

How to Do Deep Learning With SAS®

An introduction to deep learning neural networks and a guide to building deep learning models using SAS®



Contents

Introduction.....	1
Deep Neural Networks.....	1
Neural Networks Supported by SAS	4
Convolutional Neural Networks	5
Recurrent Neural Networks	6
Feedforward Neural Network.....	7
Autoencoder	7
Applications of Deep Neural Networks.....	7
Computer Vision	7
AI Assistants and Bots	7
Time Series Analysis.....	8
Natural Language Processing (NLP).....	8
Rare-Event Modeling	8
Build a Deep Learning Model Using SAS®.....	9
SAS® Platform Architecture for Training and Scoring	
Deep Learning Models.....	11
Summary.....	12
Learn More	12
Endnotes.....	12

Introduction

This paper provides an introduction to deep learning, its applications and how SAS supports the creation of deep learning models. It is geared toward a data scientist and includes a step-by-step overview of how to build a deep learning model using deep learning methods developed by SAS. You'll then be ready to experiment with these methods in SAS® Visual Data Mining and Machine Learning. See page 12 for more information on how to access a free software trial.

Deep learning is a type of machine learning that trains a computer to perform human-like tasks, such as recognizing speech, identifying images or making predictions. Instead of organizing data to run through predefined equations, deep learning sets up basic parameters about the data and trains the computer to learn on its own by recognizing patterns using many layers of processing. Deep learning is used strategically in many industries (see Figure 1).

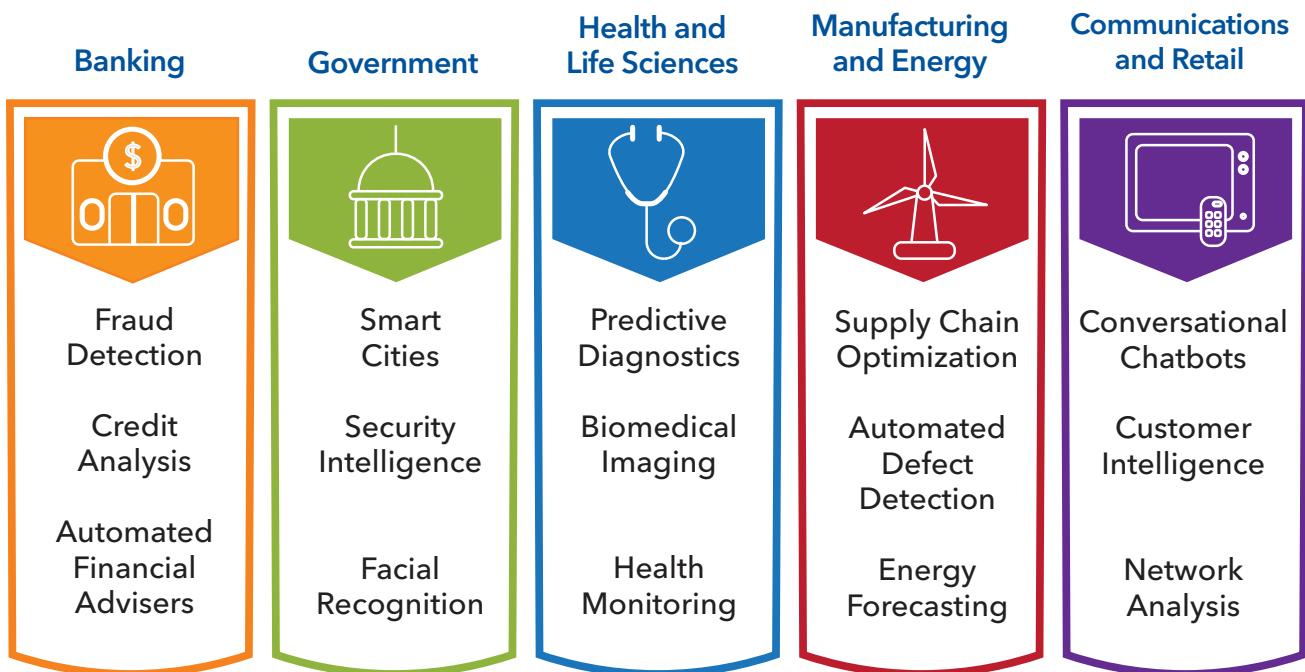


Figure 1: A few examples of how deep learning is used across a wide variety of industries.

Deep Neural Networks

Deep learning methods use neural network architectures to process data, which is why they are often referred to as deep neural networks.

Neural networks are represented as a series of interconnected nodes. A node is patterned after a neuron in the human brain. Similar in behavior to neurons, nodes are activated when there is sufficient stimuli (input). This activation spreads throughout the network, creating a response to the stimuli (output). Figure 2 shows an example of a simple neural network with its three key components: input layer, hidden layers and output layer.

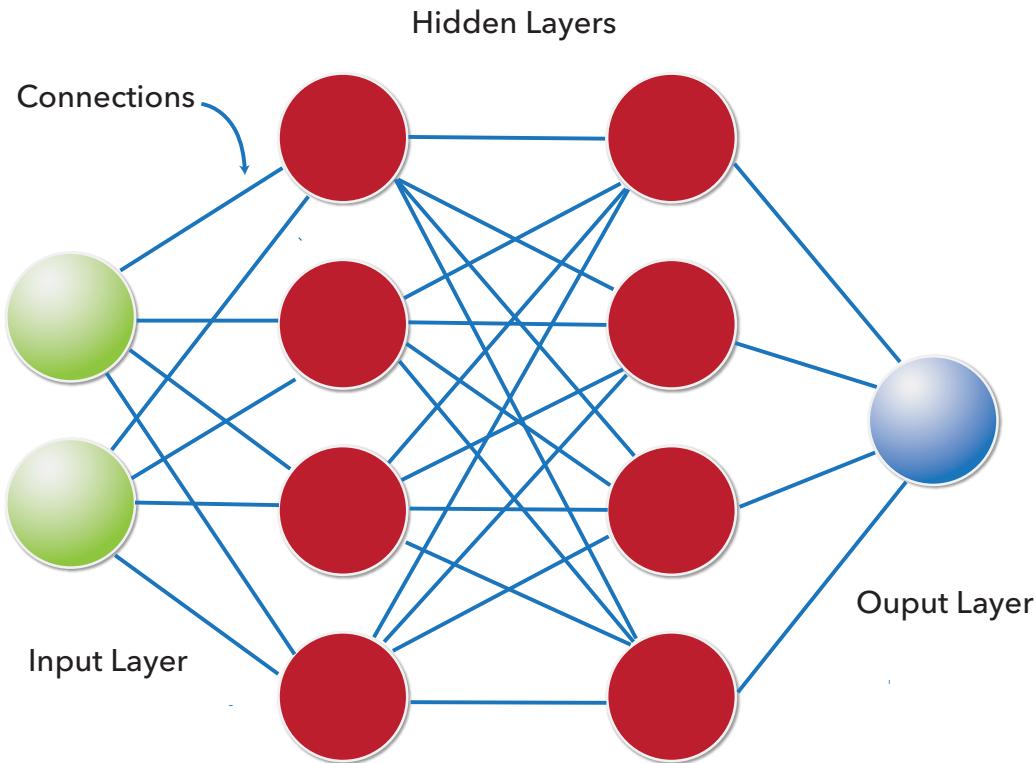


Figure 2: Organization of a simple neural network.

Here's how neural networks operate. First, data such as images, sequence data (like audio or text), etc. are fed into the network through the input layer, which communicates to one or more hidden layers. Processing takes place in the hidden layers through a system of weighted connections. Nodes in the hidden layer then combine data from the input layer with a set of coefficients (which either magnifies or diminishes the input) and assigns appropriate weights to inputs. These input-weight products are then summed up. The sum is passed through a node's activation function, which determines the extent that a signal must progress further through the network to affect the final output. Finally, the hidden layers link to the output layer - where the outputs are retrieved.

As the number of hidden layers within a neural network increases, deep neural networks are formed. (In this context, "deep" refers to the number of hidden layers in the network.) A traditional neural network might contain two or three hidden layers, while deep neural networks (DNN) can contain as many as 100 hidden layers.

Deep neural networks are typically represented by a directed acyclic graph (DAG) consisting of interconnected layers (see Figure 3).

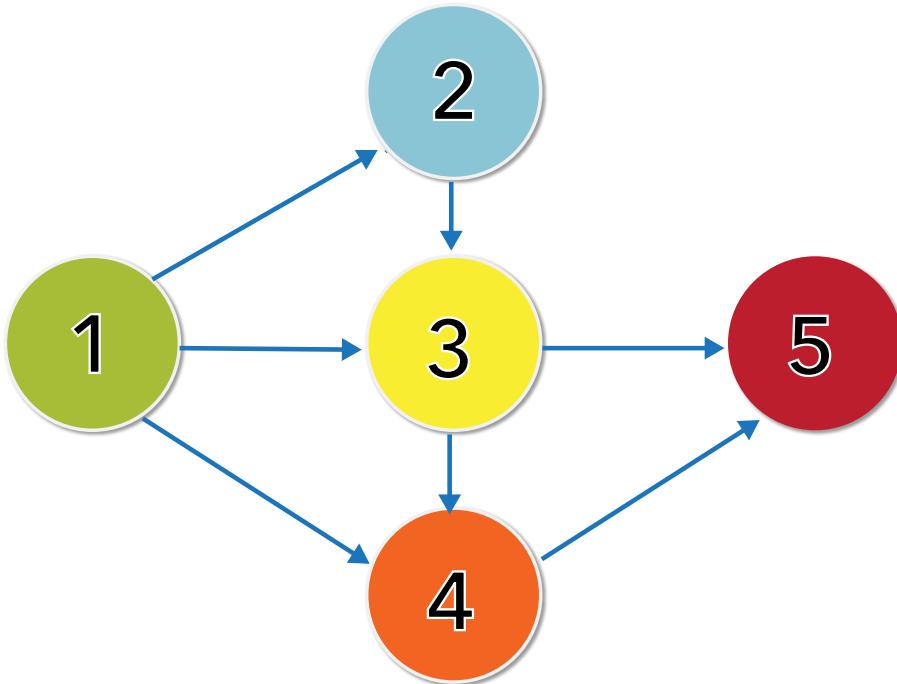


Figure 3: Example of a directed acyclic graph (DAG).

Deep learning networks minimize the need for explicit, time-consuming feature engineering techniques because of their built-in capacity to extrapolate new features from the set of features in the training set. They scale well to classification tasks that often require complex computations and are widely used for difficult problems that require real-time analysis, such as speech and object recognition, language translation and fraud detection. Finally, deep learning networks can also be used for multitask learning where models are trained to predict multiple targets simultaneously.

However, deep learning networks do have limitations. Models built from deep neural networks are not interpretable. Though it is mathematically possible to identify which nodes of a deep neural network were activated, it is hard to interpret what the neurons were supposed to model and what these layers of neurons were doing collectively to choose the final output. Because deep neural networks require substantial computational power, they can be difficult to deploy, especially in real time. Due to the many network layers, a huge number of parameters are needed to build the model. This can lead to model overfitting, which negatively affects how well the model generalizes. Last, deep learning is data hungry, typically requiring very large data sets.

Neural Networks Supported by SAS

SAS supports different types of deep neural network layers and models. Layers allow users to experiment and build their own deep learning architectures. Some of the layers that are currently supported by SAS include:

- Batch normalized layers.
- Convolutional layers.
- Fully connected layers.
- Pooling layers.
- Projection layers.
- Residual layers.
- Recurrent layers.
- Input and output layers.

SAS is also focusing its research on developing additional layers to support Inception- and Xception-like deep learning models.

The hierarchy of deep neural networks supported by SAS is shown in Figure 4.

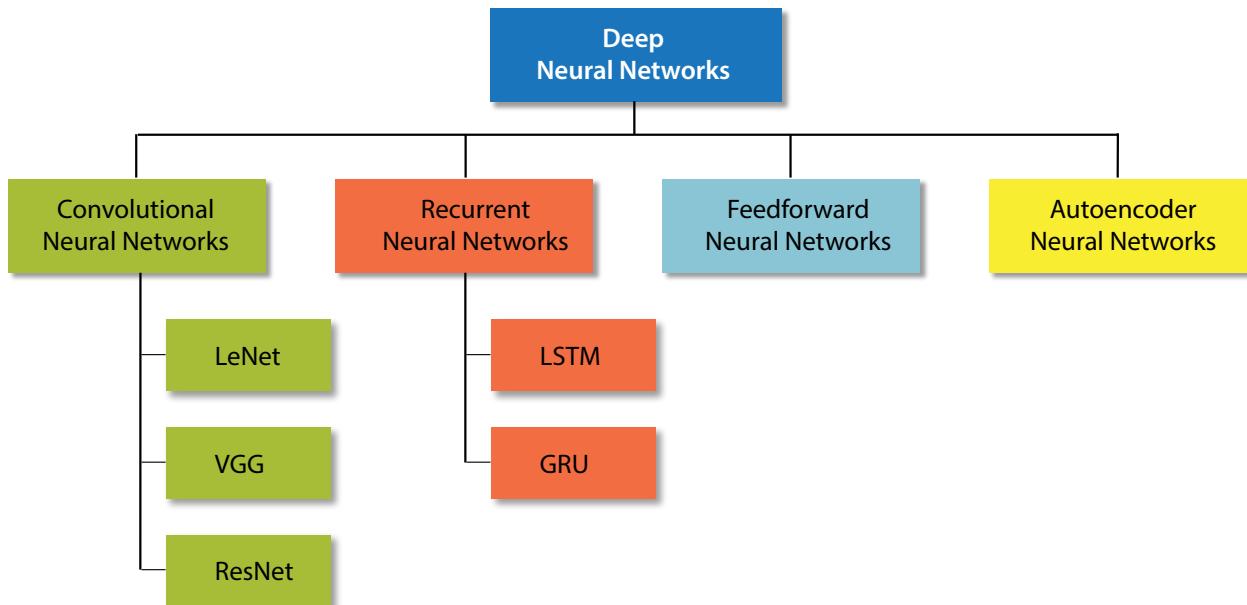


Figure 4: Deep neural networks supported by SAS.

Convolutional Neural Networks

Convolutional neural networks (CNNs) preserve the spatial structure of a problem. They are widely used in image analysis tasks. These networks use numerous identical replicas of the same neuron, enabling a network to learn a neuron once and use it in numerous places. This simplifies the model learning process and reduces errors (Waldran 2016).ⁱ

Unlike traditional neural networks, CNNs are composed of neurons that have shared weights and biases (i.e., all hidden neurons in the same layer share the same weights and biases). Hence, they use fewer parameters to learn and are designed to be invariant to object position and distortion in the given image.

The hidden layers in the network can be convolutional, pooling or fully connected:

- Convolutional. The neurons in this layer are responsible for extracting features from the input image by performing a convolution operation. This step preserves the spatial relationship between the image pixels by using only small inputs of data to learn the features.
- Pooling. The neurons in this layer help further reduce the dimensionality of the feature maps by performing downsampling. For example, max pooling takes the maximum value from a group of neurons in the previous layer and passes it as input to the next layer.
- Fully connected. All neurons in this layer are connected to every neuron from the previous layer. Using a softmax activation function produces output from this layer as a vector of probability values that corresponds to various target class labels. Each value for the class label suggests the probability that the given input image is classified as that class label.

LeNet

LeNets have a fundamental architecture with image features distributed across the entire image and convolutions that are used to extract similar features at multiple locations. They use a sequence of three layers: convolution to extract spatial features from an image, introduction of nonlinearity in the form of sigmoids and pooling using spatial average of maps to reduce dimensionality. A multilayer perceptron (MLP) is used as a final classifier.

VGG

Visual geometry group (VGG) networks are typically used for object recognition purposes. They are characterized by their simplicity, using only 3×3 convolutional layers stacked on top of each other. Reducing volume size is handled by max pooling. Two fully connected layers are then followed by a softmax classifier. Some of the model variants of VGG supported by SAS include VGG11, VGG13, VGG16 and VGG19.

Residual Neural Network (ResNet)

The depth of a neural network is commensurate to its performance in classification tasks. However, simply adding layers to a network often increases the training error and causes degradation problems where the accuracy degrades rapidly after saturating. ResNets overcome these difficulties by building deeper networks in such a way that:

- The layers fit the residual of the mapping instead of allowing the layers to fit an underlying desired mapping. This solves the degradation problem.
- Initial layers are copied from the shallow neural net counterparts and the added deeper layers are skip connections (or identity mapping) where the input is directly connected to the output. If the residual becomes small, the mapping becomes an identity mapping. This way, training error does not increase. (Dietz 2017).ⁱⁱ

Research by Ioffe and Szegedy shows that network training becomes particularly hard when the distribution of the input keeps changing whenever the weights in the previous layer change. The training time is increased by the need to use smaller learning rates and carefully initialize parameters.ⁱⁱⁱ ResNets use batch normalization to overcome this problem. Each layer's input is normalized for each mini-batch size that is defined. This process makes the network less susceptible to bad initialization and overfitting. It also accelerates the training process.

For these reasons, ResNets are considered state-of-the-art convolutional neural network models (Tamang 2017).^{iv}

Recurrent Neural Networks

Recurrent neural networks (RNNs) use sequential information such as sequence data from a sensor device (time series) or a spoken sentence (sequence of terms). Unlike traditional neural networks, all inputs to a recurrent neural network are not independent of each other since the output for each element depends on the computations of its preceding elements. Hence, connections between the nodes form a directed cycle, creating an internal memory within the networks. These networks are recurrent because they perform the same task for every element of a sequence. RNNs are used in forecasting and time series applications, sentiment analysis, text categorization and automatic speech recognition.

LSTM

LSTMs are long short-term memory models, capable of remembering dependencies for long periods of time. These models are RNN variants consisting of LSTM units. A typical LSTM unit comprises a cell, an input, an output and a forget gate. The forget gate is responsible for short-term memory in LSTMs. It controls how long a value residing in a cell must be remembered. This aspect of short-term memory is important because it makes the networks learn to forget undesired data and adjust accordingly to better fit the models. LSTMs are one of the most preferred models in deep learning because of their high accuracy measures. However, computation takes longer. The trade-off between performance and computation time should be considered when choosing the most pertinent model.

GRU

GRUs are gating mechanisms in RNNs where the flow of information is similar to LSTM networks but a memory unit is not used. They are considered computationally more efficient than LSTMs.

Feedforward Neural Network

These are simple neural networks where each perceptron in one layer is connected to every perceptron from the next layer. Information is constantly fed forward from one layer to the next in the forward direction only. There are no feedback connections in which outputs are fed back into themselves. Feedforward networks are mainly deployed in applications such as pattern classification, object recognition and medical diagnosis.

Autoencoder

Autoencoders are unsupervised neural network algorithms, primarily used for dimensionality reduction tasks. They transform the input into a lower dimensional space and then reconstruct the output back from this compact representation. In this way, the output obtained from the network is the same as the input given to the autoencoder. The layers of an autoencoder are stacked on top of each other and trained internally. The output labels are generated by the network themselves instead of learned from the labeled data (Dertat 2017).^v

Applications of Deep Neural Networks

Each type of deep neural network has advantages and disadvantages, depending upon the use. A few examples of deep learning applications are discussed below.

Computer Vision

Deep learning plays a major role in the field of computer vision. The ability to interpret raw photos and videos has been applied to problems in retail, medical imaging and robotics, to name a few. CNNs are used in applications such as facial recognition, image question answering systems, scene labeling and some image segmentation tasks. With respect to image classification, CNNs achieve a better classification accuracy on large-scale data sets because of their joint feature and classifier learning capabilities.

AI Assistants and Bots

Computer programs capable of human-level speech and comprehension have evolved rapidly with the help of deep learning. Chatbots Magazine author Henk Pelk points out that chatbots developed from traditional machine learning algorithms use heuristics based on pattern recognition and rule-based expression matching. They provide good results whenever they are able to find and match the problem against a predefined set of rules. However, issues arise when the chatbots are not able to match the problem against a rule set.^{vi}

This is where deep learning models come in handy. In contrast to rule-based methods, the bots are self-trained to respond to situations such as forecasting sales of new products or providing insights into customer churn. With a well-trained model, chatbots not only can interpret natural language input but can also generate a response for any situation.

This has enabled chatbots to communicate via text, speak in a more natural voice, understand context using natural language interaction and run deep learning models to answer analytical questions. This is in contrast to bots that use rule-based methods.

Time Series Analysis

A time series is a sequence of data points recorded at specific, equally spaced intervals over time. For example, daily high temperatures form a time series. Or, the aggregation of transactional data (cash register receipts) to form a time series of weekly sales at a retail store.

We've already discussed how RNNs are good for sequence data. This makes time series data a good match for RNNs.

Time series data often adds to the modeling complexity due to autocorrelation - where successive observations are dependent on each other. For example, today's stock market closing price is related to yesterday's price. However, with the use of RNNs, it becomes possible to learn this correlation between the data. Since RNNs form a directed loop within the neural network, each node maintains its own internal memory and preserves the state of the network during subsequent iterations by using the previous state output as the input for the successive step. This property of RNNs make them a popular choice for modeling times series data (Pant 2017).^{vii}

Natural Language Processing (NLP)

Deep learning approaches provide very high performance across numerous NLP tasks. For instance, recurrent neural networks are used for sentiment analysis, speech-to-text conversion, natural language generation and document summarization. Convolutional neural networks have varied applications in text categorization, semantic relation extraction, spam detection and categorization of search queries.

Rare-Event Modeling

Deep, fully connected networks are predominantly used in cases of rare-event modeling such as defect analysis and fraud detection. Because it is hard to isolate or detect rare events, deep learning models can often isolate fraud through progressive learning. For example, unsupervised and supervised models can be used together to monitor transactions for fraud by first detecting an incident and then evaluating that incident as fraud or not fraud - all without hindering the fraud detection system.

Build a Deep Learning Model Using SAS®

SAS offers the flexibility to run deep learning models alongside other machine learning models in SAS Visual Data Mining and Machine Learning. This SAS solution supports clustering, different flavors of regression, random forests, gradient boosting models, support vector machines, sentiment analysis and more, in addition to deep learning. An interactive, visual pipeline environment presents each project (or goal) as a series of color-coded steps that occur in a logical sequence. The flexibility of including all models in a visual pipeline provides data scientists with the power to test different modeling approaches in a single run and compare results to quickly identify champion models.

SAS Visual Mining and Machine Learning takes advantage of SAS Cloud Analytic Services (CAS) to perform what are referred to as CAS actions. You use CAS actions to load data, transform data, compute statistics, perform analytics and create output. Each action is configured by specifying a set of input parameters. Running a CAS action processes the action's parameters and data, which creates an action result. CAS actions are grouped into CAS action sets.

Deep neural net models are trained and scored using the actions in the "deepLearn" CAS action set. This action set consists of several actions that support the end-to-end preprocessing, developing and deploying deep neural network models. This action set provides users with the flexibility to describe their own model DAGs to define the initial deep net structure. There are also actions that support adding and removing of layers from the network structure.

Appropriate model descriptions and parameters are needed to build deep learning models. We first need to define the network topology as a DAG and use this model description to train the parameters of the deep net models.

The steps involved in training deep neural network models, using the deepLearn CAS action set, are as follows:

1. Create an empty deep learning model.
 - The BuildModel() CAS action in the deepLearn action set creates an empty deep learning model in the form of a CASTable object.
 - Users can choose from DNN, RNN or CNN network types to build the respective initial network.
2. Add layers to the model.
 - This can be implemented using the addLayer() CAS action.
 - This CAS action provides the flexibility to add various types of layers, such as the input, convolutional, pooling, fully connected, residual or output as desired.
 - The specified layers are then added to the model table.
 - Each new layer has a unique identifier name associated with it.
 - This action also makes it possible to randomly crop/flip the input layer when images are given as inputs.

3. Remove layers from the model.

- Carried out using the removeLayer() CAS action.
- By specifying the necessary layer name, layers can be removed from the model table.

4. Perform hyperparameter autotuning.

- dlTune() helps tune the optimization parameters needed for training the model.
- Some of the tunable parameters include learning rate, dropout, mini batch size, gradient noise, etc.
- For tuning, we must specify the lower and the upper bound range of the parameters within where we think the optimized value would lie.
- An initial model weights table needs to be specified (in the form of a CASTable), which will initialize the model.
- An exhaustive searching through the specified weights table is then performed on the same data multiple times to determine the optimized parameter values.
- The resulting model weights with the best validation fit error is stored in a CAS table object.

5. Train the neural net model.

- The dlTrain() action trains the specified deep learning model for classification or regression tasks.
- By allowing the user to input the initial model table that was built, the best model weights table that was stored by performing hyper-parameter tuning and the predictor and response variables, we train the necessary neural net model.
- Trained models such as DNNs can be stored as an ASTORE binary object to be deployed in the SAS Event Stream Processing engine for real-time online scoring of data.

6. Score the model.

- The dlScore() action uses the trained model to score new data sets.
- The model is scored using the trained model information from the ASTORE binary object and predicting against the new data set.

7. Export the model.

- The dlExportModel() exports the trained neural net models to other formats.
- ASTORE is the current binary format supported by CAS.

8. Import the model weights table.

- dllImportModelWeights() imports the model weights information (that are initially specified as CAS table object) from external sources.
- The currently supported format is HDF5.

SAS® Platform Architecture for Training and Scoring Deep Learning Models

Deep learning models are highly computationally intensive. Because of this, you need a flexible and robust in-memory server for training and scoring these complex models.

Traditionally, CPUs are the processing choice for machine learning. However, GPUs are optimal for linear algebra calculations and have a long streak of performance advantages over CPUs on many parallel computations. With a good GPU, it is possible to iterate quickly over deep learning networks and run experiments much faster, reducing the latency of operationalizing the model.

The SAS Platform architecture (see Figure 5) uses massively parallel processing (MPP) and parallel symmetric multiple processors (SMP) with multiple threading for extremely fast processing. One or more GPU processors are provided with SMP servers. Real-time training and scoring is supported by SAS Event Stream Processing.

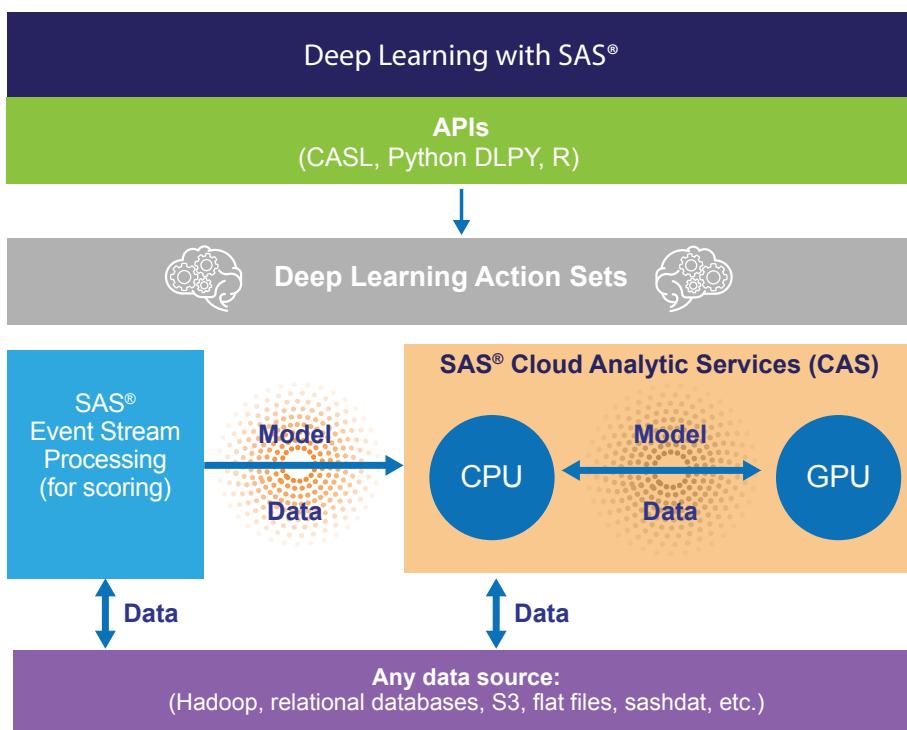


Figure 5: The SAS Platform architecture for training and scoring deep learning models.

Summary

This paper has provided detailed information about how deep learning works in terms of deep neural network architectures such as convolutional, recurrent, feedforward and autoencoder neural networks. Applications of deep neural networks in computer vision, AI assistants and bots, time series analysis, natural language processing and rare-event modeling were also discussed. Lastly, the deepLearn CAS action set developed by SAS was presented with an overview of the process to build deep neural network models using that action set as outlined. As advances in deep learning are made, SAS will also continue to advance its deepLearn CAS action set.

Learn More

SAS deep learning capabilities are included in SAS Visual Data Mining and Machine Learning. Give it a free try at sas.com/tryvdmml.

Endnotes

- i Waldron, Mike (2016). "10 Deep Learning Terms Explained in Simple English." Retrieved from <https://www.datasciencecentral.com/profiles/blogs/10-deep-learning-terms-explained-in-simple-english>.
- ii Dietz, Michael (2017). "Understand Deep Residual Networks - A Simple, Modular Learning Framework That Has Redefined State-of-the-Art." Retrieved from <https://blog.waya.ai/deep-residual-learning-9610bb62c355>.
- iii Ioffe, Sergey and Szegedy, Christian (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Retrieved from <http://proceedings.mlr.press/v37/ioffe15.pdf>.
- iv Tamang, Apil (2017). "Yet Another ResNet Tutorial (or Not)." Retrieved from <https://medium.com/@apiltamang/yet-another-resnet-tutorial-or-not-f6dd9515fcd7>.
- v Dertat, Arden (2017). "Applied Deep Learning - Part 3: Autoencoders." Retrieved from <https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798>.
- vi Pelk, Henk (2017). "Natural Language Processing and Machine Learning." Retrieved from <https://chatbotsmagazine.com/natural-language-processing-and-machine-learning-the-core-of-the-modern-smart-chatbot-8755c6343fa5>.
- vii Pant, Neelabh (2017). "A Guide for Time Series Prediction Using Recurrent Neural Networks (LSTMs)." Retrieved from <https://blog.statsbot.co/time-series-prediction-using-recurrent-neural-networks-lstms-807fa6ca7f>.

To contact your local SAS office, please visit: sas.com/offices

