# Sony Neural Network Console

# 目次

[Sony Neural Network Console](#_gjdgxs)

[目次](#_30j0zll)

[はじめに](#_3znysh7)

[インストール](#_2et92p0)

[初回起動時](#_tyjcwt)

[NNCで画像認識を行う](#_3dy6vkm)

[機械学習の流れ](#_1t3h5sf)

[プロジェクトの作成](#_4d34og8)

[データの前処理](#_2s8eyo1)

[データの用意](#_17dp8vu)

[NNCによる前処理](#_3rdcrjn)

[データセットの設定](#_26in1rg)

[ニューラルネットワークの構築](#_lnxbz9)

[学習](#_35nkun2)

[評価](#_1ksv4uv)

[より難しい認識を行う](#_okxu86pci1ny)

[実際に学習する](#_3cwg42jdpyt9)

[1.プロジェクトの作成](#_62lbn21lz2ki)

[2.前処理](#_krx5rv8as3nw)

[3.ニューラルネットワークの構築](#_k17gx5onk4dk)

[4.学習&評価](#_mt8fk4zen8pd)

[最適なニューラルネットワークを探す](#_b6tqy9fyc524)

[学習開始前の設定](#_m3i75ja2jh0)

[学習の結果を確認する](#_ub7tkxwhdbjz)

[Pythonで学習結果を扱う](#_176rc2epct99)

[1.ニューラルネットワークのPythonコードをコピーする](#_40gntjg58ejc)

[2.学習済みデータをコピーする](#_ompe6sldh9i3)

[3.ライブラリのインストール](#_4eac0zhhpe49)

# 

# はじめに

このドキュメントではSony Neural Network Console(以下NNCと略)の大まかな利用方法を書いていきます。セットアップに必要な情報は以下の手順を参考にしてください。

### インストール

1. <https://dl.sony.com/ja/> にアクセス
2. 「Windowsアプリではじめる」からサイトにしたがってダウンロード
3. ダウンロードしたneural\_network\_console\_xxx.zip をデスクトップに解凍する。（xxxはバージョン情報）
4. コンピュータにVisual Studio 2015 Visual C++Redistribution Packageが入っていない場合は<https://www.microsoft.com/ja-JP/download/details.aspx?id=52685> からパッケージをダウンロードしてインストールする

このドキュメントより詳細な情報は以下のマニュアルを参考にしてください。

マニュアル：

（オンライン）<https://support.dl.sony.com/docs-ja/>

（オフライン）neural\_network\_console\_xxx/manual\_ja.pdf

### 初回起動時

* インストールが終わったらneural\_network\_console.exeをクリックしてNNCを起動します
* Visual Studio 2015 Visual C++Redistribution Packageが入っているか？と尋ねられますが、インストールしてあるのならば問題ありません
* その後ライセンス条項が表示されます。「I Agree」→「Apply」でOK

# NNCで画像認識を行う

### 機械学習の流れ

NNCに機械学習を行わせる場合、大きく以下のような手順になります。

1. プロジェクトの作成
2. データの前処理
3. ニューラルネットワークの構築
4. 学習
5. 評価

この手順にしたがってこれ以降機械学習を行っていきます。

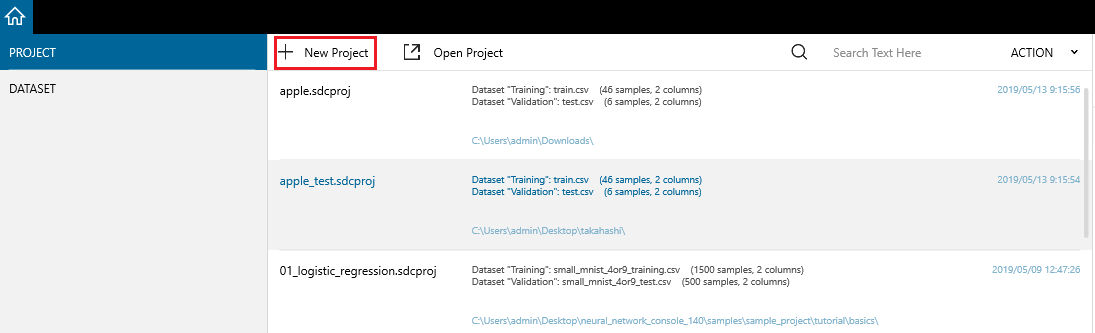
今回は画像認識を目標として、問題設定は次のようにします。

**入力：「リンゴ」か「地球」の画像**

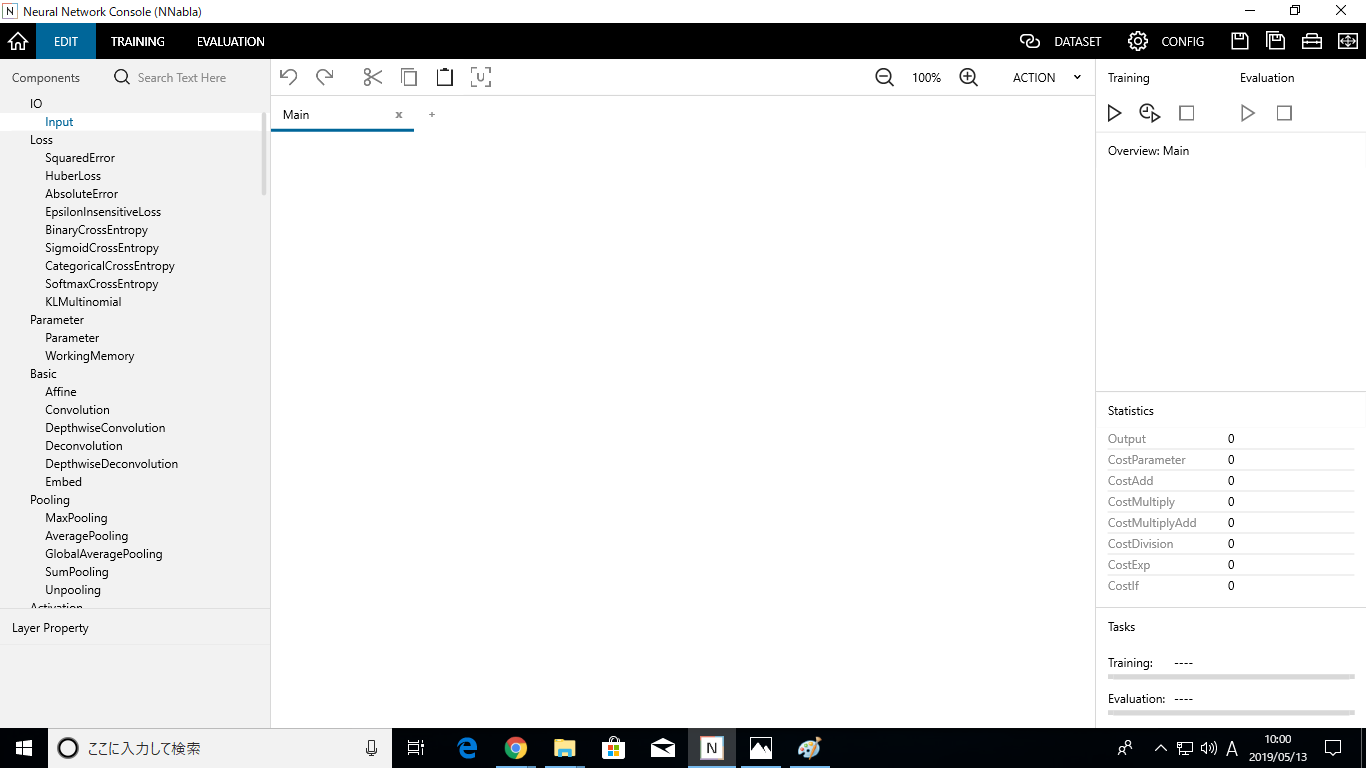
**出力：「この画像はリンゴである」または「この画像は地球である」**

### プロジェクトの作成

ホーム画面からNew Projectをクリックしてプロジェクトを作成します



以下のような画面に移動すればプロジェクトが作成されています。



### データの前処理

#### データの用意

前処理では学習させたい対象についてデータを用意します。なのでリンゴと地球の画像を大量に用意します。各ファイルは

**dataset/**

**apple/**

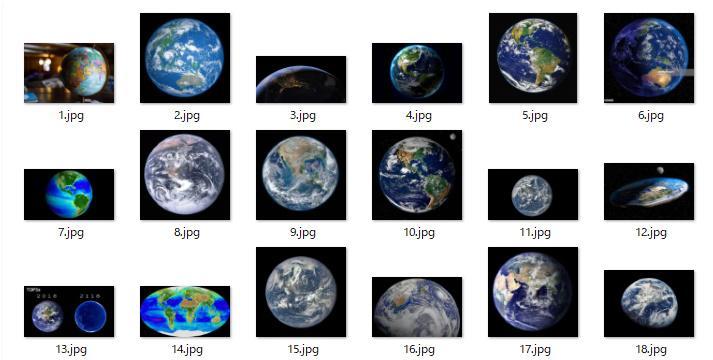
**earth/**

のようにフォルダへまとめておきます。

**dataset/**

****

**dataset/earth/**

****

対応しているフォーマットは.png、.jpg、.jpeg、.gif、.bmp、.tifです。

注：画像のファイル名は英数字のみにしておく必要があります。

また、前処理後のデータを保存するために空のディレクトリを作成しておく必要があります。ここではoutput/としてdataset/と同じ階層に作成します。

**output/**

**dataset/**

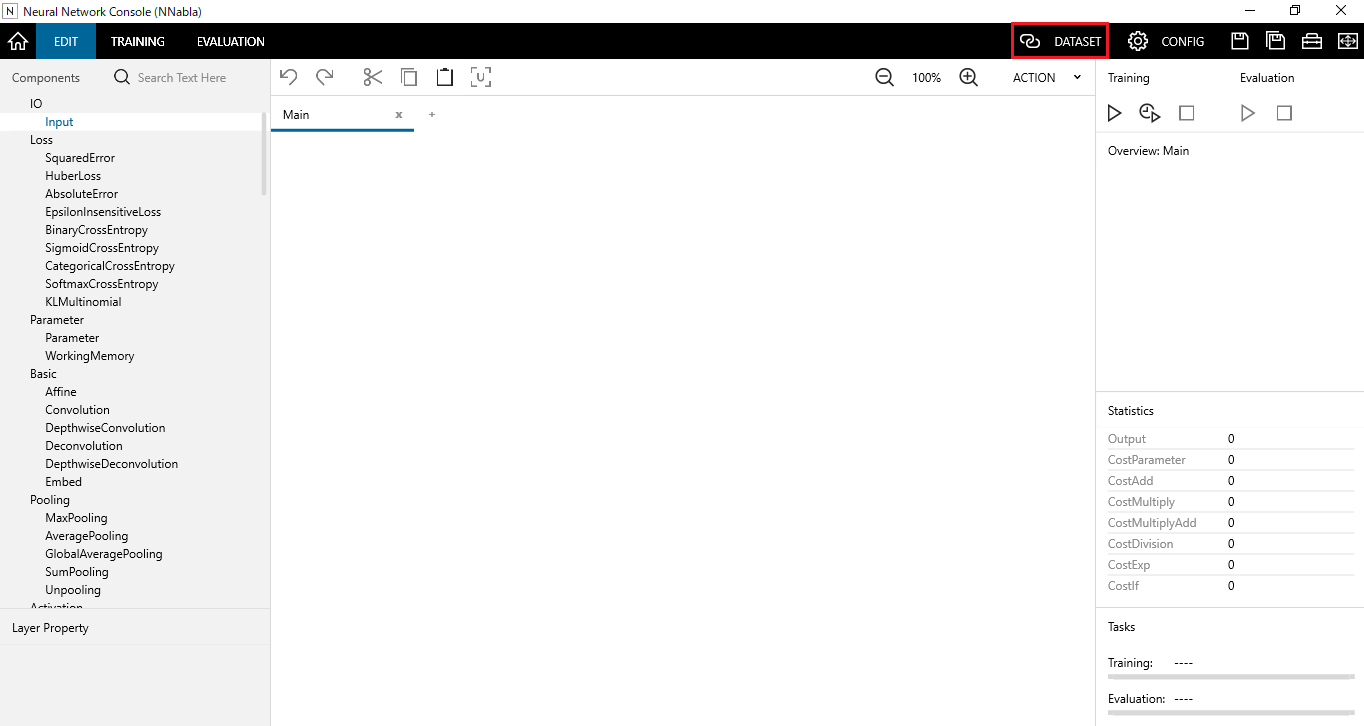
**apple/**

**earth/**

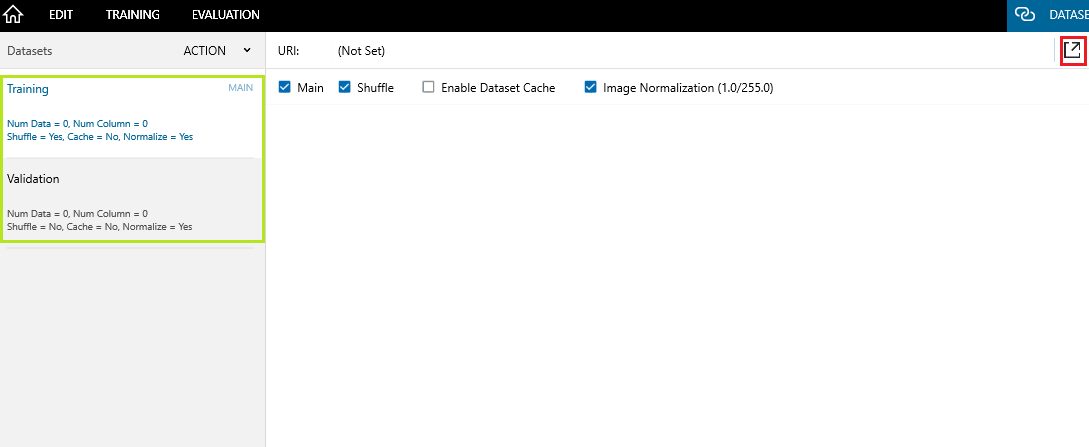
最終的に作業用のディレクトリは以上のようになります。

#### NNCによる前処理

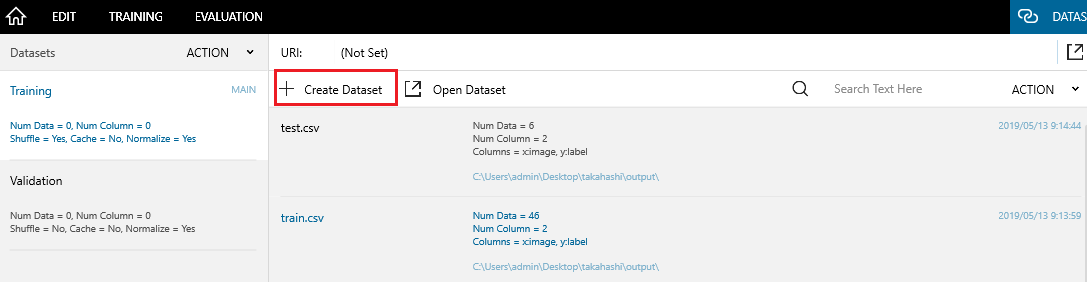
データの用意が終わったら実際に前処理を行っていきます。具体的にはサイズがバラバラなのでリサイズすることや、訓練用のデータとテスト（評価）用のデータに分けることなどが前処理になります。NNCにおいてデータセットはcsvファイルで記述されます。



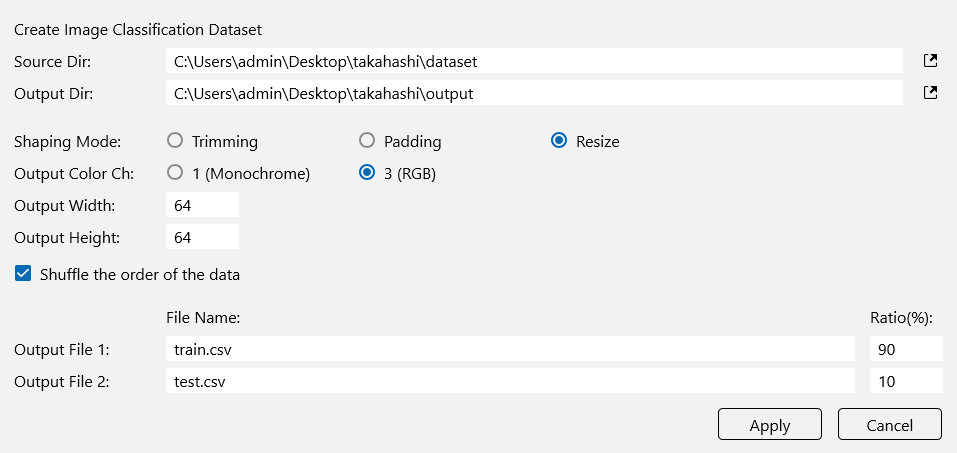
まずプロジェクト画面からDATASETをクリックします。



このような画面に移動します。左の緑の枠にTrainingとValidationとあります。Trainingに訓練用のデータセットを、Validationにテスト（評価）用のデータセットを与えます。そのためのデータセットを作成するために右上（赤枠）をクリックします。



すると既に作成されているデータセットの一覧が表示されます。今回は既存のものは使わないので、Create Datasetをクリックします。



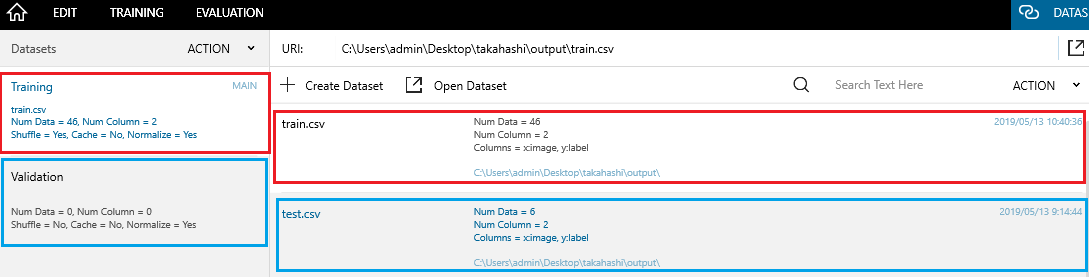
ウィンドウが表示されるので、必要事項を埋めていきます。

|  |  |
| --- | --- |
| Source Dir | 画像ごとに分類されたフォルダが設置されたフォルダを指定します。今回はdataset/下にappleとearthを置いているのでdataset/を指定します。 |
| Output Dir | 作成したデータセットを出力するフォルダを指定します。  今回はoutput/を指定します。 |
| Shaping Mode | 各フォルダに含まれる画像のアスペクト比が指定サイズの示すアスペクト比と異なる場合、アスペクト比を合わせる方法を指定します。  Trimming：トリミングでアスペクト比を合わせます  Padding：画像の端に0を挿入することでアスペクト比を合わせます  Resize：拡大縮小をすることでアスペクト比を合わせます  今回は64x64を指定サイズとして、Resize によってアスペクト比を合わせます。 |
| Output Color Ch | 出力画像のカラーチャンネル数を1（モノクロ）もしくは3（RGBカラー）で指定します。  今回はRGBカラーで行います。 |
| Output Width | 出力画像の横幅を指定します。  今回は64です。 |
| Output Height | 出力画像の縦幅を指定します。  今回は64です。 |
| Shuffle the order of the data | データの順序をシャッフルします（デフォルトでON） |
| File Name | 作成するデータセットCSVのファイル名を指定します |
| Ratio (%) | 作成するデータセットCSVに用いるデータの割合を指定します。File 1、File 2の割合の合計は100 (%)である必要があります。  今回は90%を学習に利用し、テストに10%利用します。 |

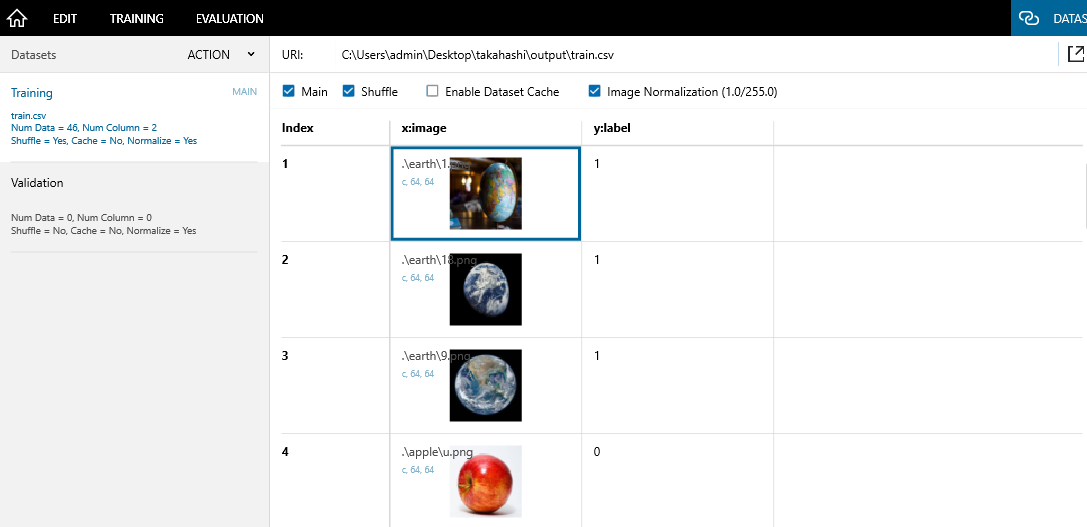
入力が完了したらApplyで前処理開始です。

#### データセットの設定

前処理によって作成されたデータセットを具体的にTrainingとValidationに割り当てます。



DATASETの画面へ戻ると新しくtrain.csvとtest.csvが追加されています。Trainingにはtrain.csv,Validationにはtest.csvを割り当てることで準備完了です。

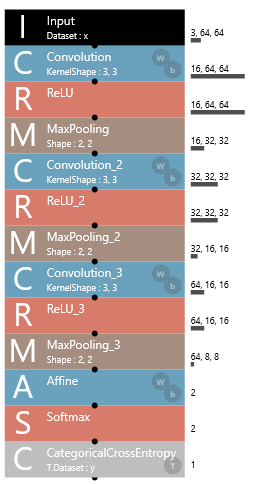


trainingにtrain.csvを割り当てるとこのような画面になります。このy:Labelというのが分類の指標となります。画像ではリンゴが0、地球が1となっています。すなわちプログラムは画像がリンゴであると推論すれば0、地球であると推論すれば1を出力するということになります。

### ニューラルネットワークの構築

前処理が終了したのでニューラルネットワークを構築していきます。

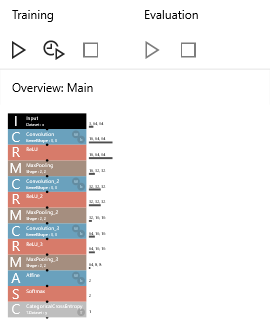
しかしこのニューラルネットワークの構築、機械学習についてよく理解していなければ難しいのでまずは他の人のものを参考にします。具体的には<http://arkouji.cocolog-nifty.com/blog/2019/02/sony-neural-net.html>　にて述べられているニューラルネットワークを構築します。



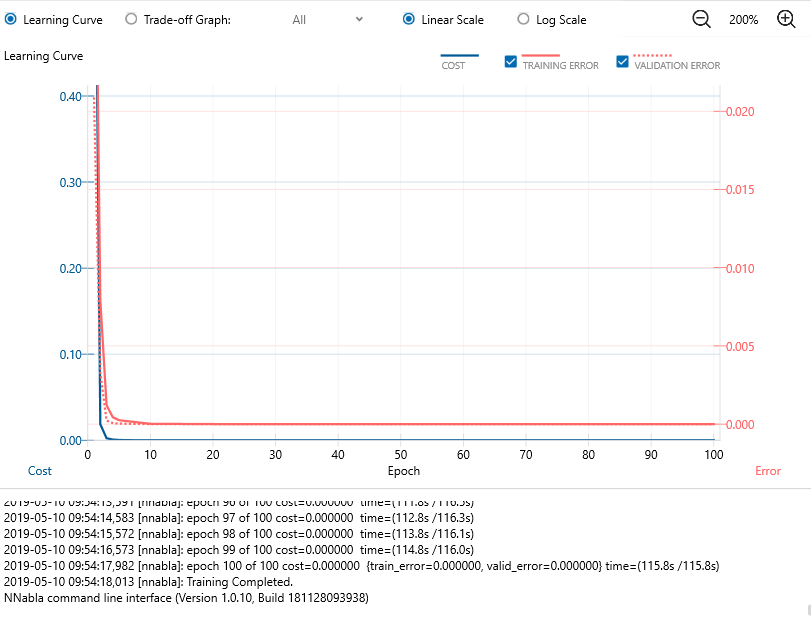
このようなネットワークが構築できました。

これを利用して次に学習を行っていきます。

### 学習



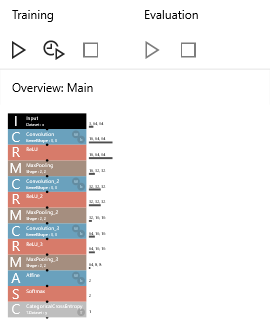
画面右側にあるTrainingの▷ボタンを押すと学習が開始されます。



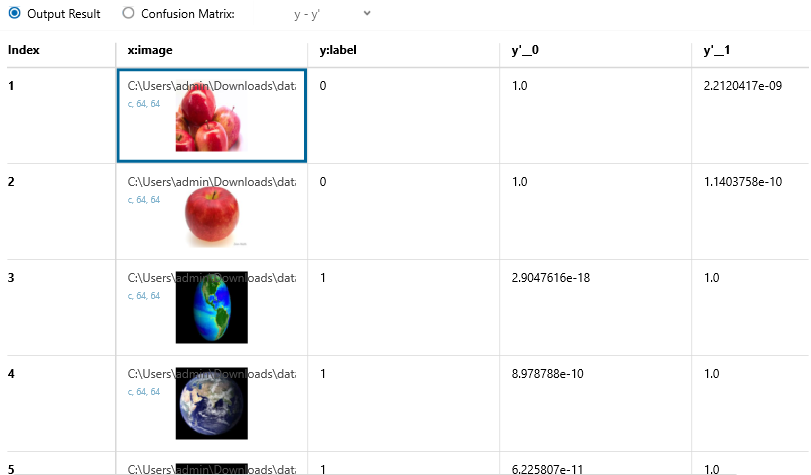
学習が進んでいくと曲線がこのように変化していきます。学習が進むにつれて間違い(VALIDATION ERROR)が少なくなっていき、正しく学習している様子を確認できます。

次に、この学習がうまくいっているかを確認するために評価を行っていきます。

### 評価



画面右側にあるEvaluationの▷ボタンを押すと評価が開始されます。



評価が終了すると以上のような画面に切り替わります。

ここでy:Labelが正しいラベルです。y’\_0,y’\_1の値はそれぞれ、画像がラベル0である確率,画像がラベル1である確率を表します。

Index1のリンゴの推論を見てみると、y’\_1つまりラベルが1である確率がほぼ0に近い値まで低いことがわかります。したがってこの画像はリンゴに分類できていると考えられます。

以下同様に、今回の学習では100%正しく判別が行えました。（画像の総数が少ないことや、リンゴと地球の相似点があまりないことが理由かもしれません。）

# より難しい認識を行う

前回までは「地球」と「リンゴ」の認識を行ってきました。これらは人間が見ても明らかに異なる対象なので、正答率は非常に高い値となりました。次に、人間が見ても少しわかりにくい対象について認識を行ってみたいと思います。

課題設定としては「さんま」と「いわし」とします。



図1:さんま 図2:いわし

特徴としてはサンマは全体的な形状、いわしは腹の点のような模様です。

### 実際に学習する

##### 1.プロジェクトの作成

新しく学習をはじめるので、前回の設定と混ざらないようにプロジェクトを作成しておきます。

##### 2.前処理

まず前処理のために画像を集めます。集めたデータは前回と同様に適当にフォルダに入れます。今回は

**sakana\_output/**

**sakana\_data/**

**iwashi/**

**sanma/**

のようにしました。

次にNNCで前処理をします。

Source Dirにsakana\_data/、Output Dirにsakana\_output/を選択します。

その他の設定は前回と同様です。

##### 3.ニューラルネットワークの構築

前回のニューラルネットワークを利用します。

前回のプロジェクトを開くとニューラルネットワークが表示されます。ctrl+A,ctrl+Cですべてコピーして、新しいプロジェクトに張り付けるだけで完了です。

##### 4.学習&評価

Trainingの▷ボタンから学習を行います。

結果は以下のようになりました。

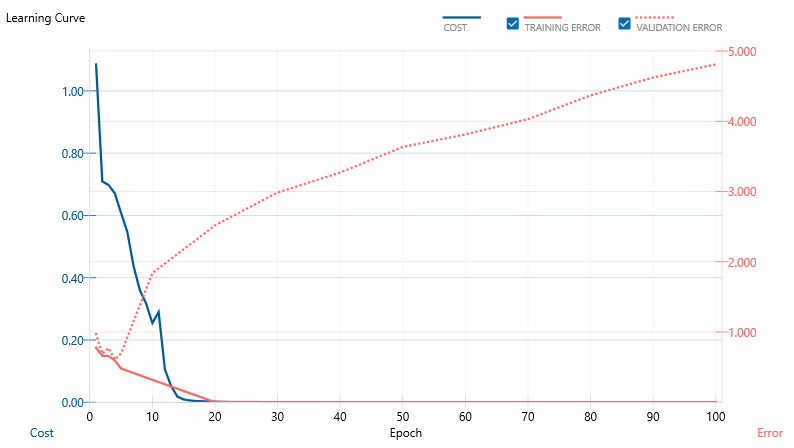


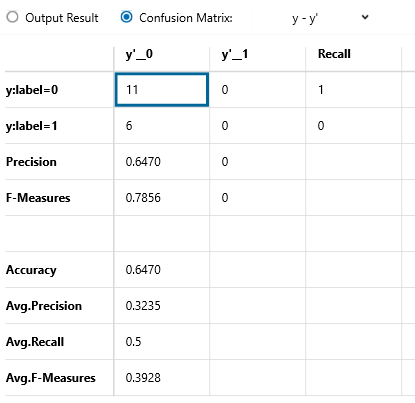
図3 sanma-iwashi Learning Curve

図3を見ると、VALIDATION ERRORである赤い点線が右肩上がりなのが読み取れます。これはおおよその意味では「うまく分類できなかった数」を意味します。X軸がおおよそ「学習時間」を意味するため、つまり「学習が進んでもうまく分類ができずにエラーが増えてしまった」という意味を持っています。

本来であれば学習が進むにつれてVALIDATION ERRORは減っていくため、うまく学習できていないことになります。つまり、さきほどのニューラルネットワークではうまくいかないようです。

この学習データでEvaluation（評価）を行ってみます。

するとConfusion Matrixは以下のようになります。



この表は

y:label=0は正解ラベルが0であり、y’\_0 y’\_1とはそれぞれプログラムが推論した結果になります。つまり1行目は正解ラベル0であるものを0と推論した画像が11個あり、1と推論した画像が0個であるという意味になります。これは誤りがないので非常にいい結果です。

しかしy:label=1つまり正解ラベルが1である画像に対して0であると推論した画像の数が6枚で1であると推論した画像が0枚という結果を表しています。

これはy:label=1であるものをまったく判別できないという最悪の結果となっています。

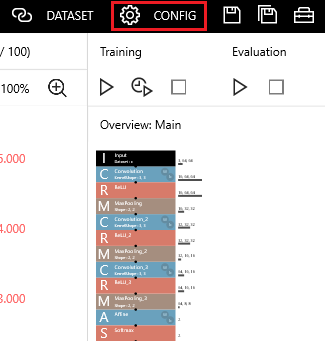
そのためAccuracy(正答率)は66.4%となっています。

### 最適なニューラルネットワークを探す

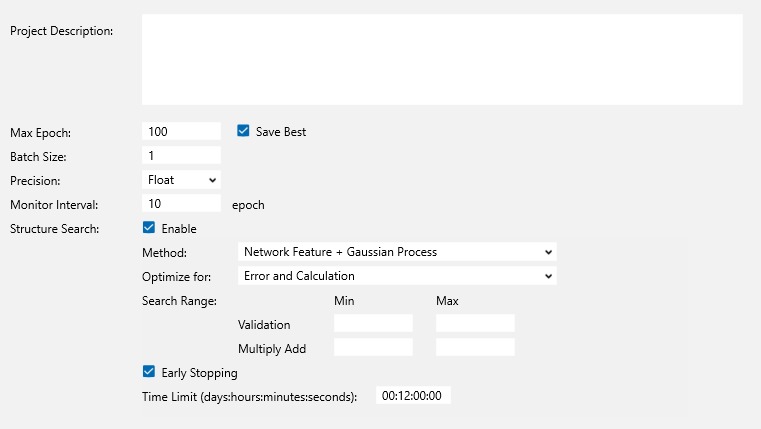
では新しいニューラルネットワークを構築する必要があります。しかしそうした設計には機械学習の専門的な知識が不可欠です。

ここではNNCの機能である「自動的に最適なニューラルネットワークを探す」というものを使ってVALIDATION ERRORを下げることを目標にしてみます。

##### 学習開始前の設定



プロジェクト画面右上のCONFIGから設定画面へ移ります。



画面中央にこのような画面が開きます。

これらの項目の詳細はNNCのリファレンス

<https://support.dl.sony.com/docs-ja/%E3%83%97%E3%83%AD%E3%82%B8%E3%82%A7%E3%82%AF%E3%83%88%E7%94%BB%E9%9D%A2%EF%BC%9A%E3%82%B3%E3%83%B3%E3%83%95%E3%82%A3%E3%82%B0%E3%82%BF%E3%83%96%EF%BC%88global-config%EF%BC%89%E3%81%AE%E4%BD%BF-2/>

に書かれています。

この画面のうちStructure Searchが「最適なニューラルネットワークを探す」という機能です。これは構築した「ニューラルネットワークを自動的に変化させる」→「学習」というプロセスを自動的に何度も繰り返して、VALIDATION ERRORの低いニューラルネットワークを見つける機能です。

設定は以下の通りです。

Structure SearchのEnableにチェック

Method : Network Feature + Gaussian Process

Optimize for : Error and Calculation

Search Range : 指定なし

Early Stoppingにチェック

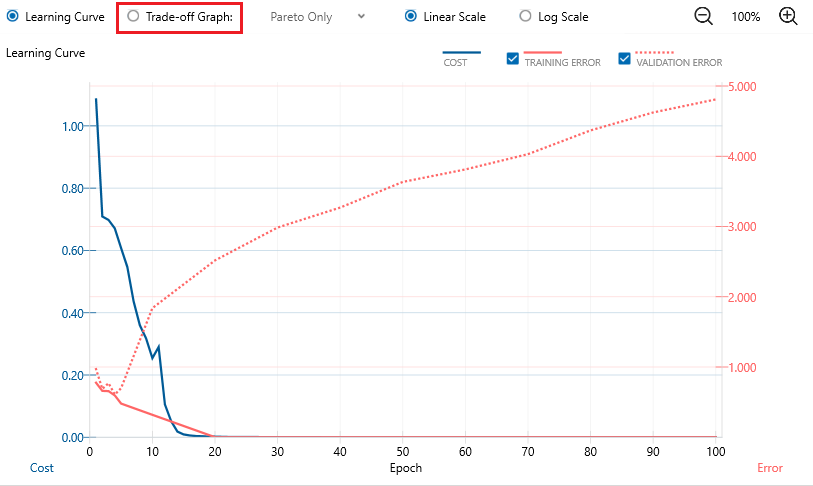
Time Limit : 00:12:00:00

とします。

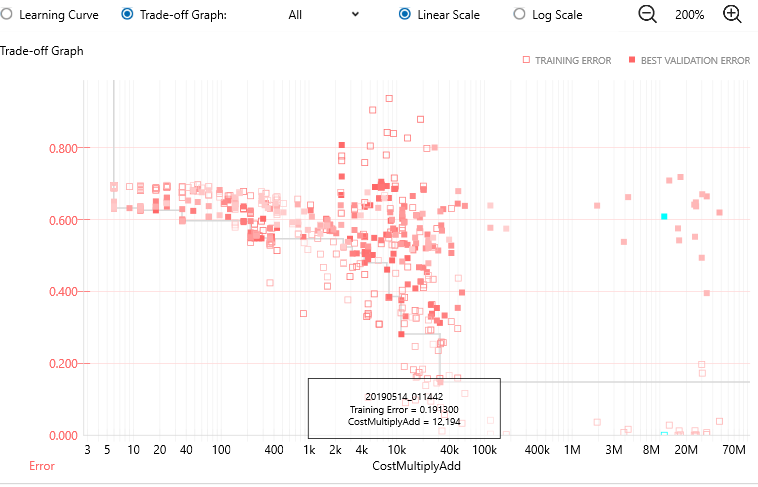
Time Limitでこの探索をやめる時間を設定できるので、今回は12時間学習させてみました。

##### 学習の結果を確認する

12時間経った結果を確認していきます。しかしあまりにも学習結果が多いので、Trade-Off Graph というものを使って何が良いのかを判断します。

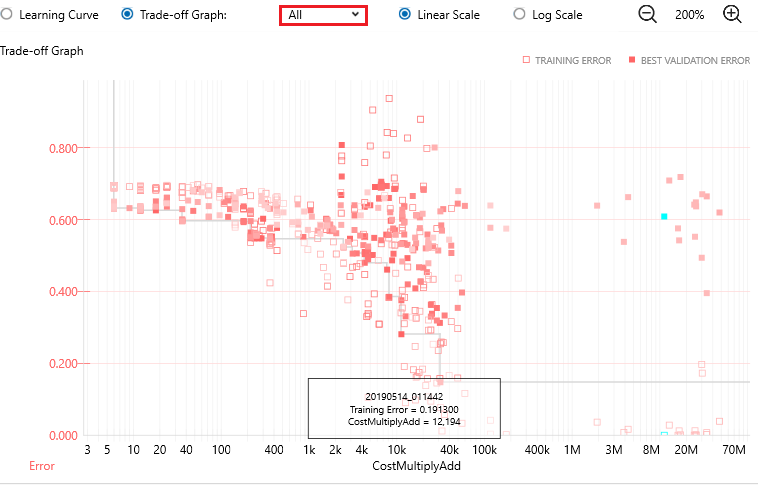


学習結果をひとつ開いて、上にあるTrade-off Graphを選択します。

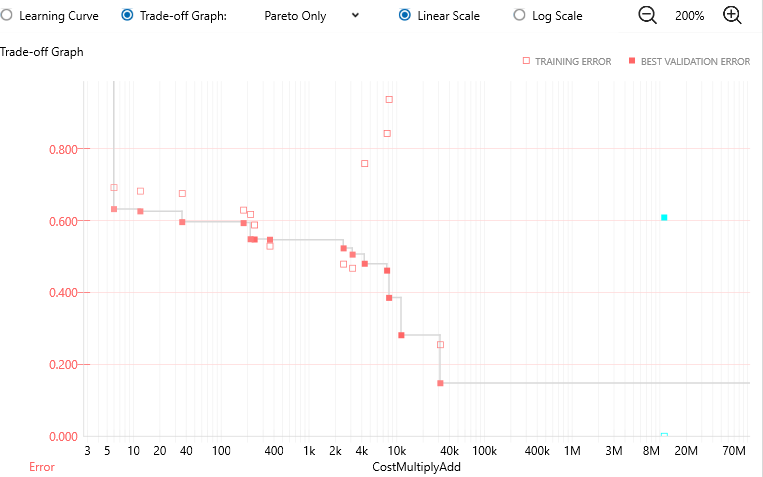


このようなグラフが表示されます。このプロットひとつが学習データひとつに対応しています。いま縦軸がErrorなので、なるべく下のほうを見たほうが良さそうだということがわかります。

また横軸がおおよその意味として学習コストを意味するため、選ぶべきなのは「縦軸の値が小さい」かつ「横軸の値が小さい」という学習データです。

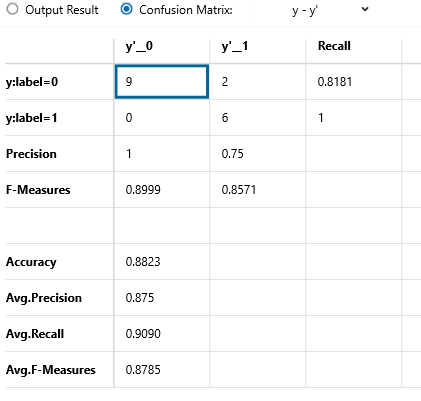
このようなデータをより分かりやすくプロットする方法があります。

上の画面で、ALLとなっている箇所から「Partero Only」選択します。



するとこのように情報が絞られます。いまほしいのはErrorが最も低いものなので40k付近にある最小のデータ点をクリックします。

すると学習データが開くので、この学習データでEvaluationを行ってみます。



結果としてはAccuracyが88%となりました。最初の66%よりだいぶ向上しました。

# Pythonで学習結果を扱う

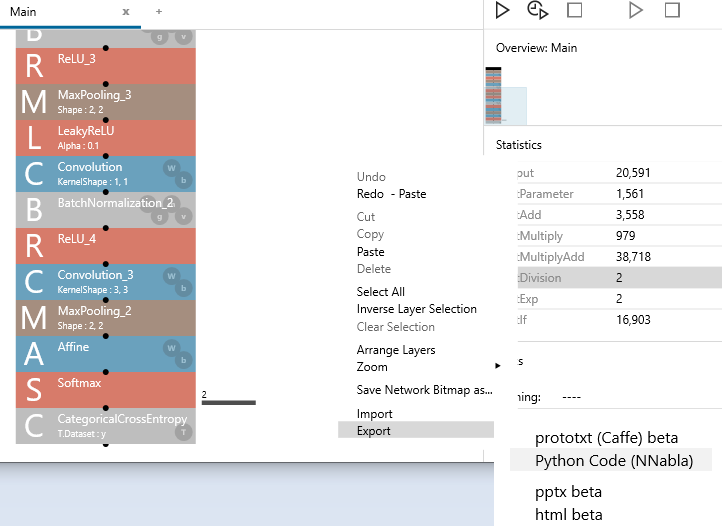
これまではNNC上で機械学習を行ってきました。今回はこれらの結果を実際のプログラムに利用する方法をまとめます。ここでは「リンゴ」か「それ以外」の認識データを作成したとします。実際に扱うデータは以下からダウンロードできます。

<https://github.com/tkhsry/NNC_WebCameraJudge>

##### 1.ニューラルネットワークのPythonコードをコピーする

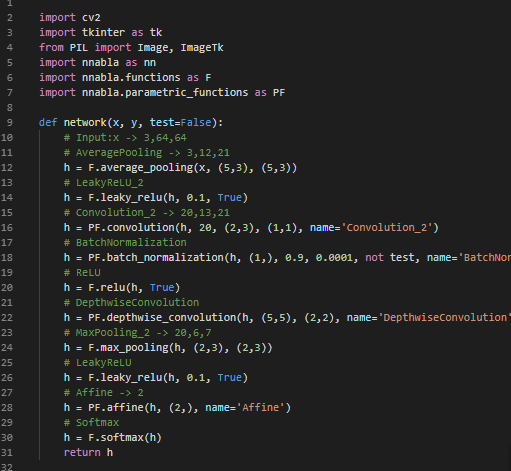
まずプロジェクト画面から利用したい学習済みのデータを開きます。

するとEDITに作成したニューラルネットワークの図が表示されます。ここで左クリックをするとメニューが開きます。



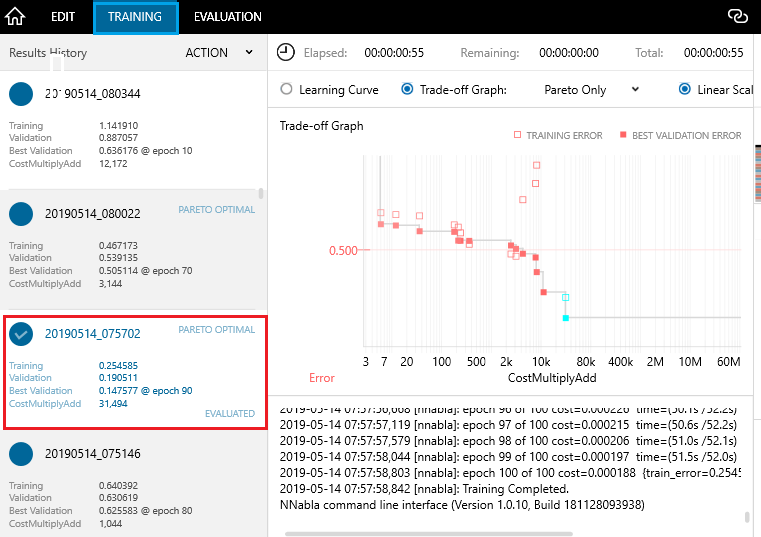
[メニュー]→[Export]→[Python Code(NNabla)]を選択すると、Pythonのコードがクリップボードにコピーされます。

これを.pyファイルに張り付けます。



##### 2.学習済みデータをコピーする

次にNNCにて現在開いている学習データを上のTRAININGから開いて、ダブルクリックします。するとその学習データが保存されているフォルダが開きます。



(赤枠をダブルクリック)



そこにresults.nnpがあるのでPythonスクリプトを開いているディレクトリにコピーします。これが学習済みデータになります。

##### 

##### 3.ライブラリのインストール

必要となるライブラリはNNablaというものです。

pip install nnabla

で完了です。

NNablaについては公式チュートリアル： <https://nnabla.readthedocs.io/en/latest/python/tutorial.html> 　を参照。

以上で学習済みデータをPythonで読み込むことができます。