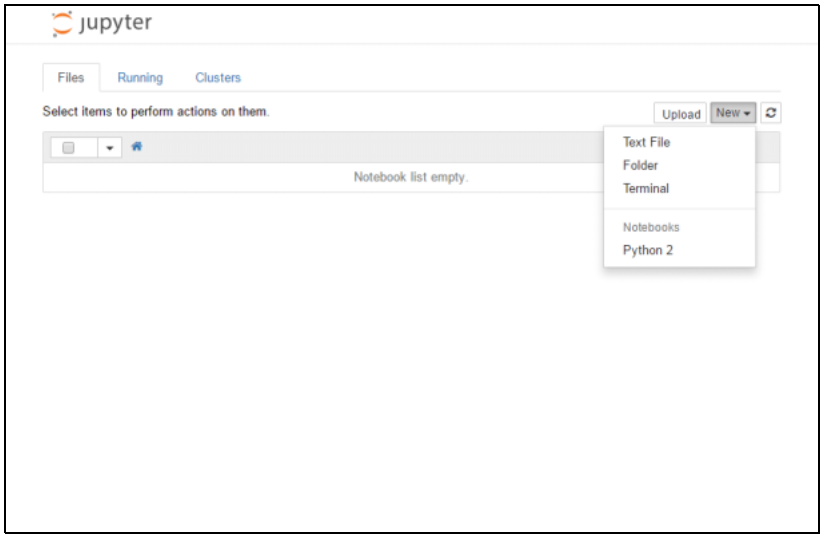
   インストール済みの Python パッケージが一覧表示されます。

3.2.5.2 Caffe

Caffe の詳細については、Caffe の公式サイトを参照してください。
http://caffe.berkeleyvision.org/

- インストール済みパッケージの確認方法
  確認方法については、「3.2.5.1 Base」(P.38) を参照してください。
- Caffe の版数の確認方法
  インストールされている Caffe の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  画面例：Python2 系



  (2) 以下のコマンドを実行します。

```
# export PATH="$PATH:/opt/caffe/.build_release/tools"
# caffe --version
```

  インストールされている Caffe の版数が表示されます。

3.2.5.3 TensorFlow

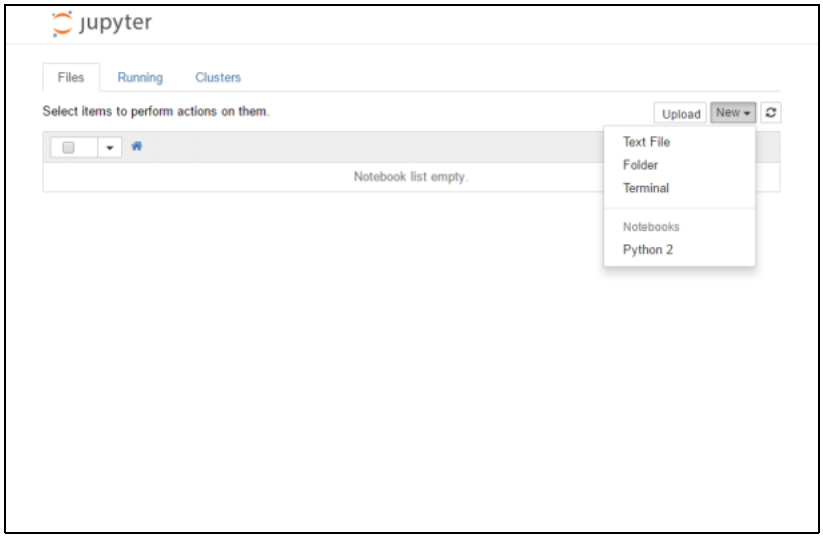
対話型種別「TensorFlow_Python2_cuda9.0_cudnn7.0」、「TensorFlow_Python3_cuda9.0_cudnn7.0」は、フレームワーク TensorFlow と Keras が使用できます。

TensorFlow の詳細については、TensorFlow の公式サイトを参照してください。
https://www.tensorflow.org/

Keras の詳細については、Keras の公式サイトを参照してください。
https://keras.io/ja/

- インストール済みパッケージの確認方法
  確認方法については、「3.2.5.1 Base」(P.38) を参照してください。
- TensorFlow の版数の確認方法
  インストールされている TensorFlow の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  画面例：Python2 系



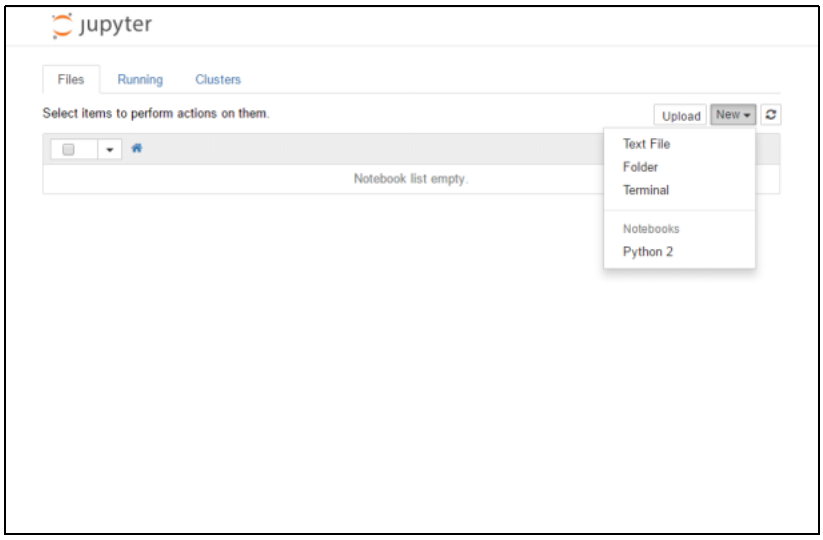
  (2) 以下のコマンドを実行します。

```
# pip show tensorflow-gpu
```

  インストールされている TensorFlow の情報が表示されます。

- Keras の版数の確認方法
  インストールされている Keras の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  画面例：対話型種別「TensorFlow_Python2_cuda9.0_cudnn7.0」



  (2) 以下のコマンドを実行します。

```
# pip show Keras
```

  インストールされている Keras の情報が表示されます。

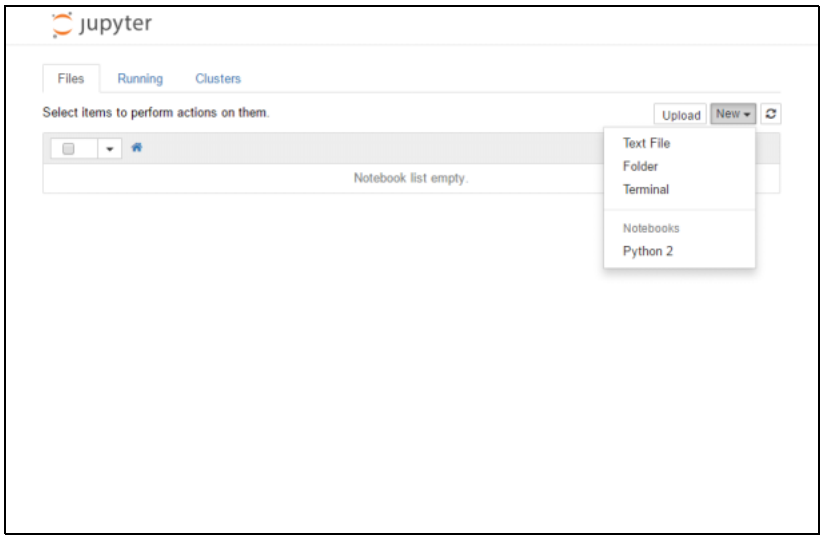
3.2.5.4 Chainer

Chainer は、Define-by-Run という仕組みを採用しており、学習しながら CNN を動的に構築／更新できます。
Chainer の詳細については、Chainer の公式サイトを参照してください。

http://chainer.org/

- インストール済みパッケージの確認方法
  確認方法については、「3.2.5.1 Base」(P.38) を参照してください。
- Chainer の版数の確認方法
  インストールされている Chainer の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  画面例：Python2 系



  (2) 以下のコマンドを実行します。

```
# pip show chainer
```

  インストールされている Chainer の情報が表示されます。

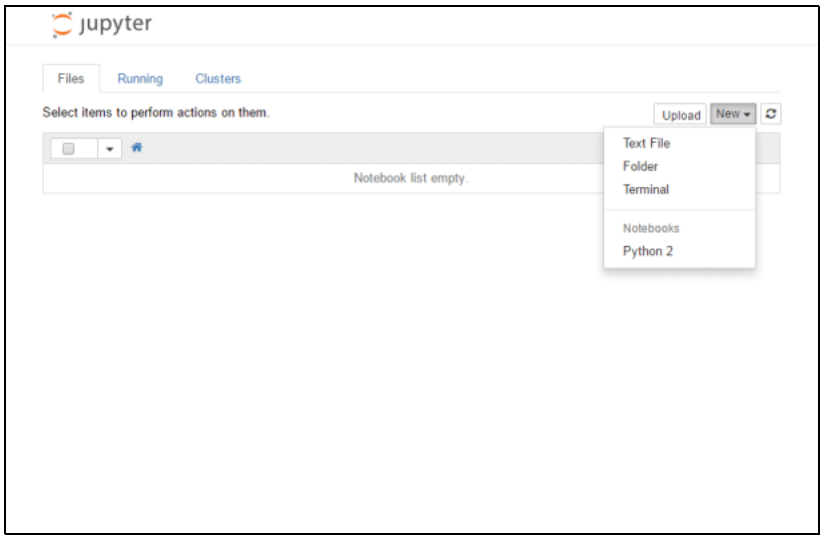
3.2.5.5 MXNet

MXNet の詳細については、MXNet の公式サイトを参照してください。

https://mxnet.apache.org/

- インストール済みパッケージの確認方法
  確認方法については、「3.2.5.1 Base」(P.38) を参照してください。
- MXNet の版数の確認方法
  インストールされている MXNet の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  画面例：Python2 系



  (2) 以下のコマンドを実行します。

```
# pip show mxnet
```

  インストールされている MXNet の情報が表示されます。

3.2.5.6 Time Series Data Analysis（時系列データ解析）

対話型種別に「TimeSeriesDataAnalysis_Python2」を選択して、時系列データ（Time Series Data）を解析するための学習および推論（分類）を行います。

富士通が開発した時系列データ解析のディープラーニング技術を採用しており、時系列データを高精度に自動分類できます。

時系列データの学習および推論は、以下の Python スクリプトを使用して実行します。

学習／テスト用：train.py
推論用：predict.py

また、UCI のサンプルデータを使用して、時系列データの学習および推論を試行できます。試行は、以下の Python スクリプトを使用して実行します。

UCI サンプルデータ学習／テスト用：train_uci.py
UCI サンプルデータ推論用　　：predict_uci.py

3.2.5.6.1 時系列データ解析について

■ 開発の背景

近年、AI（人工知能）の中核技術である機械学習の分野において、人がルールを教えることなく、自動的に物事を解釈／判断するための特徴要素を抽出できるディープラーニング技術が注目されています。
特に IoT（Internet of Things）時代において、機器からの膨大な量の時系列データが蓄積されるため、それらにディープラーニングを適用して高精度に分類することで、さらなる分析が可能となり、新たな価値の創造やビジネス領域の開拓につながることが期待されています。

■ 課題

ディープラーニングは機械学習の強力な手法であり、人工知能発展のブレークスルーとして注目されていますが、現状では画像や音声の認識など限られた種類のデータにしか有効に適用できていません。
特に、IoT 機器などに搭載されているセンサーから取得される変動の激しい複雑な時系列データは、ディープラーニングだけでなく、ほかの機械学習技術でも高精度な分類を実現することは困難でした。

■ 開発した技術

富士通では、最先端のカオス理論および位相幾何学を活用し、時系列データを高精度に自動で分類することができるディープラーニング技術を開発しました。この技術により、変動の激しい複雑な時系列データも扱えるようになりました。

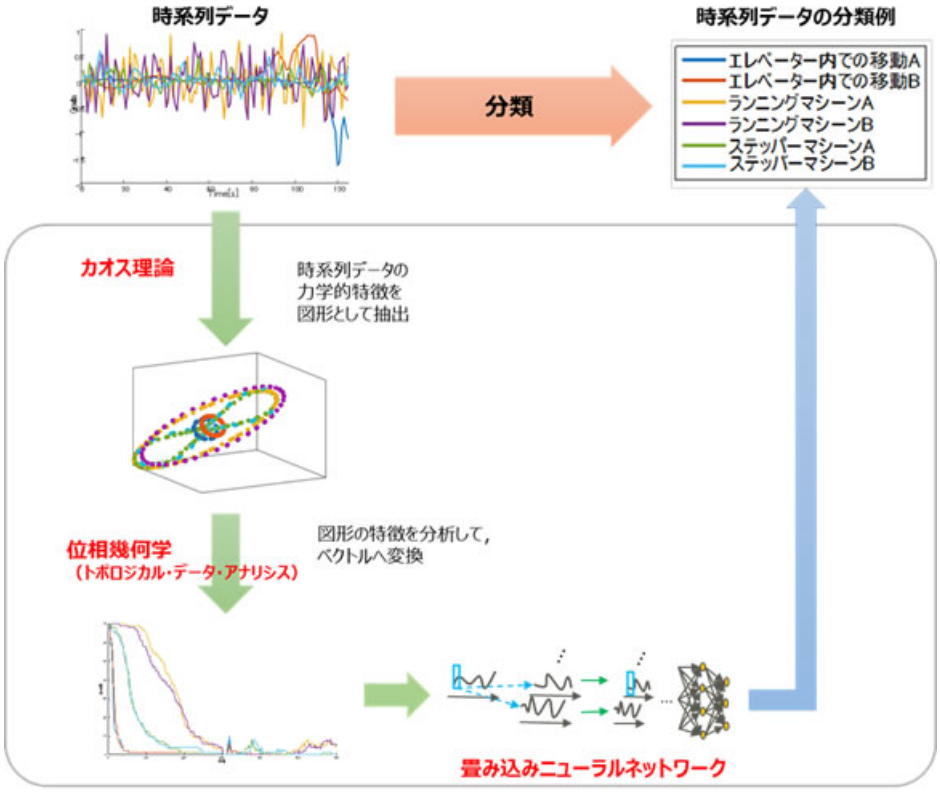
このディープラーニング技術では、以下の手順で学習し、時系列データを分類します。

1. カオス理論に基づいた、時系列データの図形化
   センサーにより観測される数値は、力学的な運動が複雑に組み合わされた結果として、表面的に現れます。この力学的な運動の仕組みを直接調べることは困難ですが、カオス理論に基づいてデータの時間変化をグラフ上にプロットしていくと、運動の仕組みごとに特徴的な軌跡を描くことが知られています。この図形化手法を用いることで、対応する時系列データを図形として区別することが可能となります。
2. 位相幾何学に基づいた、図形の数値化
   1. で得られた図形のまま機械学習を行うことは困難なため、位相幾何学に基づくデータ分析手法の 1 つであるトポロジカル・データ・アナリシス（*1）を用いて、図形の特徴を数値化します。この手法では、一般的にイメージされる図形としての特徴ではなく、図形に含まれる穴の数や、大まかな形状を特徴として分析し、独自のベクトル表現に変換します。

*1： データをある空間内に配置された点の集合とみなし、その集合の幾何的な情報を抽出するデータ分析手法のこと

3. 畳み込みニューラルネットワーク（CNN）による学習と分類
2. で得られた独自のベクトル表現を学習する CNN を新たに設計しました。この CNN を用いて学習することで、複雑な時系列データの分類が可能となります。

図 8：ディープラーニング技術による時系列データ分類イメージ解析



■ 効果

ウェアラブル機器に搭載されたジャイロセンサーの時系列データを基に、人の運動行動の分類を行う UC Irvine Machine Learning Repository（*2）のベンチマークテストにおいて、既存手法に比べて約 25%精度が向上し、約 85%の精度を達成しました。また、脳波の時系列データからの状態推論を行う分類においては、既存手法に比べて約 20%精度が向上し、約 77%の精度を達成しました。

本技術により、ディープラーニング技術の適用できるデータの範囲が時系列データにまで広がります。さらに、人による判別が困難な変動の激しい時系列データを高精度に分類できるようになることで、新たな分析が可能になります。
以下の分野での適用が期待されます。

- 行動分類／認識
- 疾病診断：心拍や脳波などで体調や病気の診断
- 介護補助：要介護者の状態測定
- 故障診断：機械の故障状態の診断
- 経済予測：株価の予測

*2： 機械学習の比較評価用に多数のデータセットを提供している、世界的に有名なリポジトリ

3.2.5.6.2 サンプルデータを使用した時系列データ解析

UCI Machine Learning Repository で公開されている、UCI サンプルデータ（Daily and Sports Activities Data Set）を使用して、時系列データの学習／テストおよび推論を試行できます。

注　意

UCI サンプルデータを使用した論文発表などを行う場合は、以下の Web サイトの引用ポリシー／リクエストに従ってください。

- http://archive.ics.uci.edu/ml/citation_policy.html
- http://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities#

UCI サンプルデータを使用した学習／テストおよび推論の試行手順については、『ユーザーズガイド』の「サンプルデータを使用した時系列データ解析」を参照してください。

3.2.5.6.3 時系列データ解析環境のディレクトリ構成

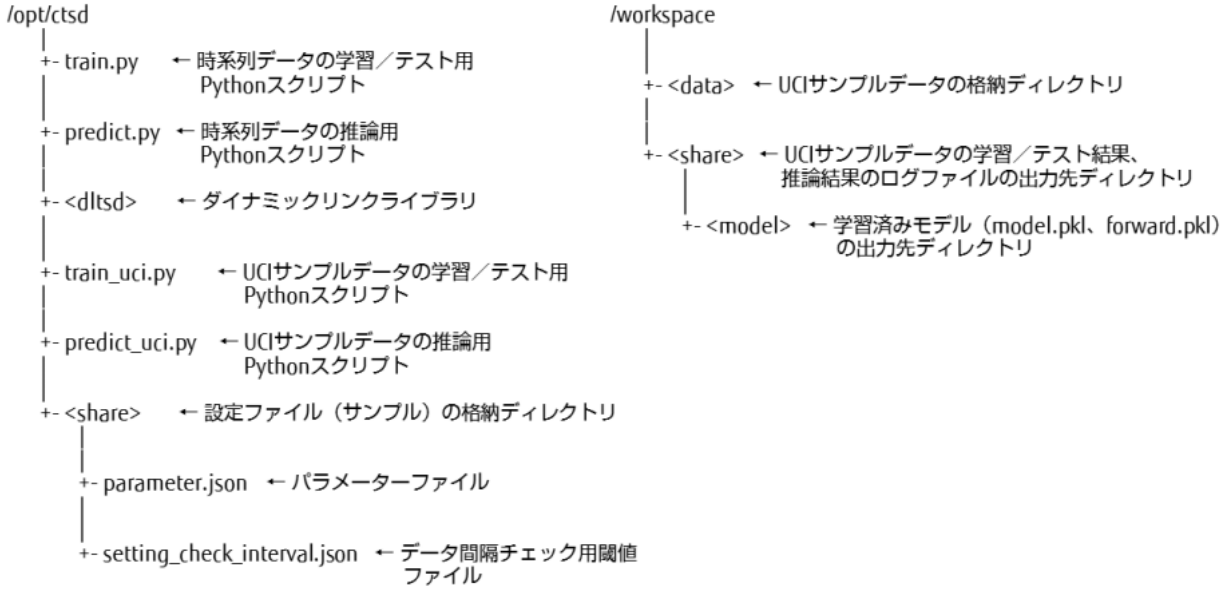
時系列データ解析環境のディレクトリ構成について説明します。

opt/ctsd配下には、各Pythonスクリプトと関連するライブラリ、設定ファイルなどが登録されています。

workspace 配下には、以下のデータを格納することを推奨します。

- UCI サンプルデータセット
- 学習／テストおよび推論で使用する時系列データセット
- 学習／テストおよび推論で使用する設定ファイル
- 学習／テスト結果のワーカーログファイル
- 学習済みモデル
- 推論結果のワーカーログファイル

図 9：時系列データ解析環境のディレクトリ構成



3.2.5.6.4 時系列データ解析で使用するデータの準備

事前に、以下のデータを準備してください。

- 学習用データ、テスト用データ、および推論用データ
- 設定ファイル
  - データ間隔チェック用閾値ファイル
  - パラメーターファイル

学習用、テスト用、および推論用データ

機器から取得した時系列データを、学習用、テスト用、および推論用のデータに変換し、時系列データセットとして、workspace 配下のディレクトリに格納します。

■ 時系列データセットの形式

ディレクトリに格納する時系列データセットは、以下の形式に従ってください。

- ファイル名の命名ルール
  - 拡張子は .csv
  - ファイル名にアンダースコア（_）は指定できない
  - 1 つのディレクトリ配下のファイル名は一意になるように命名する

  例：学習用データのファイル名を開始時刻とし、学習用の train ディレクトリに格納する場合

  ```
  指定ディレクトリ /train/20160916-120000-000.csv
  ```

- ファイルの内容
  以下の情報を、カンマ区切りで指定します。

  ```
  ID, 正解ラベル , 系列 , 時刻（省略可能）, データ
  ```

  - ID
    学習 1 組のデータを表すフィールドです。半角英数字で指定します。
    ファイル内の学習 1 組のデータについて一意である必要があります。
  - 正解ラベル
    データの正解ラベルを設定するフィールドです。半角英数字で指定します。推論に使用する時系列データセットの正解ラベルは無視されます。
  - 系列
    データ系列の括りを表すフィールドです。半角英数字で指定します。
  - 時刻（省略可能）
    UTC 形式の時刻フィールドです。学習では使用しない情報ですが、入力データが一定間隔のデータかを判定する場合に使用します。
  - データ
    一定周期に同じタイミングで測定された値を指定するフィールドです。数値を指定します。

- 例
  系列 0 と系列 1 の 2 組の時系列データで、ID に 1（正解ラベル =0）と 2（正解ラベル =1）が格納されている場合

```
#       1,0,0,2016-09-02T13:00:00.000,32.1
#       1,0,0,2016-09-02T13:00:01.000,32.4
#       1,0,0,2016-09-02T13:00:02.000,32.5
#       1,0,0,2016-09-02T13:00:03.000,32.2
#       1,0,0,2016-09-02T13:00:04.000,31.9
#       1,0,0,2016-09-02T13:00:05.000,31.5
#       1,0,0,2016-09-02T13:00:06.000,31.5
#       1,0,0,2016-09-02T13:00:07.000,31.3
#       1,0,0,2016-09-02T13:00:08.000,31.4
#       1,0,0,2016-09-02T13:00:09.000,31.9
#       1,0,0,2016-09-02T13:00:10.000,32.1
#       1,0,1,2016-09-02T13:00:00.000,78.1
#       1,0,1,2016-09-02T13:00:05.000,78.1
#       1,0,1,2016-09-02T13:00:10.000,78.1
#       2,1,0,2016-09-02T13:00:11.000,32.1
#       2,1,0,2016-09-02T13:00:12.000,32.4
#       2,1,0,2016-09-02T13:00:13.000,32.5
#       2,1,0,2016-09-02T13:00:14.000,32.2
#       2,1,0,2016-09-02T13:00:15.000,31.9
#       2,1,0,2016-09-02T13:00:16.000,31.5
#       2,1,0,2016-09-02T13:00:17.000,31.5
#       2,1,0,2016-09-02T13:00:18.000,31.3
#       2,1,0,2016-09-02T13:00:19.000,31.4
#       2,1,0,2016-09-02T13:00:20.000,31.9
#       2,1,1,2016-09-02T13:00:15.000,78.1
#       2,1,1,2016-09-02T13:00:20.000,78.1
```

■ 時系列データセットのディレクトリ構成

時系列データセットは、以下のディレクトリ構成に従って格納してください。

- 学習に使用する時系列データセットの場合
  workspace 配下に任意のディレクトリを作成し、その配下に train ディレクトリと test ディレクトリを作成します。
  学習用データを train ディレクトリに、テスト用データを test ディレクトリに格納してください。

  図 10：時系列データセット（学習用、テスト用）のディレクトリ構成

```
.
|-- train
|   `-- aaa.csv
`-- test
    `-- aaa.csv
```

- 推論に使用する時系列データセットの場合
  workspace 配下に任意のディレクトリを作成し、その配下に test ディレクトリを作成します。
  推論用データを test ディレクトリに格納してください。

  図 11：時系列データセット（推論用）のディレクトリ構成

```
.
|-- test
    `-- aaa.csv
```

注　意

学習／テストおよび推論の実行時には、train ディレクトリおよび test ディレクトリ配下に格納されているすべての csv ファイルを探索して処理します。そのため、処理する必要のない csv ファイルは格納しないでください

設定ファイル

以下の設定ファイルを作成して、workspace 配下のディレクトリに格納します。

- データ間隔チェック用閾値ファイル
  時系列データのデータ取得機器に関する情報を設定します。
- パラメーターファイル
  ネットワーク（CNN）構造や学習に関するパラメーターを設定します。

■ 設定ファイルの形式

- データ間隔チェック用閾値ファイル（json）
  /opt/ctsd/share 配下に格納されている、サンプルファイル（setting_check_interval.json）を基に作成します。

  系列ごとに、データ間隔と有効誤差を以下の形式で指定してください。

```
{
    系列 : [ データ間隔 , 有効誤差 ],
    系列 : [ データ間隔 , 有効誤差 ],
     . . .
    系列 : [ データ間隔 , 有効誤差 ]
}
```

  - データ間隔
    データ取得機器のデータサンプリング間隔（ミリ秒）
  - 有効誤差
    データ取得機器のデータサンプリングタイミングの誤差（± ミリ秒）

  データ間隔に「0」を指定した場合、または入力データファイルで時刻が省略されている場合は、データ間隔はチェックされません。

  以下に指定例を示します。

```
{ "0":[1000,100],"1":[5000,200] }
```

- パラメーターファイル（json）
  /opt/ctsd/share 配下に格納されている、サンプルファイル（parameter.json）を基に作成します。

各パラメーターを以下の形式で指定してください。

```
{
          "iteration":1,
          "parameters":[
                              {
                                                  "name":"nSheet00",
                                                  "value":"6"                (1)
                              },
                              {
                                                  "name":"nSheet01",
                                                  "value":"6"                (2)
                              },
                              {
                                                  "name":"nSheet02",
                                                  "value":"1"                (3)
                              },
                              {
                                                  "name":"nFilter00",
                                                  "value":"6"                (4)
                              },
                              {
                                                  "name":"nFilter01",
                                                  "value":"5"                (5)
                              },
                              {
                                                  "name":"nFilter02",
                                                  "value":"1"                (6)
                              },
                              {
                                                  "name":"nPooling00",
                                                  "value":"5"                (7)
                              },
                              {
                                                  "name":"nPooling01",
                                                  "value":"5"                (8)
                              },
                              {
                                                  "name":"nPooling02",
                                                  "value":"1"                (9)
                              },
                              {
                                                  "name":"addLayer_flag",
                                                  "value":"0"                (10)
                              },
                              {
                                                  "name":"gpu_flag",
                                                  "value":"0"                (11)
                              },
                              {
                                                  "name":"n_epoch",
                                                  "value":"50"               (12)
                              },
                              {
                                                  "name":"batchsize_div",
                                                  "value":"10"               (13)
                              }
               ]
}
```

(1)	nSheet00	第 1 層目の畳み込み層の出力枚数
(2)	nSheet01	第 2 層目の畳み込み層の出力枚数
(3)	nSheet02	第 3 層目の畳み込み層の出力枚数 (10) の addLayer_flag で「0」を指定している場合、設定は無効になる
(4)	nFilter00	第 1 層目の畳み込み層のフィルターサイズ
(5)	nFilter01	第 2 層目の畳み込み層のフィルターサイズ
(6)	nFilter02	第 3 層目の畳み込み層のフィルターサイズ (10) の addLayer_flag で「0」を指定している場合、設定は無効になる
(7)	nPooling00	第 1 層目の畳み込み層のプーリングサイズ
(8)	nPooling01	第 2 層目の畳み込み層のプーリングサイズ
(9)	nPooling02	第 3 層目の畳み込み層のプーリングサイズ (10) の addLayer_flag で「0」を指定している場合、設定は無効になる
(10)	addLayer_flag	畳み込み層を指定する 「0」の場合：2 層としてモデルを学習 「1」の場合：3 層としてモデルを学習
(11)	gpu_flag	GPU 使用フラグ 「-1」の場合：使用しない 「0」の場合：使用する（1GPU 使用） 「-1」、「0」以外は設定しない
(12)	n_epoch	学習回数（epoch 回数）
(13)	batchsize_div	ミニバッチサイズ（学習するデータをひとまとめにする単位）を算出するための除数

■ 設定ファイルの格納先

workspace 配下に任意のディレクトリを作成し、データ間隔チェック用閾値ファイル（setting_check_interval.json）とパラメーターファイル（parameter.json）を格納します。

3.2.5.6.5 学習／テスト

train.py スクリプトを実行して、時系列データを学習／テストします。
学習／テストの手順、および train.py スクリプトの詳細については、『ユーザーズガイド』の「時系列データ解析」を参照してください。

学習およびテストが終了すると、以下のファイルが出力されます。

- ワーカーログファイル
  train.py スクリプト実行時に指定したパスに、ワーカーログファイル（.log）が出力されます。
  ワーカーログファイルで、以下を確認できます。
  - 1epoch ごとの学習結果（train mean）とテスト結果（test mean）
    それぞれの loss 値と accuracy 値（正解率）が表示されます。
  - 学習済みモデルの出力先
    学習終了後に生成された学習済みモデル（model.pkl、forward.pkl）の出力先パスが表示されます。
- 学習済みモデル
  train.py スクリプト実行時に指定したディレクトリ配下にパラメーターファイル（parameter.json）で指定したパラメーター値に対応したディレクトリが作成され、以下のバイナリファイルが出力されます。
  学習済みモデルに関する情報が設定されています。
  - モデルファイル（model.pkl）
  - フォワードファイル（forword.pkl）
- 正解ラベル管理ファイル
  train.py スクリプト実行時に指定した学習データの格納ディレクトリ名に対応したディレクトリが作成され、正解ラベル管理ファイル（correct_label_mng.json）が出力されます。

  学習データ格納ディレクトリ名 *_dl*/correct_label_mng.json

注　意

学習の精度（loss 値や accuracy 値）が想定どおりでなかった場合は、test loss を確認して過学習がおきていないことを確認し、パラメーターファイルの学習回数（epoch 回数）を増やして、再度学習してください。精度向上が頭打ちになった場合は、パラメーターファイルの畳み込み層の設定を変更してください。

3.2.5.6.6 推論

predict.py スクリプトを実行して、時系列データを推論します。
推論の手順、および predict.py スクリプトの詳細については、『ユーザーズガイド』の「時系列データ解析」を参照してください。

推論が終了すると、predict.py スクリプト実行時に指定したパスに、ワーカーログファイル（.log）が出力されます。ワーカーログファイルで、推論に使用したデータごとの推論結果（スコア）を確認できます。

3.2.6　学習モデル形式変換について

3.2.6.1　TensorFlow

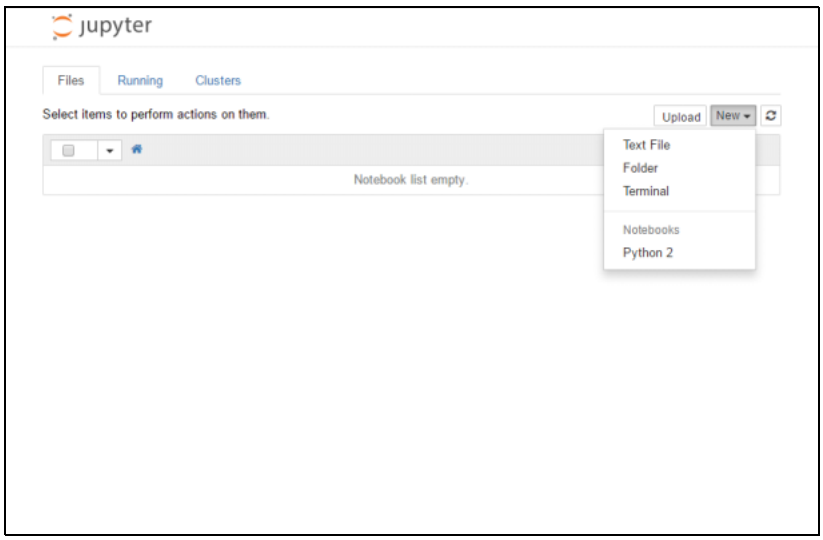
対話型種別「TensorFlow_Python2_cuda9.0_cudnn7.0」、「TensorFlow_Python2_cuda9.0_cudnn7.0」では、ONNX、onnx-tf を使用して、学習モデル形式の変換ができます。

ONNX の詳細については、ONNX の公式サイトを参照してください。
https://onnx.ai/

onnx-tf の詳細については、onnx-tf の公式サイトを参照してください。
https://github.com/onnx/onnx-tensorflow

- ONNX の版数の確認方法
  インストールされている ONNX の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。



  (2) 以下のコマンドを実行します。

```
# pip show onnx
```

  インストールされている onnx の情報が表示されます。

- onnx-tf の版数の確認方法
  インストールされている onnx-tf の版数は、以下の手順で確認できます。

  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。

  (2) 以下のコマンドを実行します。

```
# pip show onnx-tf
```

  インストールされている onnx-tf の情報が表示されます。

3.2.6.2 Chainer

対話型種別「Chainer_Python2_cuda9.1_cudnn7.1」、「Chainer_Python2_cuda9.1_cudnn7.1」では、ONNX、onnx-chainer を使用して、学習モデル形式の変換ができます。

ONNX の詳細については、「3.2.6.1 TensorFlow」(P.53) を参照してください。
onnx-chainer の詳細については、onnx-chainer の公式サイトを参照してください。
https://github.com/chainer/onnx-chainer

- ONNX の版数の確認方法
  「3.2.6.1 TensorFlow」(P.53) を参照してください。
- onnx-chainer の版数の確認方法
  インストールされている onnx-chainer の版数は、以下の手順で確認できます。
  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  (2) 以下のコマンドを実行します。

```
# pip show onnx-chainer
```

  インストールされている onnx-chainer の情報が表示されます。

3.2.6.3 MXNet

対話型種別「MXNet_Python2_cuda9.1_cudnn7.1」、「MXNet_Python2_cuda9.1_cudnn7.1」では、ONNX、onnx-mxnet を使用して、学習モデル形式の変換ができます。

ONNX の詳細については、「3.2.6.1 TensorFlow」(P.53) を参照してください。
onnx-mxnet の詳細については、onnx-mxnet の公式サイトを参照してください。
https://github.com/onnx/onnx-mxnet

- ONNX の版数の確認方法
  「3.2.6.1 TensorFlow」(P.53) を参照してください。
- onnx-mxnet の版数の確認方法
  インストールされている onnx-mxnet の版数は、以下の手順で確認できます。
  (1) 対話型環境（Jupyter）でターミナルを起動します。
  ［New］-［Terminal］を選択します。
  (2) 以下のコマンドを実行します。

```
# pip show onnx-mxnet
```

  インストールされている onnx-mxnet の情報が表示されます。

3.3 注意事項

対話型学習を使用する場合の注意事項について説明します。

- 1 テナント内で起動できる対話型学習は、一度に 1 つだけです。同時に 2 つ以上は起動できません。
- Docker イメージは、テナント単位に割り当てられたストレージ（NAS）の対話型リポジトリに保存されます。
- workspace に書き込んだデータは、ストレージ（NAS）の学習用ストレージに保存されます。
- trashbin は、最初に対話型環境を削除したときに作成されます。trashbin が存在しない場合はマウントされません。
  削除した対話型環境の workspace が空の場合、trashbin/workdir 配下へ workspace は移動されず、削除されます。
- プロセス数は最大 1,000 です。
- 対話型学習のユーザーは root ユーザーです。workspace のディレクトリの owner も root ユーザーになります。
- useradd は使用できますが、chown は使用できません。
- コマンドの仕様に一部制限があります。
- Jupyter 自体の設定は変更できません。Jupyter 自身のパッケージを置き換えた場合は、Docker イメージを起動できなくなることがあります。
- 時系列データ解析の学習または推論の実行完了までに、数時間以上かかる場合があります。
  UCI のサンプルデータセットを使用した場合の所要時間は以下です。
  学習／テスト（train_uci.py）：約 30 分
  推論（predict_uci.py）：約 5 分
- 時系列データ解析環境の /opt 配下のファイルは、時系列データ解析の環境内でだけ利用できます。
  時系列データ解析環境の /opt 配下のファイルの参照やダウンロードは許諾しておりません。ただし、/opt/ctsd/share/parameter.json、/opt/ctsd/share/setting_check_interval.json, /opt/ctsd/readme.txt については、参照を許諾しております。

第 4 章　対話型へのネットワークインターフェース追加

トピック：

この章では、対話型へのネットワークインターフェース追加機能の概要、機能、注意事項などについて説明します。

4.1 対話型へのネットワークインターフェース追加とは

対話型環境の Docker コンテナに、外部から直接接続できるネットワークインターフェース（外部接続用）を追加します。

4.2 機能概要

対話型環境にネットワークインターフェースを追加することにより、以下の機能やサービスが使用できます。

- 外部端末との接続
  Tera Term などのターミナルソフトウェアを使用して、外部の端末から対話型環境に直接接続できます。
- ファイル転送
  SCP または SFTP を使用して、大量のファイルを転送できます。
- サーバとしての利用
  対話型環境に各種サービス（サーバ機能）をインストールし、起動することで、外部の端末からサーバとして利用できるようになります。

4.3 機能詳細

4.3.1 接続構成

対話型環境と外部ネットワークの接続構成を図 12 に示します。

外部ネットワークと接続するには、以下の固定 IP アドレスが必要です。

- ブリッジの IP アドレス
  1 つ必要です。
- 各対話型環境のイーサネット 2（外部接続用）の IP アドレス
  対話型環境ごとに 1 つ必要です。
  Zinrai ディープラーニング システムでは、最大 16 の対話型環境を起動できます。お客様の利用環境に応じて、IP アドレスが 1 ～ 16 個必要になります。

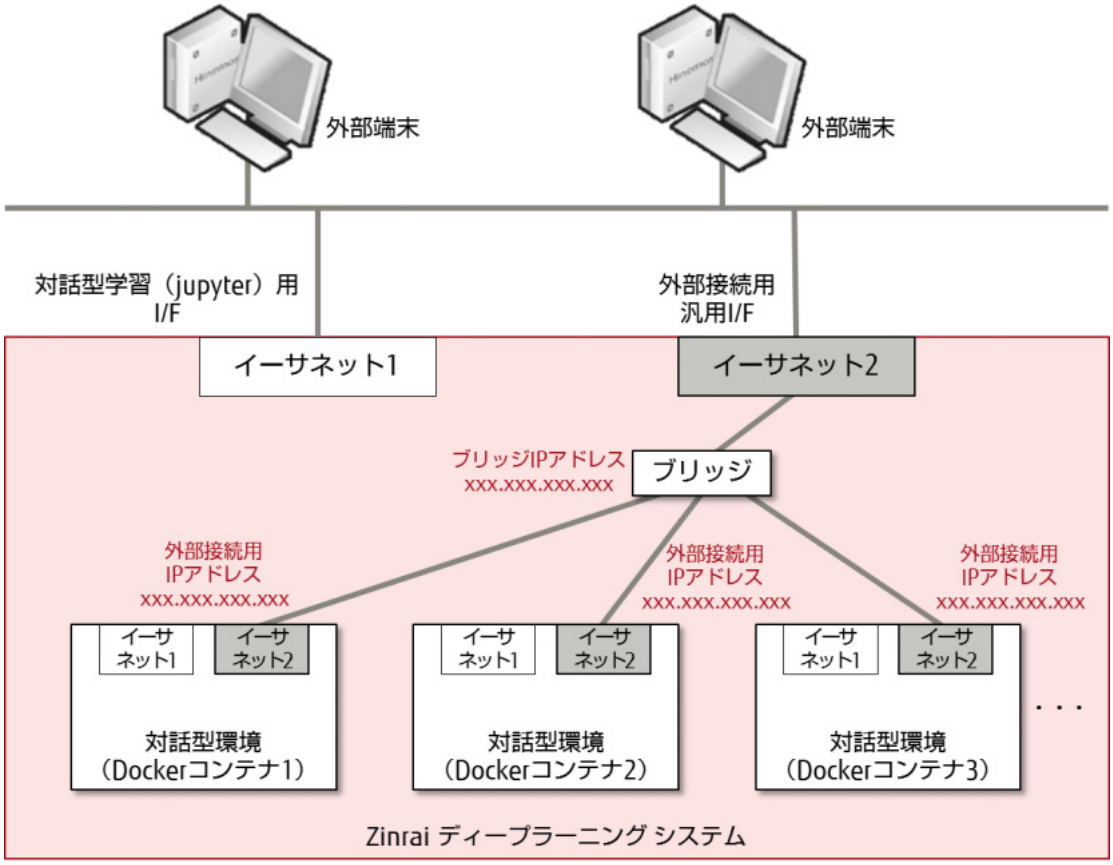
注　意

- 対話型環境を外部ネットワークと接続するには、対話型環境に外部接続用の IP アドレスを設定する必要があります。外部接続用 IP アドレスは、システム管理者に確認するか、またはターミナルで以下のコマンドを実行して確認してください。

```
ifconfig eth1
```

- Zinrai ディープラーニング システムのイーサネット 1（対話型学習用）、ブリッジ、および対話型環境のイーサネット 2（外部接続用）には、同一サブネットの IP アドレスを設定してください。これらには異なる IP アドレスを設定してください。重複する IP アドレスを設定した場合は、対話型環境の起動に失敗する可能性があります。

図 12：対話型環境と外部ネットワークとの接続構成



4.3.2　外部端末との接続

外部の端末から対話型環境に直接接続できます。
事前にファイアーウォールの設定が必要です。

設定の詳細については、『ユーザーズガイド』の「ファイアーウォールの設定」を参照してください。

4.3.3　ファイル転送

SCP または SFTP を使用したセキュリティの高いファイル転送が可能です。また、一度に大量のファイルを転送できます。

4.3.4　サーバとしての利用

対話型環境にインストールした各種サービス（サーバ機能）の起動を、supervisor 設定ファイル（/etc/supervisor/conf.d/*xxxx*.conf）に記述しておくことで、対話型環境起動時にサービスが自動的に起動されます。外部の端末から対話型環境をサーバとして利用できます。

設定の詳細については、『ユーザーズガイド』の「起動するサービスの設定」を参照してください。

4.4 注意事項

対話型へのネットワークインターフェース追加機能を利用する場合の注意事項について説明します。

- ファイアーウォール設定のエラー
  - ファイアーウォール設定ファイル（/workspace/filter/conf/filter.conf）の記述に誤りがあった場合、設定は適用されずに、/workspace/filter/result/filter.result にエラー行が出力されます。
  - 対話型環境を起動すると、ファイアーウォール設定ファイル（/workspace/filter/conf/filter.conf）を読み込んで、iptables にファイアーウォール設定が登録されます。iptables への設定でエラーが発生した場合、エラーが出力され、すでに設定されているフィルタリング設定はクリアされて、すべての通信が遮断されます。
- ファイル転送
  /home ディレクトリの容量は小さいため、大きいサイズのファイルを対話型環境に転送する場合は、事前に /workspace 配下にユーザーが書き込み可能なディレクトリを作成し、転送先に指定してください。必要に応じて、ユーザーを sudo ユーザーに登録してください。

第 5 章　エッジ連携

トピック：

この章では、エッジ端末上で推論するためのエッジ連携の概要、機能、注意事項などについて説明します。

5.1　エッジ連携とは

バッチ型学習で生成した学習済みモデルを利用して、エッジ端末上で画像データを推論できます。
エッジ連携が利用できるのは、お客様イントラネット環境内だけです。

エッジ連携では、エッジ端末上で推論するための以下の機能を提供します。

- ソフトウェアデベロップメントキットの提供
- エッジ端末上の推論で使用する学習済みモデルの提供
- エッジ推論に関連する各種データの管理

5.2　機能概要

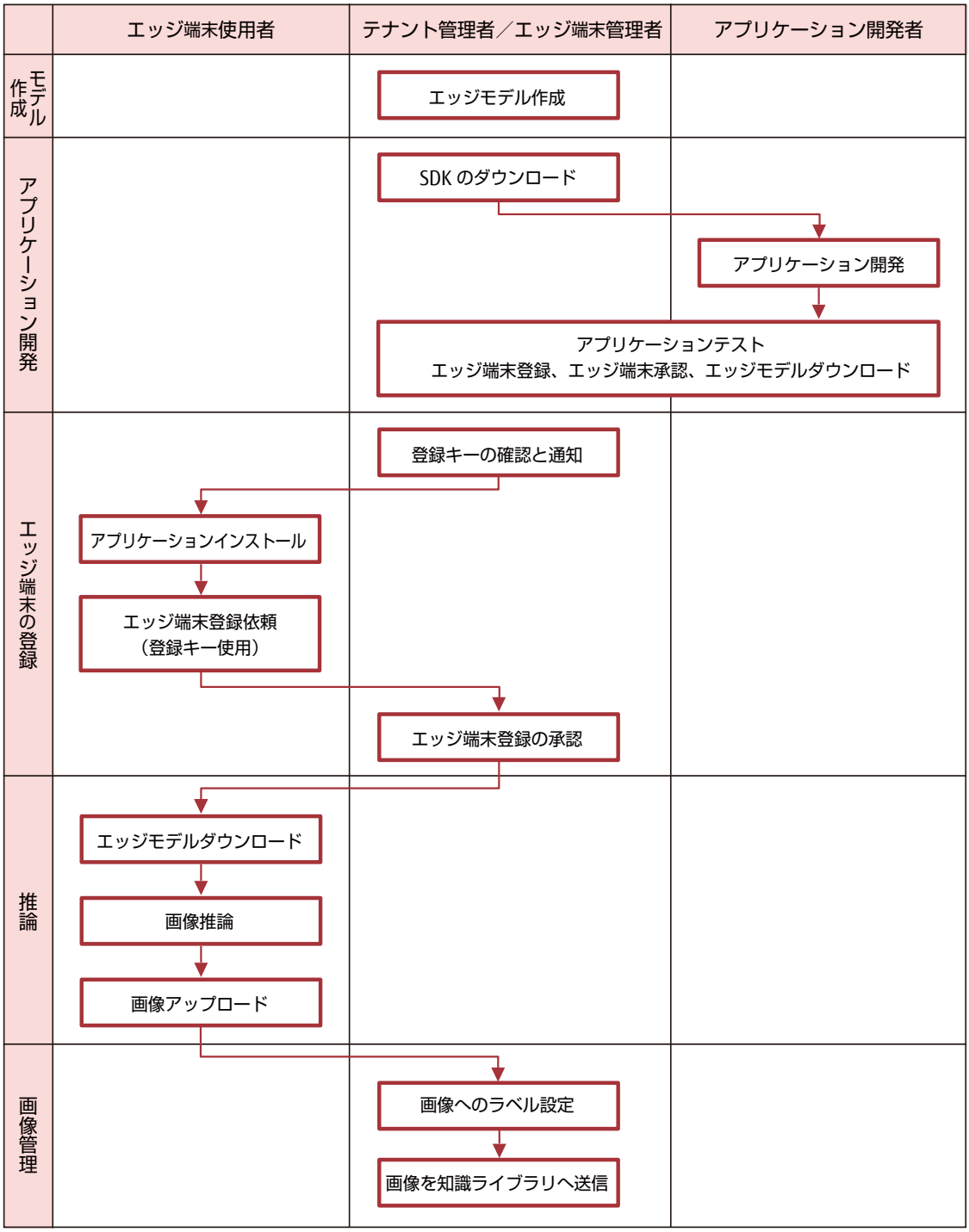
エッジ連携では、以下の機能を提供しています。

- ソフトウェアデベロップメントキット（SDK）
  エッジ端末から推論を行うアプリケーションを開発するための SDK を提供します。SDK はダウンロードして使用します。
- エッジ管理
  推論を行うエッジ端末を管理します。エッジ端末の登録承認、削除、一覧表示、エッジ端末情報の参照や更新などを行います。
- モデル管理
  バッチ型学習で生成した学習済みモデルを、エッジ推論用のエッジモデルに変換します。また、作成したエッジモデルを更新または削除できます。
- エッジモデルダウンロード
  推論で使用するエッジモデルを、エッジ端末にダウンロードできます。
- 画像アップロード
  エッジ端末上の推論で使用した画像データを、エッジ管理サーバにアップロードできます。
- 低認識率画像のプーリング
  エッジ端末上の推論結果が設定された認識率以下の場合、その画像と推論結果を自動保存します。また、必要に応じてエッジ管理サーバにアップロードできます。
- エッジ画像管理
  エッジ端末からアップロードされた画像データを管理します。アップロード画像の一覧表示、ラベルの付与、知識ライブラリへの登録、削除などを行います。

5.3　エッジ連携利用の流れ

エッジ連携を利用する場合の流れを以下に示します。

図 13：エッジ連携利用の流れ



備 考

SDK を利用して作成したアプリケーションは、お客様が管理し、利用者への配布を行います。

5.4　機能詳細

5.4.1　エッジ端末の動作環境

エッジ端末（Android および iOS）の動作環境を以下に示します。

- Android

表 5：エッジ端末（Android）の動作環境

項目	条件
対応 OS	Android 5.x, 6.x, 7.x
搭載 CPU	Qualcomm Snapdragon 800 MSM8974 相当以上
搭載 RAM	3GB 以上（*1）

*1:　推論処理で使用するメモリサイズ、および Java のヒープサイズの目安は、以下です。
　使用ヒープサイズ：エッジモデルのサイズ ＋ アプリケーションで使用するヒープサイズ
　使用メモリサイズ：上記使用ヒープサイズ ＋（エッジモデルのサイズ x 2）
例：エッジモデルサイズが 50MB、アプリケーションで使用するヒープサイズ が 30MB の場合
　　使用ヒープサイズ：80MB（= 50 + 30）
　　使用メモリサイズ：180MB（= 80 +（50 x 2））
以下のどちらかの場合、モデルの読込みでメモリ不足となり、当該エッジモデルは、端末で利用できません。
・使用ヒープサイズが、端末の Java 最大ヒープサイズを超える場合
・使用メモリサイズが、空きメモリサイズを超える場合

- iOS

表 6：エッジ端末（iOS）の動作環境

項目	条件
対応 OS	iOS 9.x , 10.x
搭載 CPU	A9 以上
搭載 RAM	2GB 以上（*1）

*1:　推論処理で使用するメモリサイズの目安は、以下です。
　使用メモリサイズ：（エッジモデルのサイズ x 2）+ アプリケーションで使用するメモリサイズ
例：エッジモデルサイズが 50MB、アプリケーションで使用するメモリサイズが 30MB の場合
　　使用メモリサイズ：130MB（=（50 x 2）+30）
使用メモリサイズが、空きメモリサイズを超える場合、モデルの読込みでメモリ不足となり、当該エッジモデルは、端末で利用できません。

5.4.2　ソフトウェアデベロップメントキット（SDK）

エッジ連携用の SDK を提供します。SDK は Web UI からダウンロードできます。

SDK では、以下が用意されています。

- アプリケーション開発のためのライブラリ
- ライブラリの API 仕様書
- ライブラリを使用したサンプルコード

Android 開発の場合は、Windows/Linux/Mac 上で AndroidStudio を使用して行います。iOS 開発の場合は、Mac 上の XCode を使用して行います。
詳細は、SDK のマニュアルを参照してください。

なお、SDK は推論エンジンとしてオープンソースの MXNet（v0.8）を使用しています。MXNet のライセンスは、Apache License, Version 2.0 です。アプリケーション開発は、Apache License, Version 2.0 のライセンス規定に準拠してください。Apache License, Version 2.0 のライセンス規定については、SDK に同梱されている LICENSE.txt を参照してください。

5.4.3　エッジ連携で使用可能な Layer 種別、Layer パラメーター

エッジ連携では、バッチ型学習で生成した学習済みモデルをエッジの推論エンジン（MXNet）用に変換した、エッジモデルを使用して推論します。

以下に、バッチ型学習で生成した学習済みモデルから MXNet に変換可能な Layer 種別、および MXNet で使用可能な Layer パラメーターを示します。

- MXNet でサポートしない Layer 種別が定義されている場合、エッジモデル新規作成の学習済みモデル選択時に、エッジ連携では使用できないことを通知する警告が表示されます。
- MXNet で使用できない Layer パラメーターが定義されている場合、MXNet では使用されません。

表 7：エッジ連携で使用可能な Layer 種別、Layer パラメーター

MXNet でサポートする Layer 種別	MXNet で使用可能な Layer パラメーター
BatchNorm	use_global_stats
	Eps
Concat	concat_dim

MXNet でサポートする Layer 種別	MXNet で使用可能な Layer パラメーター
Convolution	kernel_size
	kernel_h
	kernel_w
	Stride
	stride_h
	stride_w
	Dilation
	Pad
	pad_h
	pad_w
	num_output
	Group
	bias_term
Crop	-
Deconvolution	-
Dropout	dropout_ratio
Flatten	-
InnerProduct	num_output
	bias_term
Input	-
LRN	Alpha
	Beta
	K
	local_size
Pooling	kernel_size
	kernel_h
	kernel_w
	Stride
	stride_h
	stride_w
	Pad
	pad_h
	pad_w
	Pool
	global_pooling
ReLU	negative_slope
Sigmoid	-

MXNet でサポートする Layer 種別	MXNet で使用可能な Layer パラメーター
Softmax	-
Split	-
TanH	-

5.4.4　エッジ端末で使用できる機能

エッジ端末では、以下の機能を使用できます。

- デバイス登録
- エッジモデルダウンロード
- 推論
- 画像アップロード
- 低認識率画像のプーリング

デバイス登録

エッジ連携使用者は、テナント管理者／エッジ連携管理者から通知された登録キーを使用して、エッジ端末からエッジ端末の登録を依頼できます。

テナント管理者／エッジ連携管理者がエッジ端末の登録を承認すると、エッジ端末上で推論するための各機能が使用できるようになります。

注　意

- エッジ端末が Android の場合は、広告 ID（Advertising ID）を取得できる設定にしておいてください。エッジ端末が iOS の場合は、IDFV（identifierForVendor）を取得できる設定にしておいてください。
- デバイス登録は、お客様イントラネット接続が可能な環境で行ってください。

エッジモデルダウンロード

推論で使用するエッジモデルを、Zinrai ディープラーニング システムからエッジ端末にダウンロードできます。

注　意

- エッジモデルダウンロードは、お客様イントラネット接続が可能な環境で行ってください。
- ダウンロード時間は、エッジモデルのサイズやネットワーク環境に依存します。

推論

Zinrai ディープラーニング システムからエッジ端末にダウンロードしておいたエッジモデルを使用して、エッジ端末のカメラで撮影した画像およびエッジ端末に保存した画像の推論を行います。

画像アップロード

推論に使用した画像データと推論結果を、エッジ管理サーバにアップロードできます。
アップロードした画像は、Web UI から知識ライブラリに登録することで、追加学習の教師データとして使用できるようになります。

注　意

画像アップロードは、お客様イントラネット接続が可能な環境で行ってください。

低認識率画像のプーリング

推論の結果、画像の認識率が閾値以下の場合に、その画像と推論結果を自動保存します。
自動保存した画像と推論結果は、画像アップロード機能を使用して、エッジ管理サーバにアップロードできます。アップロードした画像は、Web UI から知識ライブラリに登録することで、追加学習の教師データとして使用できるようになります。

5.4.5　Zinrai ディープラーニング システムで使用できる機能

Zinrai ディープラーニング システム側では、以下の機能を使用できます。

- エッジ管理
- モデル管理
- エッジ画像管理
- ログ参照
- SDK ダウンロード

Web UI を使用した操作の詳細については、『ユーザーズガイド』の「エッジ連携を利用する」を参照してください。

エッジ管理

Web UI からエッジ端末の一覧表示、登録キー表示、承認、編集、および削除ができます。

● **一覧表示**

登録および登録依頼されているエッジ端末の一覧を表示します。

● **登録キー表示**

エッジ連携使用者に通知する登録キーを表示します。
エッジ連携使用者は、登録キーを使用してエッジ端末の登録依頼を行います。

● **承認**

エッジ一覧で登録の承認が完了していないエッジ端末を選択して、登録を承認します。

備 考

登録の承認が完了していないエッジ端末は、承認状態が "⚠" になっています。

承認が完了すると、承認状態が "✅" になり、エッジ端末からエッジモデルダウンロードおよび画像アップロードができるようになります。

● 編集

エッジ一覧で対象のエッジ端末を選択して、エッジ名、種別、または説明を更新します。エッジ名、種別、および説明以外の設定は更新できません。

● 削除

エッジ一覧で対象のエッジ端末を選択して、削除します。エッジ端末は複数選択できます。
エッジ端末を削除すると、エッジ端末からエッジモデルダウンロードおよび画像アップロードができなくなります。

モデル管理

エッジ推論では、バッチ型学習で生成した学習済みモデルを基に作成したエッジモデルをエッジ端末にダウンロードして、オフラインで推論を行います。

◯ 備 考

データの改ざん、不正コピー、漏洩を防止するために、データは以下の方法で保護されています。
- エッジ端末と Zinrai ディープラーニング システム間の通信経路を暗号化（SSL）

Web UI からエッジモデルの一覧表示、新規作成、編集、および削除ができます。

● 一覧表示

登録されているエッジモデルの一覧を表示します。また、エッジモデルの作成状況を表示します。
エッジモデル名をクリックすると、詳細情報が表示されます。

● 新規作成

バッチ型学習で生成した学習済みモデルを基に、エッジモデルを新規に作成します。
エッジモデル名、バージョン、モデル種別、学習済みモデル、有効期限、説明を設定します。作成を実行すると、Android/iOS 用のエッジモデルが作成されます。

● 編集

エッジモデル一覧で対象のエッジモデルを選択して、エッジモデル名、バージョン、または説明を更新します。エッジモデル名、バージョン、および説明以外の設定は更新できません。

注 意

- エッジ端末に保存されているエッジモデルは更新されません。必要に応じて、更新されたエッジモデルを Zinrai ディープラーニング システム側からダウンロードし直してください。
- 有効期限を変更する場合は、新しいエッジモデルを作成してください。

● 削除

エッジモデル一覧で対象のエッジモデルを選択して、削除します。エッジモデルは複数選択できます。

エッジ画像管理

エッジ端末からエッジ管理サーバへアップロードされた画像データを管理します。

推論で使用したエッジモデルごとに、Web UI から、アップロードされた画像の一覧表示、ラベルの付与、知識ライブラリへの登録、および画像の削除ができます。

● 一覧表示

エッジ端末からアップロードされた画像の一覧を表示します。また、知識ライブラリへの登録状況を表示します。

◯ 備 考

エッジ管理サーバ上で削除されたエッジモデルで推論したアップロード画像は、すべてのエッジモデルのアップロード画像一覧に表示されます。ただし、ラベルおよび推論スコアは表示されません。

● ラベルの付与

アップロード画像一覧で対象の画像を選択して、画像データにラベルを付与します。
ラベルを付与した画像は、知識ライブラリに登録できます。

● 知識ライブラリ登録

エッジ端末上の推論で使用した画像データを、追加学習の教師データとして使用したい場合は、画像データを知識ライブラリに登録します。

アップロード画像一覧で対象の画像を選択して、知識ライブラリに登録します。

◯ 備 考

- 知識ライブラリへ登録する画像は、ラベルが付与されており、状態が「登録中」以外である必要があります。
- 登録する画像は、2 枚以上選択してください。
- 登録する画像のラベルが「None」の場合、または状態が「登録中」の場合、登録エラーになります。

以下のデータが知識ライブラリに登録されます。

- 画像データ
- ラベル情報ファイル

● 削除

アップロード画像一覧で対象の画像を選択して、削除します。画像は複数選択できます。

◯ 備 考

すべてのエッジモデルのアップロード画像一覧で表示されている画像（エッジ管理サーバ上で削除されたエッジモデルで推論したアップロード画像）は、あるエッジモデルの一覧でその画像を削除すると、それ以外のエッジモデルの一覧からも削除されます。

SDK ダウンロード

Web UI から、SDK をダウンロードできます。

ログ参照

指定した対象日の、エッジ端末の操作履歴および Web UI の操作履歴のログ一覧を表示します。
ログの登録日時ごとに、デバイス名、操作、状態、結果、および詳細を確認できます。

5.5 注意事項

エッジ連携を利用する場合の注意事項について説明します。

- Zinrai ディープラーニング システムのサービス停止中のアクセス
  Zinrai ディープラーニング システムのサービス停止中は、エッジ端末から Zinrai ディープラーニング システムにアクセスできません。SDK のアプリケーションは、接続エラーになります。

第 6 章　知識ライブラリ

トピック：

この章では、知識ライブラリの概要、機能、注意事項などについて説明します。

6.1　知識ライブラリとは

知識ライブラリとは、お客様が利用可能な学習モデルや教師データなどが格納されているライブラリで、ストレージ（NAS）の一部に含まれます。

ストレージ（NAS）は、以下のディレクトリとデータで構成されています。

図 14：ストレージ（NAS）の構成

- 学習用ストレージ（/learning）には、学習で使用するデータを格納します。
- 知識ライブラリ（/knowledge）には、学習時に入力する教師データ、Edge 用データなどを格納します。
- 対話型リポジトリ（/jupyter）には、対話型学習で使用するデータ、Docker イメージを格納します。

6.2　機能概要

知識ライブラリでは、Zinrai ディープラーニング システムで使用する各種データを集約して、管理します。
以下の機能を提供します。

- データ転送
  知識ライブラリとエッジ／学習コンポーネント間で、データを転送します。
- データ管理
  知識ライブラリ内のデータの一覧表示およびその詳細情報表示、属性情報の更新／削除を行います。
- データマージ
  知識ライブラリに格納されている、Edge 用データをマージします。

6.3　機能詳細

6.3.1　データ転送

知識ライブラリとエッジ／学習コンポーネント間でデータを転送します。

Web UI を使用した操作の詳細については、『ユーザーズガイド』の「画像を知識ライブラリに登録する」、「知識ライブラリへのデータアップロード」、または「モデルの詳細表示」を参照してください。

● 知識ライブラリへのデータアップロード

- クライアント PC 上の学習データを、Web UI から知識ライブラリにアップロードします。
  アップロードする学習データは、事前に zip 形式に圧縮しておく必要があります。また、Web UI でアップロード状況を確認できます。
- エッジ推論で使用した画像データを、Web UI から知識ライブラリへアップロードします。アップロードした画像データ（Edge 用データ）は、バッチ型学習の追加学習の教師データとして使用できるようになります。
- バッチ型学習のスナップショット情報を、Web UI から知識ライブラリへ転送します。

● 知識ライブラリへの対話型データ登録

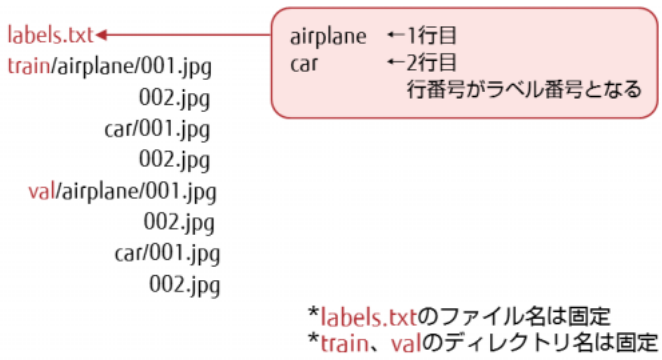
- 対話型学習の作業域（workspace）のデータ（対話型データ）を知識ライブラリに登録します。
  登録する対話型データは、事前に workspace/uploaddata 配下に作成した任意のディレクトリに配置しておく必要があります。また、Web UI で登録状況を確認できます。

6.3.1.1　登録するデータの構成

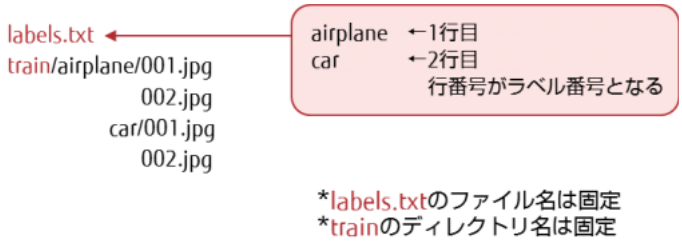
データ転送機能を使用して登録するデータは、以下のディレクトリ構成とデータ形式に従ってください。

教師データ

- ディレクトリ構成
  - train、val が別々にある場合



- train だけの場合（比率指定により、train から val を分離生成）



- データ形式
  教師データ（ラベル、画像データ）は、以下の形式に従っている必要があります。

表 8：教師データの形式（知識ライブラリ）

項目	ラベル（labels.txt）	画像データ
ディレクトリ構成	• labels.txt で定義したラベル（ディレクトリ名）が、train ディレクトリおよび val ディレクトリ配下に存在している • labels.txt で定義するラベル（ディレクトリ名）に、以下の文字を指定しない • /、¥、\、:、*、?、"、<、>、|、#、{、}、%、&、~ • \0（バックスラッシュ 0） • マルチバイト文字コード（日本語を含む）	教師データを格納するディレクトリは、「train」または「val」の固定名にする
ファイルの拡張子	ファイル名は labels.txt の制限に準ずる	• .JPG または .jpg • .JPEG または .jpeg • .PNG または .png
1 カテゴリー当たりのファイル数	1	2 ～ 2,000,000,000
ファイル名	半角英数字（A ～ Z、a ～ z、0 ～ 9）、_（アンダースコア）、-（ハイフン）、.（ピリオド）	
ファイル名の文字数	1 ～ 64 文字 + 拡張子の文字数	
ファイルサイズ	制限なし	
対応画像モード	グレースケール、RGB、RGBA、パレットモード	

モデルセット、Edge 用データ、その他

モデルセット、Edge 用データ、およびその他は、教師データとは異なり、特定の形式はありません。
その他とは、対話型学習で使用するデータを指します。
知識ライブラリへアップロードしたあとは、対話型環境の Jupyter を使用して、自由に加工することができます。

- ディレクトリ構成
  特別な形式はありません。

- データ形式
  以下の記述仕様に従っている必要があります。この範囲内で自由に指定してください。

表 9：モデルセット、Edge 用データ、その他の形式（知識ライブラリ）

項目	記述仕様
ディレクトリ名	以下の文字を使用可能とする • 半角英数字（A ～ Z、a ～ z、0 ～ 9）、_（アンダースコア）、-（ハイフン）、.（ピリオド）、" "（スペース）、！（エクスクラメーション）、$（ダラー）、’（アポストロフィー）、(（左括弧）、)（右括弧）、+（プラス）、,（カンマ）、@（アットマーク）、[（左角括弧）、]（右角括弧）、^（サーカムフレックス）、`（グレーブアクセント） 以下の文字は使用しないこと • /、¥、\、:、*、?、"、<、>、|、#、{、}、%、&、~ • \0（バックスラッシュ 0） • マルチバイト文字コード（日本語を含む）
ディレクトリ名の文字数	1 ～ 255 文字
ファイル名	以下の文字を使用可能とする • 半角英数字（A ～ Z、a ～ z、0 ～ 9）、_（アンダースコア）、-（ハイフン）、.（ピリオド） 以下の文字は使用しないこと • /、¥、\、:、*、?、"、<、>、|、#、{、}、%、&、~ • “ ”（スペース）、!、$、’、(、)、+、,、@、[、]、^、` • \0（バックスラッシュ 0） • マルチバイト文字コード（日本語を含む）
ファイル名の文字数	1 ～ 64 文字＋拡張子の文字数
ファイルサイズ	制限なし

6.3.1.2 知識ライブラリへのデータ登録

知識ライブラリへのデータ登録は、以下の方法で行います。

- Web UI のデータアップロード機能を使用する
- Web UI の対話型データ登録機能を使用する

データアップロード機能を使用する

Web UI から知識ライブラリへデータをアップロードします。

知識ライブラリ内のディレクトリ構成やデータ形式には決まりがあります。データをアップロードする前に、「6.3.1.1 登録するデータの構成」(P.73) に従ってデータを準備してください。
アップロードするデータは、「6.3.1.1 登録するデータの構成」(P.73) に示すディレクトリ形式で解凍される zip 形式のファイルにしてください。zip 形式のファイルは、ZIP64 に対応したツールで作成してください。

注　意

- 教師データが train ディレクトリと val ディレクトリにある場合は、以下のファイルおよびディレクトリを選択して作成してください。
  - labels.txt ファイル
  - train ディレクトリ
  - val ディレクトリ
- 教師データが train ディレクトリだけにある場合は、以下のファイルおよびディレクトリを選択して作成してください。
  - labels.txt ファイル
  - train ディレクトリ

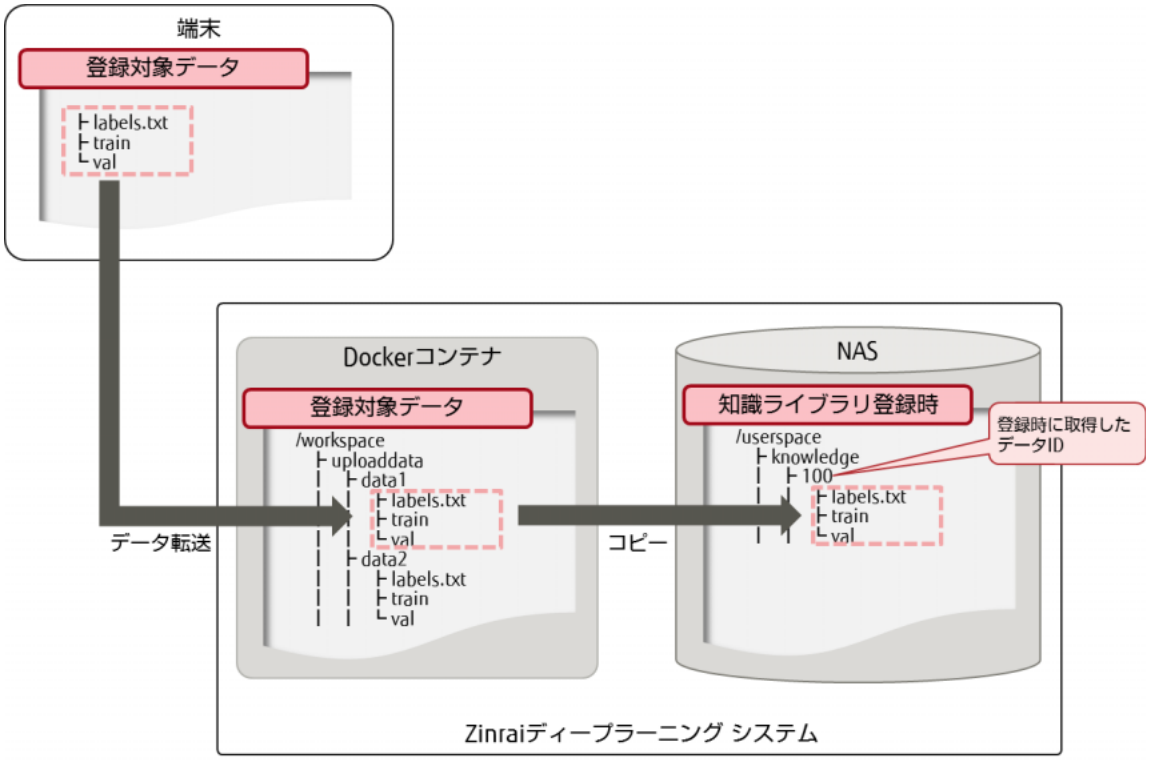
対話型データ登録機能を使用する

Web UI から知識ライブラリへ対話型データを登録します。

知識ライブラリ内のディレクトリ構成やデータ形式には決まりがあります。対話型データを登録する前に、「6.3.1.1 登録するデータの構成」(P.73) に従ってデータを準備してください。
登録する対話型データは、対話型環境の workspace/uploaddata 配下に作成した任意のディレクトリに配置してください。
ディレクトリ名は以下の条件に従ってください。
　文字数：Linux の制限による
　使用可能文字種：A ～ Z、a ～ z、0 ～ 9、_

登録を実行すると、ディレクトリ構成やデータ形式が正しいか構成チェックが実施されます。構成チェックのエラーについては、『ユーザーズガイド』の「対話型データの登録状況の確認」を参照してください。

図 15：知識ライブラリへの対話型データ登録



知識ライブラリに登録可能なデータ種別

以下に、知識ライブラリに登録できるデータを示します。

- 教師データ
- モデルセット
- Edge 用データ
- その他

6.3.2　データ管理

知識ライブラリ内のデータを管理します。
Web UI から知識ライブラリ内のデータの一覧表示およびその詳細情報表示、属性情報の更新／削除ができます。

Web UI を使用した操作の詳細については、『ユーザーズガイド』の「知識ライブラリのデータ管理」を参照してください。

● 所持データー覧の表示

知識ライブラリ内のデータの一覧を表示します。データ名をクリックすると、詳細情報が表示されます。

● 編集

所持データー覧で対象のデータ名を選択してデータを更新、またはデータ詳細画面でデータを更新できます。

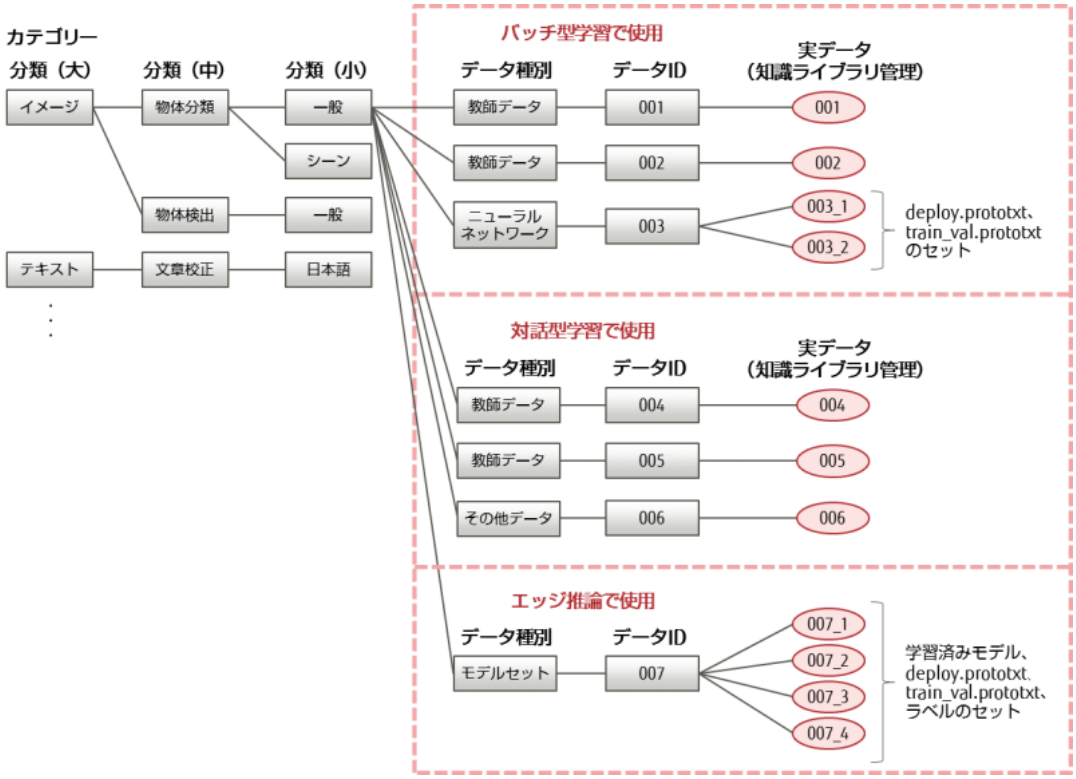
● 削除

所持データー覧で対象のデータ名を選択してデータを削除、またはデータ詳細画面でデータを削除できます。
所持データー覧から削除する場合は、複数選択して削除できます。

6.3.2.1　知識ライブラリで管理するデータの種類

知識ライブラリ内のデータは、以下に示すカテゴリー（属性分類）に従って登録し、関連付けておく必要があります。

図 16：知識ライブラリの管理データ



- カテゴリー
  分類（大）、分類（中）、分類（小）の 3 つのカテゴリーがあります。

  表 10：データのカテゴリー（属性分類）

分類（大）	分類（中）	分類（小）
イメージ	物体分類	一般
		シーン
		日本語手書き文字
	物体検出	一般
		交通
		日本語活字
	説明文生成	英語
	領域分割	一般
	物体追跡	一般
テキスト	文章校正	日本語
音声	音声認識	日本語

- データ種別
  - 教師データ
  - モデルセット
  - Edge 用データ
  - その他

6.3.3　データマージ

知識ライブラリ内のラベル（labels.txt）が同じ Edge 用データをマージします。

- 2 つ以上のマージ対象データを、1 つのデータにマージします。
- マージしたデータは、新規データとして格納されます。マージ対象データは、既存のデータとして残ります。
- マージしたデータは、さらにほかのマージ対象データとマージできます。

Web UI を使用した操作の詳細については、『ユーザーズガイド』の「知識ライブラリのデータのマージ」を参照してください。

6.3.3.1　マージ可能なデータ

マージ可能なデータは、以下の条件を満たしているデータです。

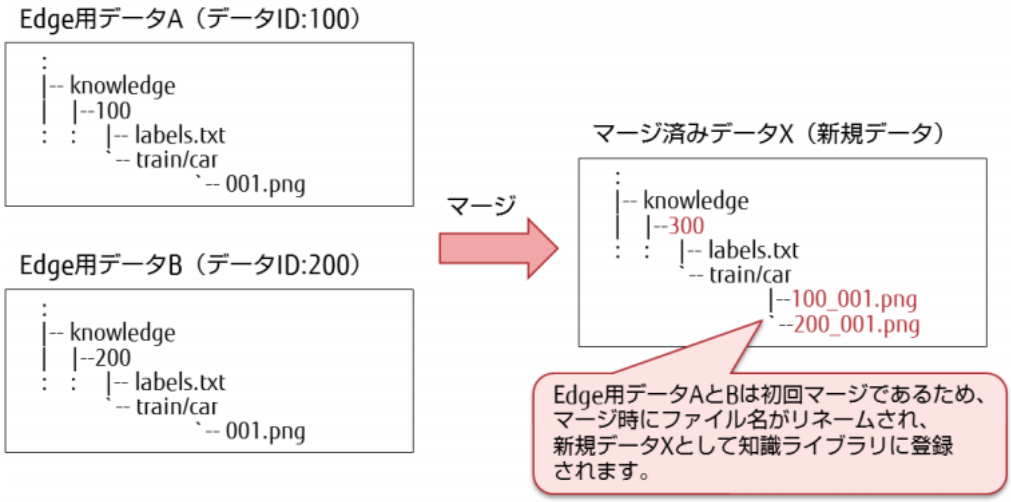
- 所持データ一覧に表示されている
- データ種別が Edge 用データ
- ラベル（labels.txt）が同じ

6.3.3.2　ファイルのリネーム

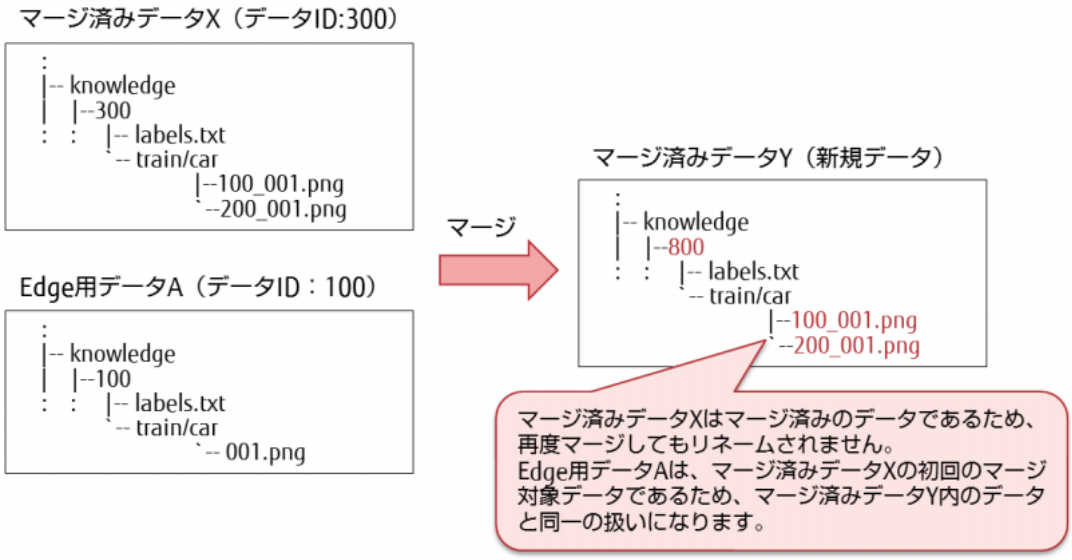
データをマージすると、ファイル名は「マージ対象データ ID_ 元のファイル名」にリネームされます。知識ライブラリでは、データがマージ済みデータか、マージ未データかを管理しています。リネームはマージ未データをマージする場合に行います。マージ済みデータをマージする場合はリネームされません。

以下に、データのマージ例を示します。

- 初回マージの場合
  Edge 用データ A と Edge 用データ B のマージ



- マージ済みデータと、一度マージ対象データに選択したデータをマージした場合
  マージ済みデータ X と Edge 用データ A のマージ



注　意

- データをマージすると、ファイル名の先頭に「マージ対象データ ID_」（最大 11 文字）が付加されます。
- 一度マージされたデータは、すでにリネームされているため、再度リネームされることはありません。
- データ ID はディレクトリごとに一意であるため、一度マージされたデータを再度マージするときにデータの重複が発生しますが、マージするデータ同士は同じものであり、マージされません。
- マージ対象データのファイル名は 244 文字以下にしてください。リネームの結果、ファイル名が 255 文字を超えると、マージ失敗（ファイル名文字数超過）となります。その場合は、ファイル名の文字数を 244 文字以下に変更し、再度新規アップロードを行ってください。

6.4　注意事項

知識ライブラリを使用する場合の注意事項を説明します。

- データを削除すると、データは復元できません。削除するときは十分に注意してください。
- 一度にマージ可能なデータの数は、「所持データ一覧で選択した 1 データ」＋「マージ対象選択画面で選択可能な 99 データ」の合計 100 データです。101 以上のデータをマージする場合は、マージしたデータをさらにマージしてください。
- マージ操作中に別画面でマージ対象データの削除操作が行われた場合は、データマージの受付けに失敗します。
- 対話型データの登録完了後、不要になった workspace/uploaddata 配下のデータは削除してください。

用語集

A

Advanced learning rate option

より詳細な学習率を指示します。

AlexNet

画像認識用多層ニューラルネットの1つです。2012年の画像認識プログラムコンテスト(ILSVRC)で優勝し、現在の画像認識ニューラルネットのベースとなっています。

Android

Google 社によって開発された、Linux カーネルの OS です。スマートフォンおよびタブレット用 OS として多く採用されています。

B

Base Learning Rate

初期学習率です。

Batch Accumulation

バッチ平滑化のサイズを指示します。Batch Accumulation(=iter_size) * batch_size にパラメーターの更新が平滑化されます。

C

Caffe

オープンソースとして、Berkeley Vision and Learning Center が中心となって開発している Deep Learning 用フレームワークです。

.caffemodel

Caffeにおいて、指定された学習イテレーション回数または中断時に作成される学習パラメーターのスナップショットファイルです。推論、追加学習、学習再開に利用します。

Caffe ログファイル

Caffe が、学習開始、学習中、学習終了、異常発生時に出力するログファイルです。

CPU

Central Processing Unit 。画像データベースファイルの読込み処理や学習データ共有処理などで利用します。

Crop

画像変換の種類です。元画像の中央付近を中心に切り出します。

CropSize

画像データとして切り出す画像のサイズです。

D

Deep Learning

ニューロン網を模した Deep Neural Network (DNN) に、画像や音声などを学習させる AI 技術です。

Deep Neural Network

ニューロン網を模した多層ネットワークです。

deploy

学習したモデルを使って認識します。

deploy.prototxt

認識に使うニューラルネットワークの設定です。

Dev_edge

エッジ端末のアプリケーション開発者用ロールです。

Dev_inference

推論開発者用ロールです。

Dev_learning

学習開発者用ロールです。

DNN

Deep Neural Network の略称です。

E

epoch

学習の 1 サイクルを示す単位です。1 エポックの中で、ランダムにシャッフルした学習データを用いてミニバッチにより学習を行い、モデルのパラメーターを更新します。

F

FIFO 方式

複数の対象を取り扱う順序を表し、最初に入れたものを最初に取り出す（First-In First-Out）方式です。

Fill

画像変換の種類です。元画像の幅・高さの比を保ち指定サイズに画像全体を収めます。

G

GoogLeNet

画像認識用多層ニューラルネットの1つです。2014年の画像認識プログラムコンテスト(ILSVRC)で優勝しています。

GPU

Graphics Processing Unit 。コンピュータの画像処理を行うプロセッサです。CPU に比べ並列計算処理能力にすぐれているため、Deep Learning における学習処理にも適しており、活用されています。複数の GPU を利用して学習処理を並列実行します。

H

Half Crop & Half Fill

画像変換の種類です。Crop と Fill の効果を半分ずつ適用します。

I

iOS

Apple 社によって開発された OS です。Apple 社製スマートフォンおよびタブレット用 OS として採用されています。

J

Jupyter

Web ブラウザから操作できる対話型評価環境です。

L

LeNet

画像認識用多層ニューラルネットの 1 つです。AlexNet や GoogLeNet と比較すると小規模な構成となっています。MNIST を使った手書き文字認識でよく利用されます。

LMDB

Caffe が標準で対応している Key-Value 型のデータベースです。一般的に、ディープラーニングフレームワークでは学習時に効率的なデータの読取りを行うため、事前に画像データなどをデータベースに変換します。

M

max_iter

最大イテレーション数です。

MNIST

手書きの数字の画像ファイルデータベースです。

MPI

Message Passing Interface 。並列計算用ライブラリの標準規格です。

N

network.prototxt

Caffe において、主に DNN のレイヤー構成を定義する設定ファイルです。

NN

ニューラルネットワークの略称です。

NN 最適化

学習済みのモデルから、より教師データとの誤差が少ない学習モデルを見つけ出すことです。NN 最適化機能では、ハイパーパラメーターの組合せを多数生成し、実際に Caffe で学習を行い、その評価結果の一覧を行います。

P

PreviousNetwork

直前に学習したニューラルネットワークです。

prototxt

Caffe のニューラルネットワーク、学習パラメーターが記述された定義ファイル群です。

R

Random seed

ニューラルネットワークの weight の初期乱数のシードを指示します。指示がないと毎回異なる系列の乱数になります。

S

SDK

Software Development Kit 。アプリケーション開発に必要なライブラリや API 仕様などをひとまとめにしたものです。エッジ連携では、エッジ推論のアプリケーションを開発するための SDK を提供しています。

Snapshot interval

何エポックごとにスナップショットを出力するかを指示します。

solver.prototxt

Caffe において、network.prototxt で定義された構成の DNN に対し、主に学習中の動作を決定する属性設定ファイルです。

.solverstate

Caffe において、指定された学習イテレーション回数または中断時に作成される、DNN の学習状態のスナップショットファイルです。学習再開に利用します。

Solver type

最適化アルゴリズムを選択します。

Squash

画像変換の種類です。元画像全体を指定サイズに拡縮します。

T

Top predictions

どのカテゴリーだと認識されたか、認識率の高い順に並べたものです。

Train

データを入力して学習します。

train_val.prototxt

学習に使うニューラルネットワークの設定です。

V

Val

正しく学習できているか評価します。

Validation interval

何エポックごとに検証を行うか指示します。

Z

Zinrai ディープラーニング システム

Zinrai 商品群の 1 つです。ディープラーニングに必要となる高速な GPU やそれを支えるツールなどを提供します。

あ

イテレーション

学習パラメーターのバックワード処理による更新を行う学習処理単位です。

イメージタイプ

静止画像のフォーマット種別です。

エッジ端末

学習済みモデルの推論を実施する端末です。Android、iOS などのスマートフォン端末やタブレット端末を指します。

エッジ推論

Android、iOS などの各プラットフォームを使って、オフラインの状態で推論することです。

エッジモデル

エッジ端末用にコンバートされた、学習済みモデルのことです。

エッジ用 DL ライブラリ

エッジ端末上で撮影した画像などを、Zinrai ディープラーニング システムで作成した学習済みモデルを基に推論する機能です。

重み

複数のニューロンが結合しているとき、その結合の強さを示すデータ群です。学習処理によって値が変化し、最適化が試みられます。

か

学習

モデルとデータセットを使用して、ディープラーニングを行うことです。

学習 GPU サーバ

Zinrai ディープラーニング システムにおいて、Deep Learning 学習用に利用する GPU 搭載サーバです。

学習サービス

ユーザーの教師データを用い、代表的なディープラーニングフレームワークと GPU を使った学習を行うことができるサービスです。

学習処理

画像や音声などを認識できるようにするために、DNN が自身の学習パラメーターを計算／更新する処理です。

学習済みモデル

学習が終わったときに出力される以下の成果物です。

- deploy.prototxt（推論用のニューラルネットワーク）
- Caffemodel ファイル

学習済みモデルの保護機能

学習済みモデルや通信経路の暗号化による改ざん／漏洩防止機能、および学習済みモデルに対する有効期限設定機能です。

学習データ共有処理

複数 GPU を使用した学習処理において、各 GPU で行われている学習パラメーターの更新量の計算結果を、互いにデータ共有して平均化などを行い均一化する処理です。

学習パラメーター

DNN のニューロン間の重みデータセットです。

学習用画像 DB

学習用画像データベースファイルの略称です。

学習用画像データベースファイル

Caffe において、DNN への入力データとなる画像ファイルのアーカイブです。学習（学習パラメーターの更新処理を伴う処理）に利用します。

学習率

バックワード処理で誤差を基に学習パラメーターを変更する際に、その変更度の大きさを決める係数です。

画像 DB

画像データベースファイルの略称です。

画像管理

エッジ端末からアップロードされた画像に対するラベル設定、知識ライブラリ転送などを行う管理機能です。

画像データベースファイル

Caffe において、DNN への入力データとなる画像ファイルのアーカイブです。それぞれ学習用とテスト用があります。

画像ファイル

Caffe においては、複数ファイルをアーカイブしてデータベースファイルを作成します。これを DNN の入力に与え、学習処理に利用します。

カテゴリー名

認識分類する対象名です。

関連データセット

ワーキングセットやモデルで使用しているデータセットのことです。

関連モデル

ワーキングセットで使用しているモデルのことです。学習時に使用したデータセットに紐付くモデルのことです。

教師データ

DNN の学習処理の中で、入力層に対する入力データに対し、出力層の出力結果として期待されるデータです。

誤差

DNN の入力層に対する入力データに対し、出力層における実際の出力データと教師データの差分の大きさを特定関数で評価した値です。

さ

サブミットジョブ

ジョブをジョブキューに投入することです。

システム管理者

Zinrai ディープラーニング システム環境全体を管理します。

出力層

DNN において、DNN 全体の結果データ出力を担うレイヤーです。一般的に、入力層の対極にある最終層が担います。

ジョブ

学習、推論、教師データ作成など、お客様のデータを加工したり解析したりする作業全般です。

ジョブ ID

ジョブを制御するための識別名です。

ジョブキュー

ジョブ実行の待ち行列です。

推論

DNN において、学習済みデータを用いて、入力データに対する出力結果評価を行うことです。

推論サービス

ユーザーの学習モデルを利用した推論環境の構築と、その環境を運用する機能を提供するサービスです。

スタンダードネットワーク

システムが提供する標準的なニューラルネットワークです。

スナップショット

学習途中のニューラルネットワークの荷重、ソルバーの状態を保存したファイルです。Caffe においては、.caffemodel と .solverstate を合わせてスナップショットと呼びます。

正解率

学習中／学習後の DNN において、テスト入力データ群に対して、期待される結果出力（正解）が得られた割合です。出力された最上位候補が正解である割合を Top1 正解率、上位 5 個の候補の中に正解が含まれている割合を Top5 正解率と呼びます。

ソフトウェアデベロップメントキット

アプリケーション開発に必要なライブラリや API 仕様などをひとまとめにしたものです。エッジ連携では、エッジ推論のアプリケーションを開発するための SDK を提供しています。

ソルバー

バックワード処理で、誤差を基に学習パラメーターを変更する際のアルゴリズムです。Caffe でのソルバーの既定値は SGD（確率的勾配降下法）です。
http://caffe.berkeleyvision.org/tutorial/solver.html

た

探索ステージ当たりの子世代数

1 つのステージ中で、個体をいくつ生成するのかを定義するパラメーターです。個体の多様性を表したもので、探索対象のモデルが複雑な場合は、子世代数が多い方が、最適値を見つけやすくなります。

探索ステージ数

遺伝的アルゴリズムでは世代数に対応するパラメーターです。1 つのステージで、個体（ハイパーパラメーターの組合せ）の生成と、教師データとの誤差の少なかった個体の選抜が行われます。選抜した結果は、次のステージの個体生成に使われます。そのステージを何回繰り返すのかを定義するパラメーターです。

端末管理

Zinrai ディープラーニング システムで作成した学習済みモデルなどを利用するエッジ端末を、管理するための機能です。

端末承認

登録依頼のあった端末を、エッジ連携として使用して問題ないかを判断し、承認する操作です。

端末登録（依頼）

エッジ連携で使用する端末から、登録依頼する操作です。

端末認証

Zinrai ディープラーニング システム側とエッジ端末側でデータ通信するときに、端末登録されたものとだけ通信可能とする認証機能です。

知識ライブラリ

Zinrai ディープラーニング システムの、ユーザーの学習モデルを管理するライブラリです。

中間層

DNN において、入力層と出力層の間に配置されるレイヤー群です。

データセット

以下を含むものです。

- caffe 入力用に教師データを LMDB に変換したデータ
- 平均画像データ（Deep Learning で学習の入力となるデータ）

テナント

Zinrai ディープラーニング システムサービスの管理単位で提供されるサービスプラットフォームです。

テナント管理者

テナント内全体を管理します。

登録キー

エッジ連携で使用する端末から登録依頼する際に指定する、テナントごとに割り振られたキーです。

な

ネットワーク名

モデルの構造に付けられた名前です。

ニューラルネットワーク

人間の脳の神経回路の仕組みを模したモデルです。

は

ハイパーパラメーター

学習の結果に影響を与える、DNN や学習特性に関する各種パラメーターです。

バックエンド DB

機械学習で使用するデータベースの種類です。LMDB と LevelDB があります。

バックワード処理

DNN での学習処理において、フォワード処理で得られた出力結果と期待した出力結果との差を確認し、期待した結果が得られやすいように学習パラメーターを変更する一連の処理です。多層になるネットワークで、出力側の層から順に処理することからこう呼びます。

バッチサイズ

1 イテレーションごとに抽出する学習データ件数です。

フォワード処理

DNN での学習処理において、画像などの入力データを学習パラメーターに基づき処理し、出力結果を得る一連の処理です。多層になるネットワークで、入力側の層から順に処理することからこう呼びます。バックワード処理とは異なり、基本的には、フォワード処理中に学習パラメーターの更新は行いません。

平均画像

学習画像の正規化に使用する全画像の平均を取った画像です。

ま

モデル並列

複数の計算ノードで、多段になっているネットワーク層を分担して担当し、並列処理する手法です。層の深いネットワークを学習させる際に効果的ですが、処理ノードによって計算量にばらつきがあるため同期の際の待合せ時間が長くなる傾向があります。

や

有効期限設定

エッジ端末上で学習済みモデルを利用できる期限を設定する機能です。

ユーザーデータ

お客様の持ち込んだデータや学習支援ツールで作成したデータです。

- 教師データ
- prototxt（ニューラルネットワーク、学習パラメーターが記述された定義ファイル）
- 教師データ作成ツールの出力結果

ら

レイヤー

DNN を構成するネットワーク層です。Caffe を含む多くのディープラーニングフレームワークでは、役割や特徴が異なる多種のレイヤーを提供しています。

レイヤー名

ニューラルネットワークを構成する各レイヤーに付けられた名前です。

FUJITSU AI ソリューション Zinrai ディープラーニング システム
機能説明書

発行日 2018 年 10 月