マシンラーニングとマンモグラフィー

この記事は、インテル® デベロッパー・ゾーンに公開されている「Machine Learning and Mammography」の日本語参考訳です。

はじめに

この記事では、組織像で乳癌の一種である浸潤性腺乳管癌 (IDC) (英語)[1] を検出できるように、既存のディープラーニング・テクノロジーを使用して人工知能 (AI) を訓練する方法を紹介します。具体的には、陰性および陽性の組織像のデータセットで TensorFlow* (英語)[3] と転移学習 (英語)[4] を使用して、畳み込みニューラル・ネットワーク (英語)[2] を訓練する方法を示します。AI を使用して IDC を検出する方法を示すだけでなく、モノのインターネット (IoT) と AI を組み合わせることで、医療業界で使用できる自動システムを作成する方法を示します。

乳癌は女性にとって最も一般的な癌の 1 つであり、多くの関心を集め続けています。2018 年には、アメリカだけで推定 266,120 件の新たな診断が下されると予想されています。AI を使用することで、医療スタッフがマンモグラフィー画像を手動で検査する工程を大幅に減らし、時間とコストを軽減するだけでなく、最終的には命を救うことにつながります。この記事では、インテルのテクノロジーを使用して IDC を検出可能なディープラーニング・ニューラル・ネットワークを作成する方法を紹介します。

IDC 分類器の概要

IDC 分類器を作成するため、インテル® AI DevCloud (英語)[5] を使用してニューラル・ネットワークを訓練し、インテル® Movidius™ (英語) 製品 [6] を使用してエッジで推論を行い、UP Squared (英語)[7] デバイスを使用して API を介して訓練済みモデルにアクセスできるようにします。そして、Raspberry Pi* (英語)[8] デバイスを使用する IoT 接続のアラームシステムを使って、IoT JumpWay* (英語)[9] を介した IoT と AI を組み合わせてインテリジェントな自動医療システムを作成します。

このプロジェクトは、私が長年開発してきた TASS (英語)[10] というコンピューター・ビジョン・プロジェクトに端を発しています。TASS は、さまざまな手法、フレームワーク、ソフトウェア開発キット (SDK) を使用して実装されているオープンソースの顔認証プロジェクトです。

浸潤性腺乳管癌

IDC は乳癌の最も一般的な形態の 1 つです。癌が乳管で発症し周囲の組織に浸潤します。米国癌協会によると、すべての乳癌診断の約 80% が IDC であり、アメリカだけで年間 180,000 人の女性が IDC と診断されています。

畳み込みニューラル・ネットワーク

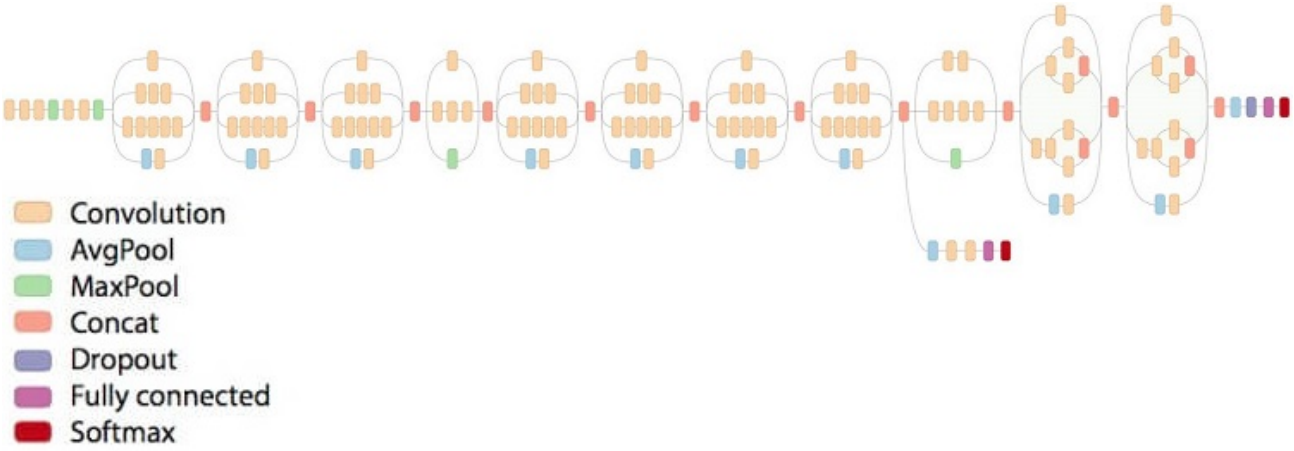

図 1. Inception v3 アーキテクチャー (出典 (英語))

畳み込みニューラル・ネットワーク (CNN) は、ディープラーニング (英語)[11] ニューラル・ネットワークの一種です。CNN はコンピューター・ビジョンで広く採用されており、過去数年の間にコンピューター・ビジョンのパフォーマンスは、従来のニューラル・ネットワークと比較して格段に向上しました。しかし、訓練時間と正解率のトレードオフがあることが報告 (英語)[12] されています。

転移学習

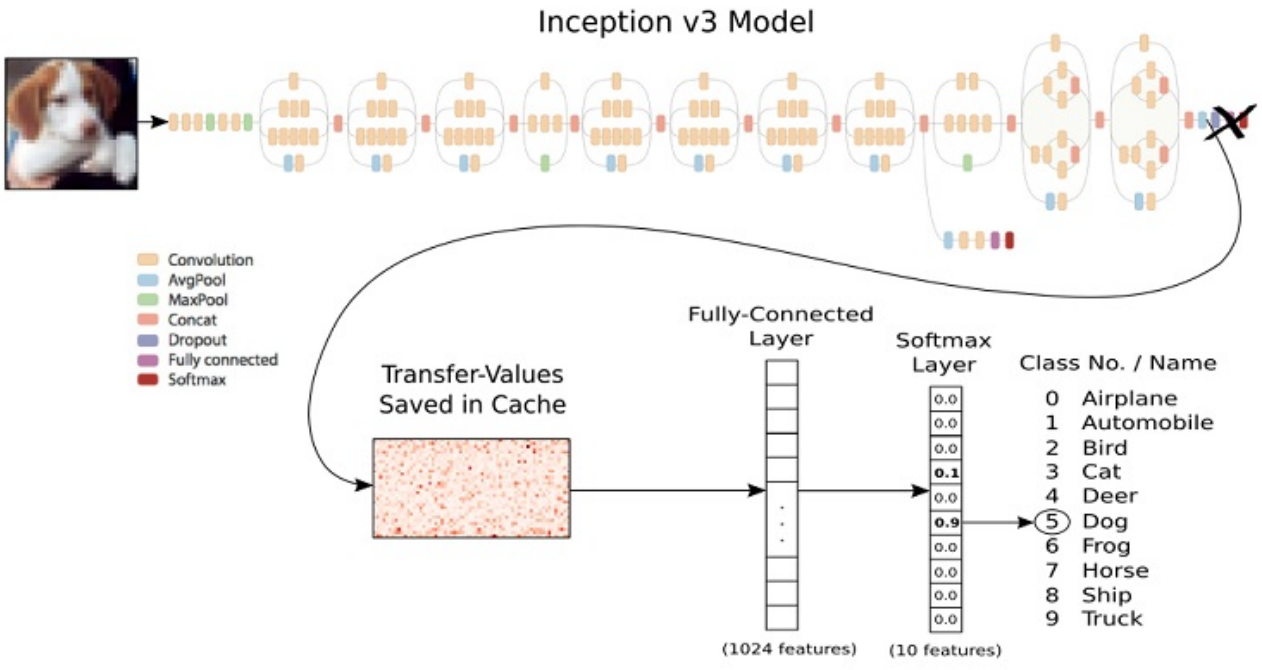

図 2. Inception v3 の移転学習 (出典 (英語))

移転学習は、既存のモデルの最後の層を再訓練することで、訓練時間を短縮するだけでなく、必要なデータセットのサイズも縮小します。移転学習に使用可能な最も有名なモデルの 1 つは、Google* の Inception v3 モデル (英語)[13] です。このモデルは、非常に強力なデバイス上で、1,001 クラスの数千個の画像を使用して訓

練されています。最後の層を再訓練できるため、モデルが最初の訓練で習得した知識を保持して、より小さなデータセットに適用することで、広範な訓練や強力な計算能力を必要とせず高い正解率の分類が可能です。TASS のあるバージョンでは、Raspberry Pi* 3 デバイス上で移転学習を使用して Inception v3 モデルを再訓練しているため、移転学習を理解する上で役立つでしょう。

インテル® AI DevCloud

インテル® AI DevCloud は、マシンラーニング・モデルとディープラーニング・モデルを訓練するプラットフォームであり、インテル® Xeon® スケーラブル・プロセッサー・ベースのサーバーで構成されたクラスターです。このプラットフォームは、TensorFlow*、Caffe*、Keras*、Theano*、インテル® Distribution for Python* など、さまざまなフレームワークとツールを無料で提供します。グラフィックス・プロセシング・ユニット (GPU) は非常に高価であり、無料で利用できるインテル® AI DevCloud は、マシンラーニング・モデルやディープラーニング・モデルの訓練方法を習得したい人々に最適です。

このプロジェクトでは、データの並べ替え、モデルの訓練、そして評価にインテル® AI DevCloud を使用します。この記事を補完するため、プロジェクト全体を再現できるように、こちら (英語) でチュートリアルとすべてのコードを提供しています。

インテル® Movidius™ ニューラル・コンピュート・スティック

インテル® Movidius™ ニューラル・コンピュート・スティックは、低電力のエッジデバイス上でコンピューター・ビジョン・モデルの推論プロセスを強化する新しいハードウェアです。インテル® Movidius™ 製品は、Raspberry Pi* や UP Squared などのデバイスに装着可能な USB です。デバイスからインテル® Movidius™ 製品へ処理を移行することで分類プロセスを高速化します。開発者は、既存の TensorFlow* や Caffe* スクリプトを使用してモデルを訓練し、開発マシンにインテル® Movidius™ ニューラル・コンピュート SDK をインストールすることで、インテル® Movidius™ 製品と互換性のあるグラフをコンパイルできます。軽量の API は低電力のデバイスにインストールして、インテル® Movidius™ 製品を介して推論を行うことができます。

コードを記述する

それでは、乳癌細胞を陰性と陽性に分類するコンピューター・ビジョン・プログラムの作成に取り掛かりましょう。ここでは、インテル® Movidius™ 製品向けのグラフの訓練とコンパイル手順を示します。IoT 接続デバイスを含む完全なチュートリアルは GitHub* リポジトリー (英語) を参照してください。先に進む前に、リポジトリーの手順に従って IoT JumpWay* デバイスをセットアップしてください。これは分類テストの前に行っておく必要があります。

インテル® Movidius™ ニューラル・コンピュート SDK を開発デバイスにインストールする

最初に、**インテル® Movidius™ ニューラル・コンピュート SDK** を開発デバイスにインストールする必要があります。この SDK は、訓練済みモデルを インテル® Movidius™ 製品と互換性のある形式に変換します。

```
$ mkdir -p ~/workspace
$ cd ~/workspace
$ git clone https://github.com/movidius/ncsdk.git
$ cd ~/workspace/ncsdk
$ make install
```

次に、インテル® Movidius™ 製品をデバイスに装着して、次のコマンドを実行します。

```
$ cd ~/workspace/ncsdk
$ make examples
```

インテル® Movidius™ ニューラル・コンピュート SDK を推論デバイスにインストールする

次に、**インテル® Movidius™ ニューラル・コンピュート SDK** を Raspberry Pi* 3 や UP Squared デバイスにインストールします。分類器は、この SDK を使用してローカルの画像または API を介して受け取った画像を推論します。インテル® Movidius™ 製品が装着されていることを確認します。

```
$ mkdir -p ~/workspace
$ cd ~/workspace
$ git clone https://github.com/movidius/ncsdk.git
$ cd ~/workspace/ncsdk/api/src
$ make
$ sudo make install
$ cd ~/workspace
$ git clone https://github.com/movidius/ncappzoo
$ cd ncappzoo/apps/hello_ncs_py
$ python3 hello_ncs.py
```

訓練データを準備する

このチュートリアルでは、Kaggle* の乳癌組織像で IDC を予測する (英語) データセットを使用しますが、任意のデータセットにも使用できます。このチュートリアルで使用する陽性と陰性の画像は **model/train** ディレクトリーにあります。使用するデータセットが決まったら、**model/train** ディレクトリーに配置します。サブディレクトリー名は整数にする必要があります。このチュートリアルでは、陰性と陽性を表す 0 と 1 を使用します。テストでは、陽性と陰性の例を 4400 件ずつ使用し、訓練全体の正解率は 0.8596 (「訓練結果」を参照)、正検出の平均信頼度は 0.96 になりました。オリジナルの画像サイズは 50px x 50px でしたが、Inception v3 は 299px x 299px で訓練されているため、画像サイズを 299px x 299px に変更しました。画像が元のサイズであることが望ましいため、さまざまなデータセットを試して、結果にどのような影響があるか確認するとよいでしょう。

パラメーターをチューニングする

ネットワークの設定は、**model/confs.json** ファイルで分類器の設定を編集することでいつでもチューニングできます。

```
"ClassifierSettings":{
    "dataset_dir":"model/train/",
    "log_dir":"model/_logs",
    "log_eval":"model/_logs_eval",
    "classes":"model/classes.txt",
    "labels":"labels.txt",
    "labels_file":"model/train/labels.txt",
    "validation_size":0.3,
    "num_shards":2,
    "random_seed":50,
    "tfrecord_filename":"200label",
    "file_pattern":"200label_%s_*.tfrecord",
    "image_size":299,
    "num_classes":2,
```

```
    "num_epochs":60,
    "dev_cloud_epochs":60,
    "test_num_epochs":1,
    "batch_size":10,
    "test_batch_size":36,
    "initial_learning_rate":0.0001,
    "learning_rate_decay_factor":0.96,
    "num_epochs_before_decay":10,
    "NetworkPath":"",
    "InceptionImagePath":"model/test/",
    "InceptionThreshold": 0.54,
    "InceptionGraph":"igraph"
}
```

訓練を開始する

次のファイルとフォルダーをインテル® AI DevCloud にアップロードします。

```
model
tools
DevCloudTrainer.ipynb
DevCloudTrainer.py
Eval.py
```

アップロードしたら、**DevCloudTrainer.ipynb** の手順に従って、データの並べ替え、モデルの訓練、モデルの評価を行います。

結果を訓練する

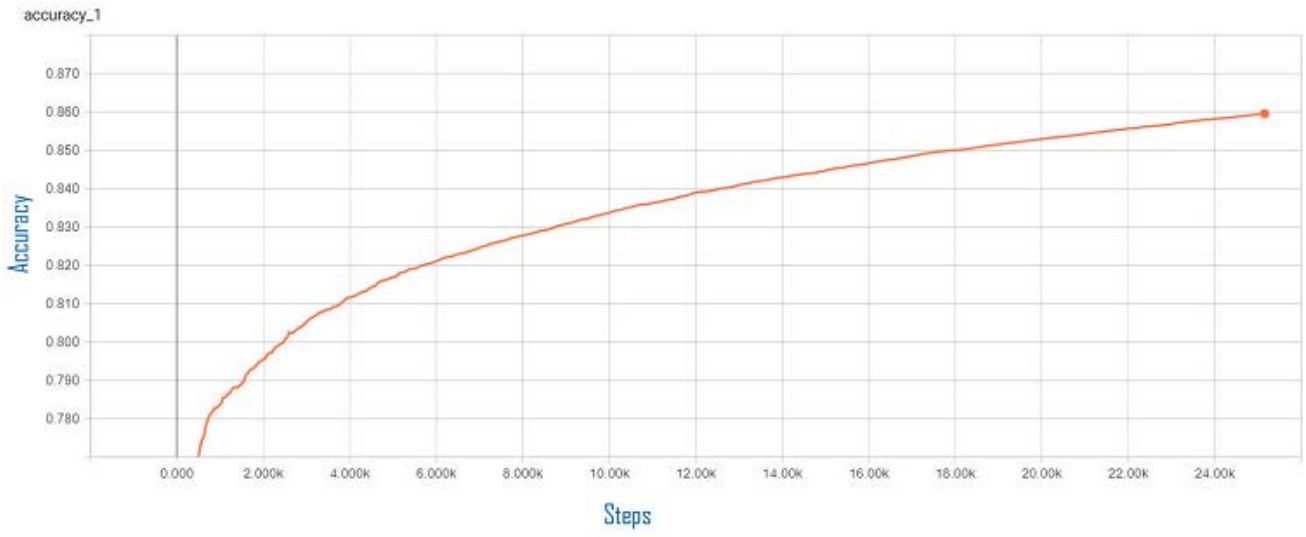

図 3. TensorBoard* の訓練の正解率

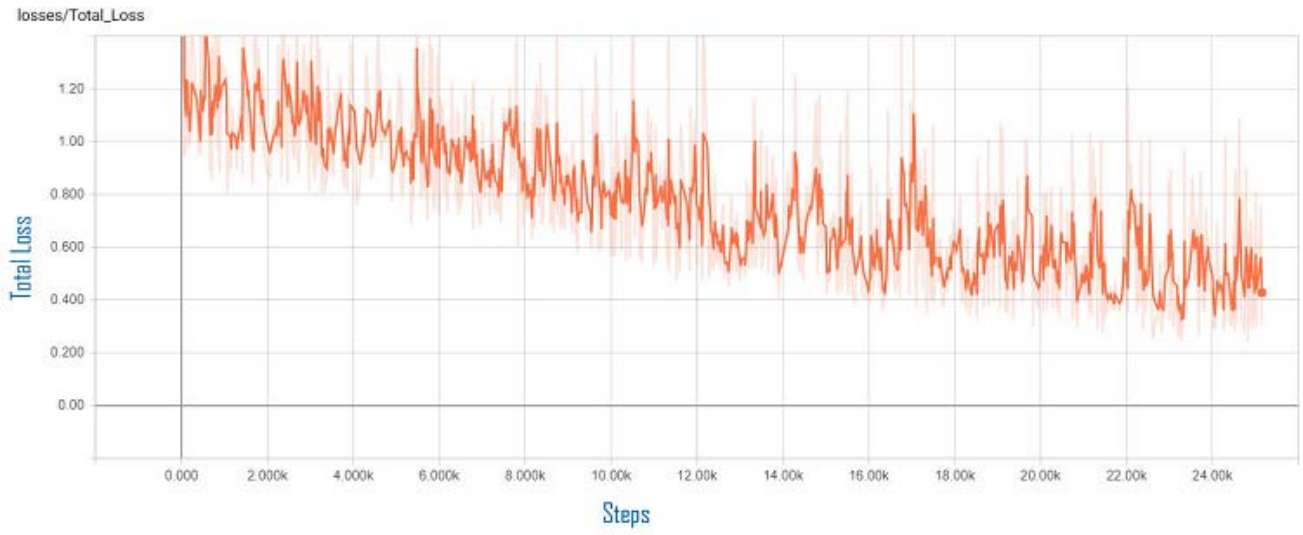

図 4. 訓練誤差

モデルを評価する

インテル® AI DevCloud 上での訓練が完了したら、評価ジョブを実行します。

評価結果

```
INFO:tensorflow:Global Step 1: Streaming Accuracy: 0.0000 (2.03 sec/step)
INFO:tensorflow:Global Step 2: Streaming Accuracy: 0.8889 (0.59 sec/step)
INFO:tensorflow:Global Step 3: Streaming Accuracy: 0.8750 (0.67 sec/step)
INFO:tensorflow:Global Step 4: Streaming Accuracy: 0.8981 (0.65 sec/step)
INFO:tensorflow:Global Step 5: Streaming Accuracy: 0.8681 (0.76 sec/step)
INFO:tensorflow:Global Step 6: Streaming Accuracy: 0.8722 (0.64 sec/step)
INFO:tensorflow:Global Step 7: Streaming Accuracy: 0.8843 (0.64 sec/step)

----------------------------------------------------------------------

INFO:tensorflow:Global Step 68: Streaming Accuracy: 0.8922 (0.81 sec/step)
INFO:tensorflow:Global Step 69: Streaming Accuracy: 0.8926 (0.70 sec/step)
INFO:tensorflow:Global Step 70: Streaming Accuracy: 0.8921 (0.63 sec/step)
INFO:tensorflow:Global Step 71: Streaming Accuracy: 0.8929 (0.84 sec/step)
INFO:tensorflow:Global Step 72: Streaming Accuracy: 0.8932 (0.75 sec/step)
INFO:tensorflow:Global Step 73: Streaming Accuracy: 0.8935 (0.61 sec/step)
INFO:tensorflow:Global Step 74: Streaming Accuracy: 0.8942 (0.67 sec/step)
INFO:tensorflow:Final Streaming Accuracy: 0.8941
```

この評価結果は、Final Streaming Accuracy が 0.8941 であることを示しています。

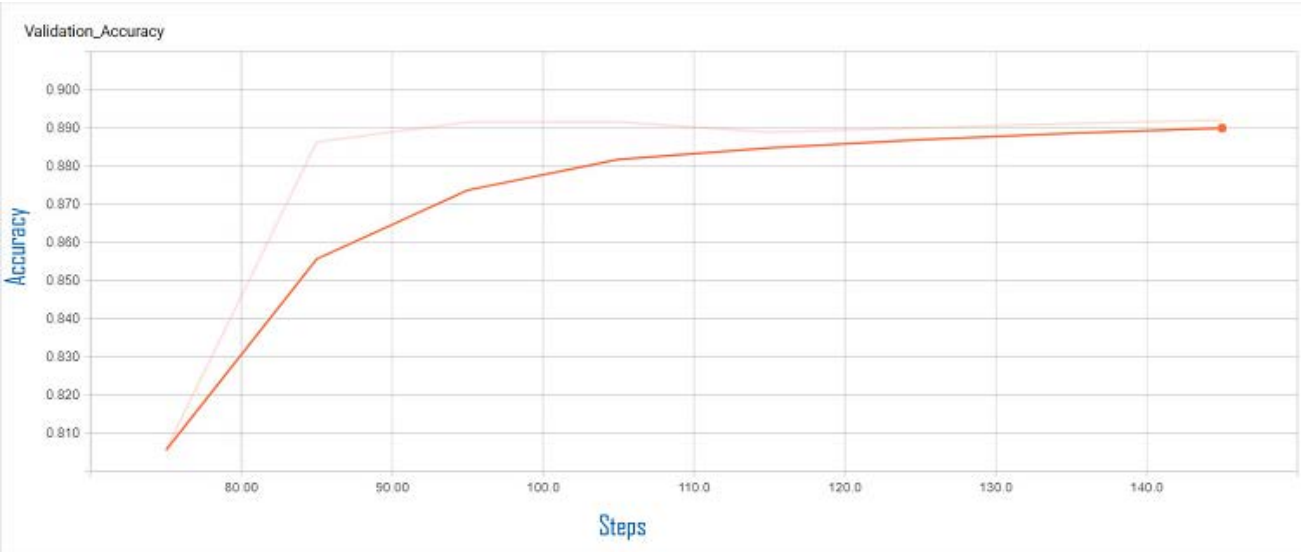

図 5. 評価の正解率

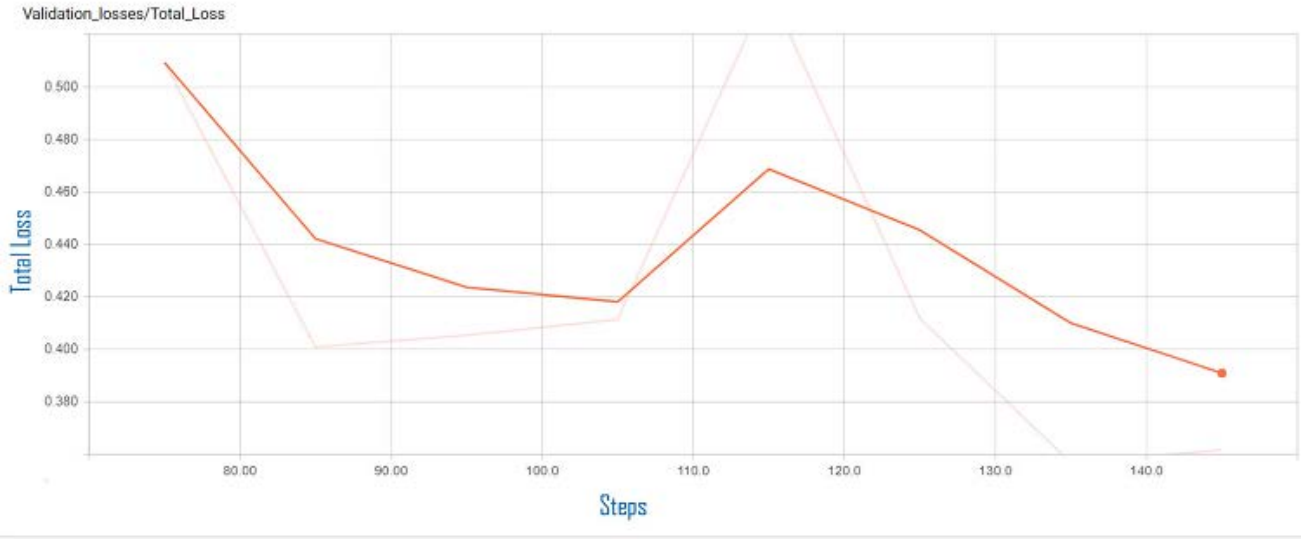

図 6. 評価誤差

モデルをダウンロードする

訓練が完了したら、**model/DevCloudIDC.pb** と **model/classes.txt** を開発マシンの **model** ディレクトリーにダウンロードする必要があります。インテル® Movidius™ 製品がセットアップされ、接続済みであることを確認してから、開発マシンで次のコマンドを実行します。

```
$ cd ~/IoT-JumpWay-Intel-Examples/master/Intel-Movidius/IDC-Classification
$ ./DevCloudTrainer.sh
```

DevCloudTrainer.sh の内容を以下に示します。

```
#IDC Classification Trainer
mvNCCompile model/DevCloudIDC.pb -in=input -on=InceptionV3/Predictions/Softmax -o
igraph
python3.5 Classifier.py InceptionTest
```

1. インテル® Movidius™ 製品向けにモデルをコンパイルします。
2. テストします。

不明な画像でテストする

シェルスクリプトの実行が完了すると、テストプログラムが開始します。ここで使用した例では、0 と 1 (IDC 陰性と IDC 陽性) の 2 つのクラスがあります。AI は画像が IDC 陽性ではないと判断すると 0 に分類し、IDC 陽性であると判断すると 1 に分類します。

```
-- Loaded Test Image model/test/negative.png

-- DETECTION STARTING
-- STARTED: :  2018-04-24 14:14:26.780554

-- DETECTION ENDING
-- ENDED:  2018-04-24 14:14:28.691870
-- TIME: 1.9114031791687012

*******************************************************************************
inception-v3 on NCS
*******************************************************************************
0 0 0.9873
1 1 0.01238
*******************************************************************************

-- Loaded Test Image model/test/positive.png

-- DETECTION STARTING
-- STARTED: :  2018-04-24 14:14:28.699254

-- DETECTION ENDING
-- ENDED:  2018-04-24 14:14:30.577683
-- TIME: 1.878432035446167ß

TASS Identified IDC with a confidence of 0.945

-- Published to Device Sensors Channel

*******************************************************************************
inception-v3 on NCS
*******************************************************************************
1 1 0.945
0 0 0.05542
*******************************************************************************

-- INCEPTION V3 TEST MODE ENDING
-- ENDED:  2018-04-24 14:14:30.579247
-- TESTED:  2
-- IDENTIFIED:  1
-- TIME(secs): 3.984593152999878
```

開発マシン上でテストすると、上記のような結果が得られるはずです。上記の結果では、プログラムが陰性と陽性を正しく分類していることが分かります。次に、エッジ上でテストしてみましょう。

エッジ上での推論

すべての訓練とテストが完了したら、**Server.py** と **Client.py** を使用して API を処理するサーバーをセットアップします。

次の手順は、サーバーをセットアップして、陽性と陰性の予測をテストするのに役立ちます。

1. 乳癌組織像で IDC を予測する (英語) データセットを使用すると、**positive.png** および **negative.png** を使用できます。これらは元々、このデータセットに含まれていたものです。そうでない場合は、テストに使用したデータセットから陽性と陰性の例を選択して、これらの画像と置き換えます。
2. サーバーは、localhost で開始するように設定されています。この設定を変更するには、**Server.py** の行 281 と **Client.py** の行 38 でホスト名を変更します。サーバーを実行したままにして外部からアクセスする場合は、Let's Encrypt* などでセキュリティー保護すべきです。
3. サーバーに使用する UP Squared や Raspberry Pi* 3 デバイスに次のファイルとフォルダーをアップロードします。

   ```
   model/test/
   model/classes.txt
   model/confs.json
   tools
   igraph
   Server.py
   ```

4. ターミナルを開いて、Server.py のフォルダーに移動し、次のコマンドを実行します。

   ```
   $ python3.5 Server.py
   ```

5. サーバーが開始され、分類する画像を受け取るため待機します。
6. 上記の手順をすべて完了したら、次のコマンドを実行して開発マシンでクライアントを開始します。

   ```
   $ python3.5 Client.py
   ```

陽性と陰性の組織像が Raspberry Pi* 3 または UP Squared デバイスに送信され、予測が返されます。

```
!! Welcome to IDC Classification Client, please wait while the program
initiates !!

-- Running on Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609]

-- Imported Required Modules
-- IDC Classification Client Initiated

{'Response': 'OK', 'ResponseMessage': 'IDC Detected!', 'Results': 1}
{'Response': 'OK', 'ResponseMessage': 'IDC Not Detected!', 'Results': 0}
* Running on http://0.0.0.0:7455/ (Press CTRL+C to quit)

-- IDC CLASSIFIER LIVE INFERENCE STARTING
-- STARTED: :  2018-04-24 14:25:36.465183

-- Loading Sample
-- Loaded Sample
-- DETECTION STARTING
-- STARTED: :  2018-04-24 14:25:36.476371
```

```
-- DETECTION ENDING
-- ENDED:  2018-04-24 14:25:38.386121
-- TIME: 1.9097554683685303

TASS Identified IDC with a confidence of 0.945

-- Published: 2
-- Published to Device Warnings Channel

-- Published: 3
-- Published to Device Sensors Channel

********************************************************************************
inception-v3 on NCS
********************************************************************************
1 1 0.945
0 0 0.05542
********************************************************************************

-- IDC CLASSIFIER LIVE INFERENCE ENDING
-- ENDED:  2018-04-24 14:25:38.389217
-- TESTED:  1
-- IDENTIFIED:  1
-- TIME(secs): 1.9240257740020752

192.168.1.40 - - [24/Apr/2018 14:25:38] "POST /api/infer HTTP/1.1" 200 -

-- IDC CLASSIFIER LIVE INFERENCE STARTING
-- STARTED: :  2018-04-24 14:25:43.422319

-- Loading Sample
-- Loaded Sample
-- DETECTION STARTING
-- STARTED: :  2018-04-24 14:25:43.432647

-- DETECTION ENDING
-- ENDED:  2018-04-24 14:25:45.310354
-- TIME: 1.877711534500122

-- Published: 4
-- Published to Device Warnings Channel

-- Published: 5
-- Published to Device Sensors Channel

********************************************************************************
inception-v3 on NCS
********************************************************************************
0 0 0.9873
1 1 0.01238
********************************************************************************

-- IDC CLASSIFIER LIVE INFERENCE ENDING
-- ENDED:  2018-04-24 14:25:45.313174
-- TESTED:  1
-- IDENTIFIED:  0
-- TIME(secs): 1.89084792137146

192.168.1.40 - - [24/Apr/2018 14:25:45] "POST /api/infer HTTP/1.1" 200 -
```

ここでは、UP Squared デバイス上でインテル® Movidius™ 製品を使用していますが、分類の正解率は開発マシン (NVIDIA* GeForce* GTX 750 Ti 搭載の Linux* デバイス) とほぼ同じです。分類プロセスにかかった時間がわずかに異なるだけです。また興味深いことに、上記の結果のほうが私の GPU でモデルを訓練するよりも正確です。

IoT の接続性

IoT デバイスのセットアップは、GitHub* リポジトリーのチュートリアルに従って行うことができます。ここでは、プロジェクトが何を行い、提供される概念実証をほかの医療アプリケーションでどのように使用できるかについて詳しく説明します。

ここでは、Raspberry Pi* デバイスを使用して IoT 接続のアラームシステムを作成します。セットアップが完了すると、サーバーに送信された画像の分類結果が、Raspberry Pi* デバイスと通信する IoT 上でアクションをトリガーします。このサンプルでは、癌が検出された場合、赤い LED がオンになりブザーが鳴ります。癌が検出されなかった場合は、青い LED がオンになります。これは非常に単純な概念実証ですが、医療スタッフの負担を軽減し、うまくいけば早期の正確な検出によって命を救うことができる強力なアプリケーションの可能性を示しています。

関連情報 (英語)

1. 浸潤性腺乳管癌
2. 畳み込みニューラル・ネットワーク
3. TensorFlow*
4. 転移学習
5. インテル® AI DevCloud
6. インテル® Movidius™ 製品
7. UP²
8. Raspberry Pi*
9. IoT JumpWay*
10. TASS
11. ディープラーニング
12. 移動ロボット工学におけるディープ・ニューラル・ネットワークと従来のビジョン・アルゴリズムの比較
13. コンピューター・ビジョン向け Inception アーキテクチャーの再考

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。