Deep Learning I Python med SAS DLPy: Hva kan vi bruke det til og hvorfor?

Ina Conrado

Analytics & AI Advisor

Nordics Analytics Practice

What is Deep Learning and SAS DLPy?

What is Deep Learning?

Deep learning is a subset of **machine learning** that trains a computer to **perform human-like tasks**, such as speech recognition, image identification and prediction making. It **improves the ability** to classify, recognize, detect and describe using data.

Building Models: Open Source to SAS

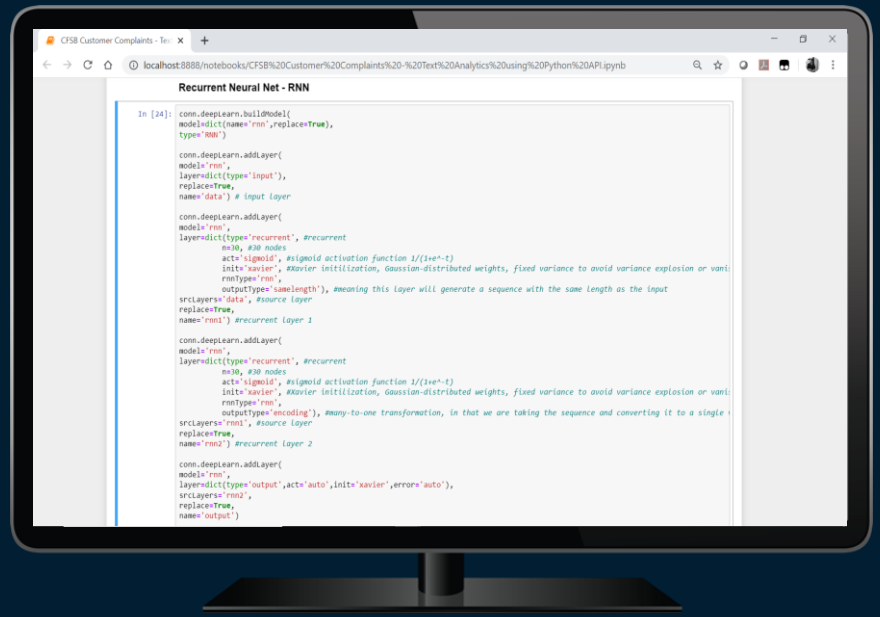
DLPy- Deep Learning for Python

SAS DPLy is an API wrapper around the SWAT package but with access to SAS deep learning functions, serving as a bridge from native open source deep learning models to SAS.

Prerequisites: SWAT, python 3+, SAS Viya 3.4+, SAS VDMML 8.3+.

What can you do with DLPy?

- Leverage pre-built deep learning networks (LeNet, VGG, ResNet, DenseNet, Yolo,etc.) for image, text, time series data, with similar deep learning syntax
- Built-in model analysis tools (heat maps, feature maps, etc.)
- Model creation with tasks (image classification, sequence labelling, speech recognition, etc.)



Why use DLPy?

Building Models: Open Source to SAS

For this option, open source code is written in an IDE and then submitted to SAS Viya's in-memory distributed computing engine, CAS* through APIs*.

Benefits for this option include:

Ability to program in language and text editor of choice **for added user flexibility.**

Programmers can transfer the heavy lifting part of their jobs onto a CAS server **for quicker processing.**

Creates a single analytics data store **for centralized governance.**

Building Models: Open Source to SAS

How Does SAS Communicate with Open Source Languages?

To leverage SAS Viya's CAS server, you would connect to the CAS server from your code editor.

Programming interface for editing code

Program in an open source language from a local or remote editor of your choice.

SAS Viya's engine, CAS*

CAS is an in-memory distributed computing engine that lets you process huge amounts of data quickly.

How does it work?

Building Models: Open Source to SAS

DLPy- Deep Learning for Python

How can you use DLPy?

- Import SWAT, DLPy package & functions
- Connect with a CAS server
- Load SWAT Action Sets *(if needed)*

- Load data into CAS
- Possible exploration steps:
  - Summary statistics
  - Check frequency

- Prepare data (if needed):
  - Image resizing
  - Shuffle
  - …
- Build DL models
  - CNN
  - LSTM
  - …

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 1: Create Connection

Install SWAT & DLPy packages

*To use DLPy, you will need to import both SWAT and DLPy as **DLPy does not do any analytic operations itself**. It is a wrapper around the SWAT package which communicates with a SAS CAS server to do the analytics.*

```
import swat
```
→ To talk to SAS CAS

```
import DLPy
```
→ To use SAS Deep Learning *

```
import matplotlib.pyplot as plt
%matplotlib inline
```
→ To use Python native visualization package

** If DLPy is not installed, you can install it via:*

```
pip install sas-DLPy
```
OR
```
conda install -c sas-institute sas-DLPy (for Anaconda)
```

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 1: Create Connection

Import specific deep learning functions

*To learn about DLPy packages, you can use **help** method or **dir** method for different level of the package. More information can be found on GitHub or this API page.*

`help(DLPy.model)`

```
NAME
    dlpy.model - The Model class adds training, evaluation, tuning and featu
re analysis routines to a Network

CLASSES
    builtins.object
        FeatureMaps
    dlpy.network.Network(dlpy.layers.Layer)
        Model
    dlpy.utils.DLPyDict(collections.abc.MutableMapping)
        DataSpec
        DataSpecNumNomOpts
```

`dir(DLPy.model)`

```
['AdamSolver',
 'Box',
 'DLPyDict',
 'DLPyError',
 'DataSpec',
 'DataSpecNumNomOpts',
 'FCMPLR',
 'FeatureMaps',
 'FixedLR',
```

Example:

```
from DLPy.splitting import two_way_split
from DLPy.applications import *
from DLPy.image_embedding import *
```

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 1: Create Connection

Connect to CAS server via SWAT*

To enable the API communication with CAS, you will first need to establish a connection with a SAS CAS server using the SWAT package.

```
conn=swat.CAS(cashost, casport, userid, password)
```

Argument	Description	Example
cashost	Hostname for the CAS Server	'sas.server.com'
casport	Port to be used for the connection	5570 (by default) ***
user	User ID for the connection	'DOMAIN\firstname.lastname'
password	Password for the User ID specified	'password'

To check if the connection is working, you can run a ***serverstatus()*** *method*

```
conn.serverstatus() **
```

**** 5570 for binary connection (Linux-Linux); 8777 for http connection (Win-Linux)*

** Similar to the SWAT section*

** conn=swat.CAS

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 1: Create Connection

Load CAS deep learning Action sets

The SAS CAS deep learning Actions currently consist of three Action Sets:

deepDnn

Use the **deepDnn** Action Set to build fully-connected neural networks.

deepRnn

Use the **deepRnn** Action Set to build RNN models that use raw text inputs.

deepLearn

The **deepLearn** Action Set is a superset of deepDnn and deepRnn and can be used to build DNN, CNN, and RNN models.

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 2: Load Data in CAS and Explore

Load data into CAS

*If your data is not already loaded into the CAS server, you will need to load the data. Data upload can be done either through SWAT or through DLPy, depending on the data type *.*

Using DLPy:

```
train_image=DLPy.ImageTable.load_files(conn,path='/data/images')
```

Using SWAT:

```
train_image=conn.table.loadTable(caslib='CASUSER',
        path='/data/images/img.sashdat',casout=dict(name='data',
        replace=True))
```

** You can only use SWAT to upload files from client to CAS with supported data types (Excel, SQL, CSV, SAS, Stata, etc.). See here for more details. With image data or audio data, you will need to use DLPy.*

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 2: Load Data in CAS and Explore

Explore the data - examples

- **label_freq** → To perform a label frequency analysis
- **image_summary** → To perform a statistical summary (min, max, median, etc.)
- **kurtosis** → To return the kurtosis of the values in each column

label_freq

	Level	Frequency
Ant-Man	1	241
Aquaman	2	201
Avengers	3	216
Batman	4	779
Black Panther	5	460

image_summary

Image Table Metadata Summary	
jpg	409
minWidth	170
maxWidth	1024
minHeight	127
maxHeight	1024
meanWidth	912.147
meanHeight	778.499

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 3: Prepare and Build Deep Learning Models*

Best practice #1: Knowing which pre-built models are available in DLPy **

Tasks	Example models	Example

Computer Vision***	**LeNet Models**
	VGG Models (VGG11, VGG13, VGG16, VGG19)
	ResNet Models (ResNet18, ResNet34, ResNet50, ResNet101, ResNet152)
	DenseNet Models
	DarknetModels
	Inception Models (v3)
	YoloV2 and **Tiny_Yolov1** Models
		`LeNet5(conn[, model_table, n_classes, ...])`
		
		`ResNet18_SAS(conn[, model_table, ...])`
		
		`InceptionV3(conn[, model_table, n_classes, …])`
		
		`Yolov2(conn, anchors[, model_table, …])`

Natural Language Processing	**Text Classification** **Text Generation** **Sequence Labelling** **Speech Recognition**	`TextClassification(conn[, model_table, …])` `SpeechRecognition(conn[, model_table, …])`

** The use of different deep learning models is different depending on the source algorithms so we will focus on best practices in this section*

*** You can also build custom models by creating and defining layer by layer of a NNet*

**** See full list of computer vision models in this link*

Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 3: Prepare and Build Deep Learning Models

Best practice #2: Make sure to understand your model architecture well

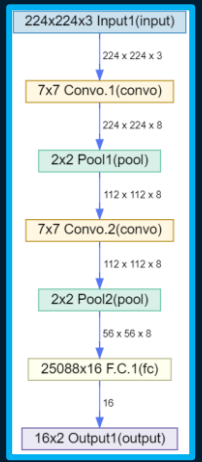
It's important to understand a deep learning model architecture well prior to training to validate if there are any modifications should be made (for example, auto encoder will require a symmetrical architecture).

DLPy provides architecture information tasks to help summarizing the model architecture.

Example:

```
model.print_summary()
model.plot_network()
```

	Layer Id	Layer	Type	Kernel Size	Stride	Activation	Output Size	Nu Pa
0	0	data	input			None	(224, 224, 3)	(0
1	1	Conv1	convo	(3, 3)	(2, 2)	Identity	(112, 112, 32)	(8
2	2	bn_Conv1	batchnorm			Relu	(112, 112, 32)	(0



Building Deep Learning Models: Open Source to SAS

Using DLPy

Step 3: Prepare and Build Deep Learning Models

Best practice #3: Make sure to check Loss function over the training process

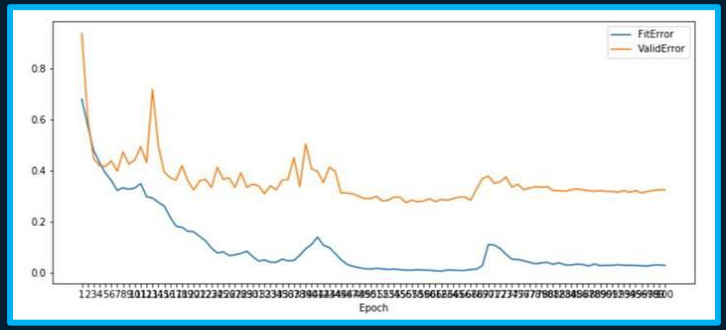
It's always helpful to understand the training history across epochs (1 epoch = one time an entire data set is passed both forward and backward through a neural network) to monitor and optimize the training process.

Example:

```
model.training_history
model.plot_training_history()
```

Model Training History Table

Epoch	Learning Rate	Loss	FitError
1	0.0001	1.166900	0.138720
2	0.0001	0.145040	0.070122
3	0.0001	0.130477	0.066311
4	0.0001	0.106502	0.054878
5	0.0001	0.096979	0.051829



Building Deep Learning Models: Open Source to SAS

Using DLPy

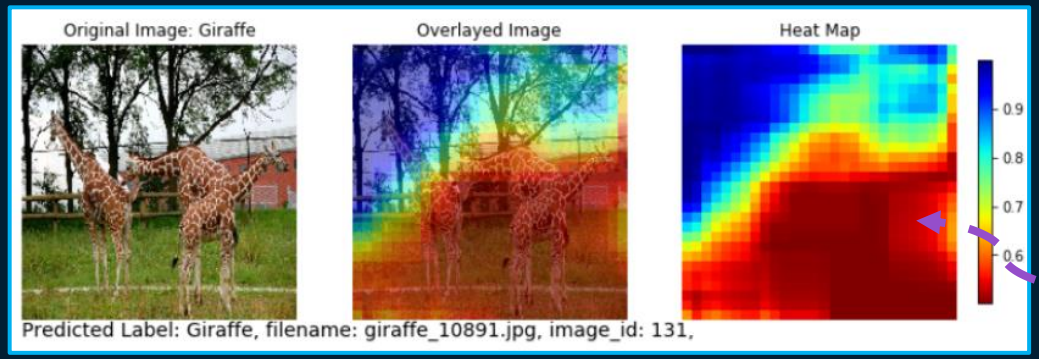
Step 3: Prepare and Build Deep Learning Models

Best practice #4: To be prudent, check the quality of your trained model

Heat maps are a good way to use color visualization to identify if a trained model has captured the most useful information from the original data. It's useful to tell which parts of the data carry the most information about the whole data (like variable importance in machine learning).

Example:

```
model.heat_map_analysis ()
```



Example of Making Use of Pre-trained Models

- A transformer is a deep learning model that **adopts the mechanism of attention, differentially weighing the significance of each part of the input data**.

- **Bidirectional Encoder Representations from Transformers** (**BERT**) is a transformer-based machine learning technique for natural language processing (NLP) pre-training. It was originally developed by Google, but the technique is utilized to develop for additional languages.
  - Many of these variations can be found in huggingface: https://huggingface.co/transformers/pretrained_models.html

- https://github.com/sassoftware/python-dlpy/blob/master/examples/transformers/bert_load_model_quickstart.ipynb

Further Reading

Github page:

https://github.com/sassoftware/python-dlpy

Examples:

https://github.com/sassoftware/python-dlpy/tree/master/examples

https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4429-2020.pdf

Blogs:

https://blogs.sas.com/content/subconsciousmusings/2019/03/13/sas-deep-learning-with-python-made-easy-using-dlpy/

Thank you!

sas.com