

■概要

IBM Power Systems 汎用モデル (S914 : 4core) の None GPU モデルで「IBM i の横に Driverless AI」と題して、H2O Driverless AI(以下 Driverless AI)の導入および稼働検証を実施しました。

■背景・課題

IBM Power Systems 汎用モデル (None GPU) でも以前から Driverless AI を使用して、数値データや文字データから洞察 (需要予測、不正検知など) をすることができましたが、それに加えて画像認識による画像分類ができるようになりました。(2020年7月に発表)
最初に IBM Power Systems 汎用モデルの S914 に IBM i ホストクライアント区画があり、追加で Linux クライアント区画(RHEL7.6)環境を作成して Driverless AI を導入するのか？
続いて、Driverless AI 導入後、どのように画像認識 (画像分類) を行うのか？
上記について実際に検証確認いたしました。

■製品概要

・ Driverless AI

数値データ・文字データから需要予測や不正検知のモデルを全自動で高精度に導出し、現場でポータブルに予測や検知を実行できます。



New: Driverless AIが
画像認識に対応
(2020/07~)



※Driverless AI の画像認識（画像分類）について

IBM Power Systems 汎用モデルで利用する場合、「画像ベクトライザー」の手法です。

画像認識（分類）には2つの手法があります

iguazu

手法1: Embeddings Transformer (画像ベクトライザー)

- 学習済みモデルで画像を特徴量変換
- 画像を数値ベクトルに変換
学習済みモデルのglobal average pooling層から得られる
- CPUでも実行可能
- 従来の文字・数値の表データに画像の列が入ったイメージ
- 事前トレーニング済みのモデルを使うが、モデルのファインチューニングも可能

手法2: Automatic Image Model (自動ディープラーニング)

- ディープラーニングで重みを学習
 - Tensorflowフレームワーク
- 画像認識でのAutoML
- GPU必須
- 従来の文字・数値の表データの延長ではなく、画像群のみが扱える
- ハイパーパラメータとモデルのアーキテクチャー選択を自動で行う
- 分類と回帰に対応

メリットとしては、GPU 非搭載でも利用可能なところです。

画像認識（分類）の2つの手法比較

iguazu

	Embeddings Transformer (画像ベクトライザー)	Automatic Image Model (自動ディープラーニング)
概要	事前学習済みモデルの重みで画像を数値ベクトル群に変換	画像認識モデルをディープラーニングで作成
サーバー環境	GPU不要(CPU環境でも可能)	GPU必要
ジョブの所要時間	15~30分程度	30分~数10時間
認識精度	中	高
推論パイプライン	Python, <u>MOJO</u> ※	Python

※数値/文字における機械学習モデルと同様の方法にて、画像の推論もMOJOパイプラインを使って行える。

- 現在はEmbeddings Transformerのみ。

- Automatic Image ModelはMOJOサポートなし。Pythonパイプラインは対応済

■構成イメージ

● Driverless AI 環境

使用した IBM Power Systems の仕様は以下の通り

モデル：S914(9009-41A：4core)、CPU：1core、MEM：12GB、

OS：RHEL7.6（IBM i ホスト-Linux クライアント）環境

Driverless AI：V1.9.0.6

画像認証で使ったサンプルデータ：

- ・ bird.zip（モデルデータとして、鳩、カラス、カモメなど 16 種 323 枚の鳥画像）
- ・ TEST_Bird.zip（予測用データとして、鳩、カモメの 2 種 4 枚の鳥画像）

■検証内容

● Driverless AI の導入検証（Linux 環境の認証完了してる事が前提です）

S914 の IBM i 区画の横に Linux 区画を作成して認証処理まで完了します。

その Linux 環境に Driverless AI の導入します。以下の URL を参照して実施

<https://dai-doc-jp.au-syd.mybluemix.net/install/ibm-docker.html#install-on-ibm-with-cpusl>

- ① yum install コマンドより、Docker を導入
- ② systemctl start コマンドより、Docker を開始
- ③ systemctl enable コマンドより、再起動しても自動で Docker が起動済にできる設定
- ④ docker load コマンドより、Driverless AI を導入
- ⑤ docker run コマンドより、Driverless AI の起動
- ⑥ [http://IP アドレスまたはホスト名:12345/](http://IPアドレスまたはホスト名:12345/) にアクセスしてログイン画面が表示確認

● Driverless AI の画像認証（画像分類）検証

- ① Driverless AI にログイン
- ② 「データセット」の「データの追加」-「ファイルをアップロードする」より、bird.zip のモデルデータと、TEST_Bird.zip の予測用データを取り込む
- ③ 取込み後の確認を bird.zip をクリックして「詳細」-「生データ」にて画像を確認
- ④ 確認後、再度 bird.zip をクリックして「EXPERIMENT」をを選択し、「ターゲット列を選択」で“label”選択して「EXPERIMENT の開始」を押下
- ⑤ 完了後、上部メニューから「Experiment」を選択し、作成された bird.zip をクリックしてモデルデータ分析結果が表示する。
- ⑥ 「他のデータセットで予測」をクリックし、「TEST_Bird.zip」を選択して「完了」ボタンを押下することで画像分類予測を実施する。
- ⑦ 分析が完了して「開く」を押下し、「予測結果のダウンロード」をクリックすると CSV ファイルに出力され、ベクトル数値の割合で分類されます。

■ 検証結果

・DriverlessAI の導入に関しては、「検証内容」の記載の通りで問題無く導入ができました。

・データセット画面

検索する値または日付を入力します (例: 15/09)

名前	ファイルのパス	サイズ	行	列	Status	作成日
store_demand_2017_sample.csv	...ple.csv.1570650563.5379715.bin	6KB	365	3	[クリックして操作]	2019/10/10 4:49:13
store_demand_2019-2016.csv	...s-2016.csv.1570650538.7524.bin	18MB	731K	4	[クリックして操作]	2019/10/10 4:48:58
yoshin_test.csv	...est.csv.1570634315.7218497.bin	53KB	999	12	[クリックして操作]	2019/10/10 0:18:35
yoshin_train.csv	...ain.csv.1570634067.1508299.bin	8MB	150K	12	[クリックして操作]	2019/10/10 0:14:26
TEST_Bird.zip	...Bird.zip.1570108919.687761.bin	231KB	4	1	[クリックして操作]	2019/10/3 22:21:59
bird.zip	...ird.zip.1559856069.3275193.bin	37MB	323	2	[クリックして操作]	2019/10/1 0:07:48

・画像データ確認画面

bird.zip (モデルデータサンプル)

データ詳細: bird.zip (10d30964-e394-11e9-82aa-0242ac110002)

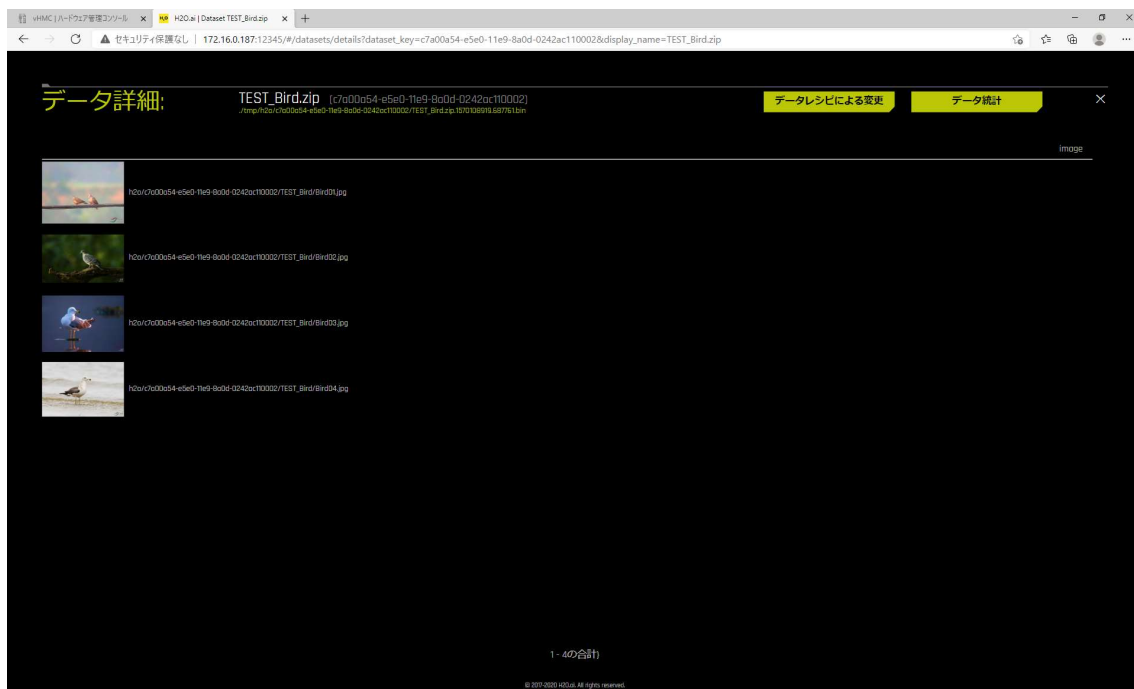
データレシビによる変更 | データ統計

Image	label
	Streptopelia
	Streptopelia
	Streptopelia
	Streptopelia
	Streptopelia
	Streptopelia
	Streptopelia
	Streptopelia

1 2 3 4 5 6 7 8 9 10 ... 15 次のページ
1 - 23の合計

TEST_Bird.zip (予測用データサンプル)

上2つは鳩、下2つはカモメです。

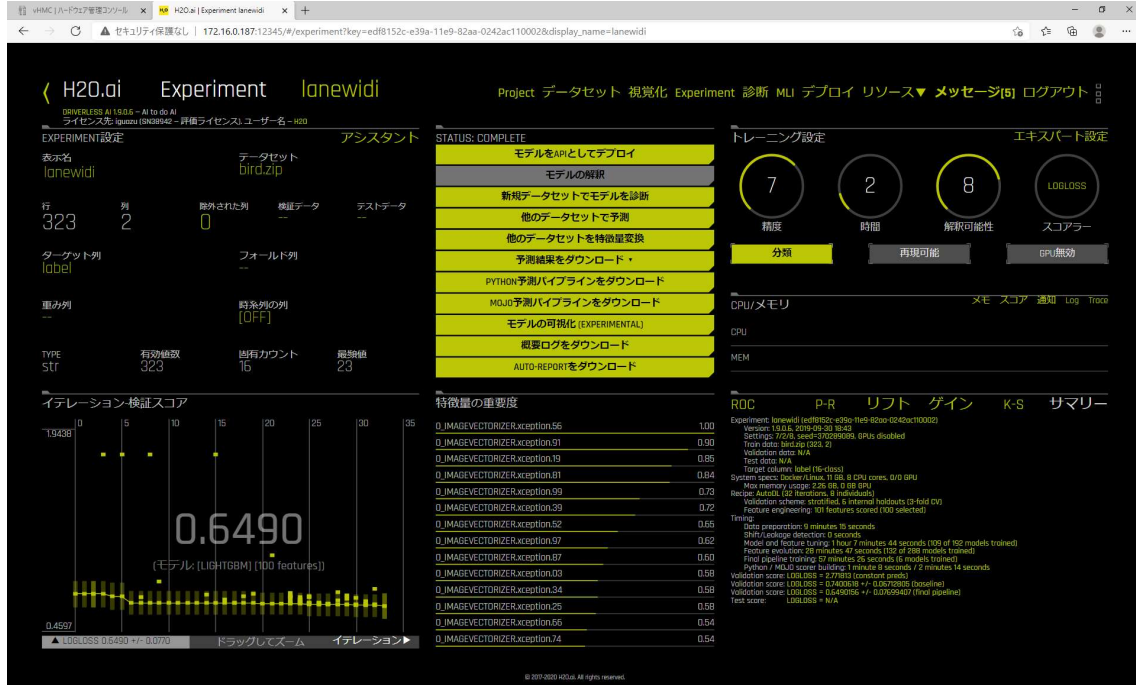


・ EXPERIMENT 指定画面 (データセット名 : bird.zip、ターゲット列名 : label)

The screenshot shows the H2O.ai 'Experiment' configuration page. The page displays various settings for an experiment named 'bird.zip'. Key settings include: 323 rows, 2 target columns, and 16 features. The 'トレーニング設定' (Training Settings) section shows a precision of 7, time of 2, and a solution count of 8. The 'エキスパート設定' (Expert Settings) section shows 'LOGLOSS' as the scoring metric. The page also includes a '客設定の詳細' (Customer Settings) section on the left and a '再実行可能' (Can be re-executed) button at the bottom.

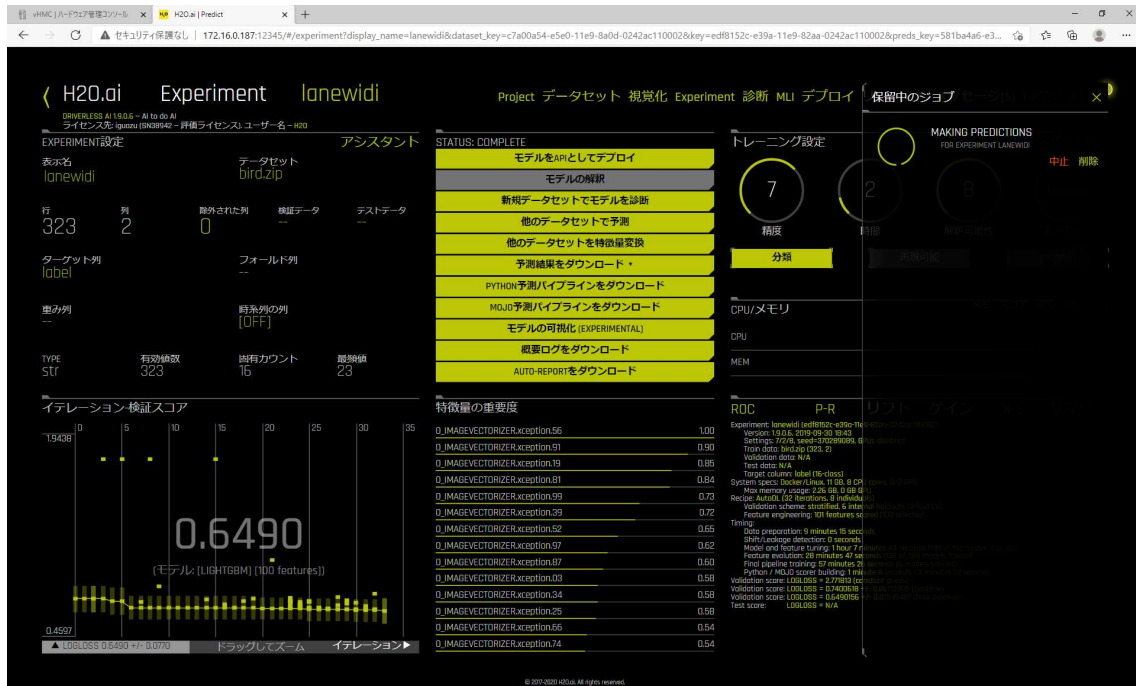
・ EXPERIMENT 処理がモデルデータ完了後の画面

※因みに完了までに 2:46:45 (約 3 時間) で処理が完了しました。



・ 「他のデータセットで予測」 処理より TEST_Bird.zip で分析中 (画面右横)

※約 1 分で結果が出ました



・処理後、CSV ファイルの内容（ベクトル数値から確率（9 割以上）で鳥の種類を分類）

image	labelAcro others (アカリ) スズメ)	labelAcro cephalus ツグク	labelAethya アヒ	labelButor us カラス)	labelCorvus us カラス)	labelDend. us カラス)	labelGallus us カラス)	labelHaliaeetus us カラス)	labelLarus us カラス)	labelMerg. us カラス)	labelNum. us カラス)	labelPhasianus us カラス)	labelPipilo us カラス)	labelPyronotus us カラス)	labelStreptopelia us カラス)	labelUring. us カラス)
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird01.jpg	0.003772	0.005891	0.002769	0.004687	0.004855	0.001777	0.001882	0.001466	0.004318	0.001262	0.002352	0.001506	0.002585	0.000813	0.938444	0.0038
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird02.jpg	0.019748	0.006331	0.002741	0.004395	0.006128	0.006266	0.005123	0.010687	0.004396	0.004637	0.004267	0.003201	0.006852	0.005896	0.907153	0.003078
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird03.jpg	0.00167	0.001819	0.001268	0.000865	0.002275	0.002044	0.005175	0.002253	0.950293	0.005507	0.004493	0.00139	0.002387	0.001887	0.001938	0.006304
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird04.jpg	0.00251	0.005394	0.003159	0.002707	0.003939	0.002244	0.005674	0.003012	0.943614	0.005344	0.003781	0.002385	0.0043	0.002251	0.006735	0.003542

※割合設定を9割にして、上2つが「鳩」、下2つが「カモメ」という結果で分類。

指定の値より大きい

次の値より大きいセルを書式設定:

0.9

書式: 濃い赤の文字、明るい赤の背景

OK キャンセル

image	labelAcro others (アカリ) スズメ)	labelAcro cephalus ツグク	labelAethya アヒ	labelButor us カラス)	labelCorvus us カラス)	labelDend. us カラス)	labelGallus us カラス)	labelHaliaeetus us カラス)	labelLarus us カラス)	labelMerg. us カラス)	labelNum. us カラス)	labelPhasianus us カラス)	labelPipilo us カラス)	labelPyronotus us カラス)	labelStreptopelia us カラス)	labelUring. us カラス)
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird01.jpg	0.003772	0.005891	0.002769	0.004687	0.004855	0.001777	0.001882	0.001466	0.004318	0.001262	0.002352	0.001506	0.002585	0.000813	0.938444	0.0038
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird02.jpg	0.019748	0.006331	0.002741	0.004395	0.006128	0.006266	0.005123	0.010687	0.004396	0.004637	0.004267	0.003201	0.006852	0.005896	0.907153	0.003078
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird03.jpg	0.00167	0.001819	0.001268	0.000865	0.002275	0.002044	0.005175	0.002253	0.950293	0.005507	0.004493	0.00139	0.002387	0.001887	0.001938	0.006304
h2o/c7a00a64-e5e0-11e9-ba04-c242ac110002/TEST_Bird/Bird04.jpg	0.00251	0.005394	0.003159	0.002707	0.003939	0.002244	0.005674	0.003012	0.943614	0.005344	0.003781	0.002385	0.0043	0.002251	0.006735	0.003542

実際は NoneGPU 環境では処理上、以下のような画面上に数値表現する機能はありません。
※以下は画像と数値をマージ加工したものです



■所感

IBM Power Systems 汎用モデル (S914:None GPU) で検証しましたがモデルデータが出来た後の予測分析もストレスなく分析結果が得られました。

S914 の 4core モデルは、最初から 4core 分がアクティベーションされています。

IBM i が 1core で利用している場合、3core 分余ったリソースで Linux 区画を構築して Driverless AI を利用することで需要予測や不正検知や画像分類といった付加価値が付けられるのではないのでしょうか？

是非、IBM i の横から AI を始めてみませんか？