Paper SAS2103-2018

Real-Time Image Processing and Analytics Using SAS® Event Stream Processing

Frederic Combaneyre, SAS Institute Inc., Cary, NC

ABSTRACT

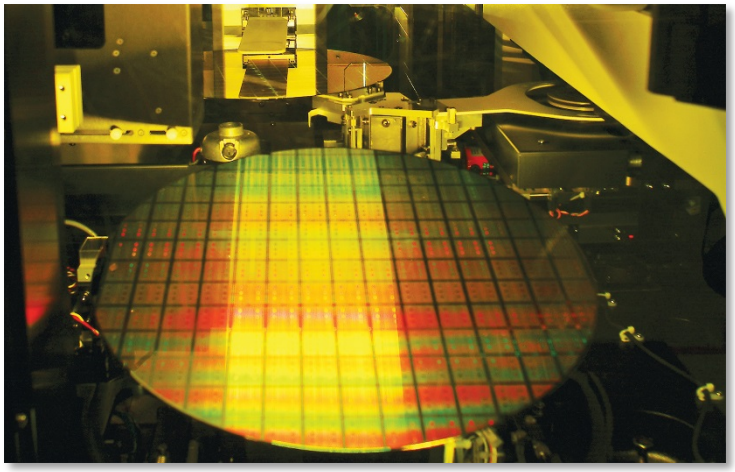
Image processing is not new and has been here for a while. But now, with the advent of the Internet of Things, this has become one of the most promising and exciting applications of real-time streaming analytics technologies. Historically focused on character or face recognition, the image-processing world has evolved to include enhanced predictive and prescriptive analytics applications using real-time streams of rich image and video content. From predictive maintenance in manufacturing to health care or security, there are countless examples of image analytics domain applications. In this session, we explore how to implement image analysis with SAS® Event Stream Processing and how to deploy image-processing analytic models created with SAS® Viya®.

INTRODUCTION

Images are taking over everywhere. Social media is seeing constant growth of images created and shared over networks. Whether it is a photo, a selfie, or a video, our world is increasingly filled with more images and videos and less text data. Image analysis (also known as “computer vision”) is the ability of computers to recognize attributes within images. Smartphone apps such as Google Photos and Apple Photos use some basic image analysis features to recognize faces and categorize them.

Analyzing images is not restricted to photo applications and social media. Industries are now also using images as a medium to extract more insightful data for their business needs. For example, innovations in the semiconductor industry drive the ability to do more on ever smaller machines, but perhaps equally important is the ability to optimize manufacturing processes. In the digital printing of semiconductor components, a one-in-a-billion failure rate for droplets might sound like an acceptable rate. This is less so when considering that up to 50 million droplets can be pushed per second, leading to an unacceptable defect rate of one every 20 seconds.

One very large semiconductor company is using image processing and machine learning to automatically detect defective wafers early in the manufacturing process in order to avoid low yield. Wafers are thin slices of semiconductor material, such as a crystalline silicon, used in electronics for the fabrication of integrated circuits. These wafers have 570 chips on them and each chip contains more than 2 billion transistors.

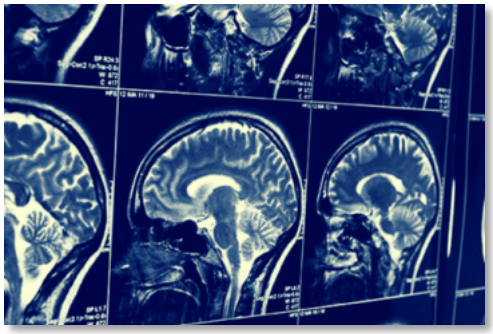


The company’s goal is to cut the number of defective wafers by one-third, which will have a big impact on the bottom line. It is a very complex manufacturing process that involves forming the transistors though two-dimensional printing and connecting them through a multi-layer process.

During the production process, pictures of the wafers are taken in a very specific way and then, in addition to the use of usual signal data coming from sensors, these images are processed, analyzed using different techniques such as convolutional neural networks, and correlated with a library of 40+

known defect characteristics in order to detect wafers with unacceptable characteristics early and take any appropriate actions. Because time to decision is important, preemptive analytics on streaming sensor and image data play a key role in finding indications of where and when defects are looming.

In a very different domain like health care, some researchers estimate that medical images, which are becoming the largest and fastest-growing data sources in the industry, account for at least 90 percent of all medical data and this amount continues to grow. From daily X-ray images in a dentist office to three-dimensional magnetic resonance imaging in hospitals, images have become the biggest driver of data growth in health care, generating millions of terabytes of data every year just in the United States alone. For example, in order to estimate the weight of a baby during pregnancy, radiologists measure the circumference of the baby's head. Technicians select the image they think will be the most appropriate for the measurement and manually contour the outline of the baby's head, thus making this a very subjective and time-consuming process. When considering the human element and the fact that most radiologists are presented with hundreds or even thousands of images daily, new methods are required to extract insights from those images more efficiently. Due to this volume and to the nature of images in general, extensive cognitive input is required to process this data. Whether it is to automate the work of radiologists or to detect diseases that would have remained undetected by the human eye, biological image analytics using deep learning and artificial intelligence (AI) techniques will have a tremendous impact on medical imaging in the near future.



SciSports is a Dutch sports analytics company founded in 2012 that advises and builds solutions to provide professional football organizations with the competitive edge needed to maintain or improve their ranking and status. Because traditional football data companies focus only on the player that has the ball, thereby leaving the performance of the other players undocumented, SciSports wanted to take advantage of the immense amount of data happening away from the ball in order to provide a more complete picture for scouts, coaches, and clubs hoping to find the next rising star or undervalued player. The goal is to calculate the quality, talent, and value of more than 200,000 players to help football clubs make better, and often very lucrative, decisions about player acquisitions, as well as to make way for innovations in everything from player recruitment to virtual reality for fans.

SciSports uses computer vision to track and monitor football players and extract many different types of metrics. Instead of having players wearing devices during matches, the SciSports method consists of setting up 14 cameras in stadiums to capture all movements on the field and analyzing images in real time to track players positions, actions, direction and speed of passing, sprinting strength, jumping strength, player movements, and how close the ball stays at the foot after a first touch. Advantages of this approach as compared to using wearable devices on players are that data about opponents and about the ball are available, in addition to data about the team. This is something that will be extremely valuable for SciSports' customers and will be a strong differentiator in the market.

Using deep learning and AI capabilities, SAS is helping SciSports by developing object detection models, including very specific YOLO (You Only Look Once) models, to extract information in real time during matches. The ability to deploy deep learning models using SAS Event Stream Processing on small GPU-enabled devices, move onto cameras at the edge or in memory, and then perform inferencing in real time is cutting-edge science. These examples reflect the

ever-increasing number of use cases that are benefiting from performing image analytics on streaming data.

SAS STREAMING ANALYTICS LIFECYCLE

Implementing an image analytics model for either image classification or object detection follows the SAS streaming analytics lifecycle depicted in Figure 1.

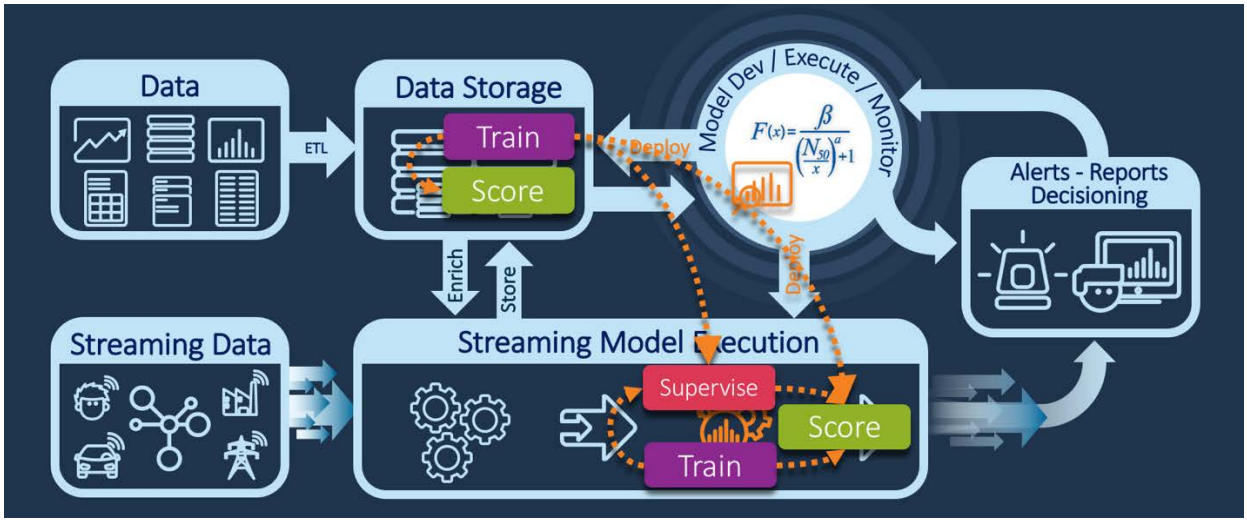


Figure 1. SAS Streaming Analytics Lifecycle

Before deploying model and score images being streamed in real time, models must first be developed, trained, and tested using training data sets. This is usually done using SAS® Visual Data Mining and Machine Learning or through any SAS Scripting Wrapper for Analytics Transfer (SWAT) clients such as Python. Once a model has been developed and tested, it will be deployed in a SAS Event Stream Processing project that is dedicated to run models on real-time data streams.

This paper focuses on the deployment of models with SAS Event Stream Processing, so details of model development have been omitted. At the time of this writing, multiple deep learning algorithms dedicated to image analysis (for example, convolutional neural networks and VGG16) are already supported in SAS Viya 3.3, and others such as recurrent neural networks and YOLO models will be added in future releases.

SAS® EVENT STREAM PROCESSING AND IMAGES

SAS® Event Stream Processing Blob Data Type

The architecture of SAS Event Stream Processing is designed to process events, and thus it processes images as events. SAS Event Stream Processing 4.3 has introduced a new “blob” data type that is dedicated to storing binary data, including images and sounds. It also natively recognizes image formats such as *.jpg, *.png, *.gif, *.tiff, *.bmp, *.ico, and *.webp by reading binary header content, but any image format is supported and can be stored into a blob field to be processed by SAS Event Stream Processing.

Publishing Images to SAS® Event Stream Processing

Images published to SAS Event Stream Processing can be single pictures, or they can be produced by video cameras. In the latter case, video streams must first be converted into single frames, and each of these frames must be published to the SAS Event Stream Processing server in multiple events, because most video formats are dedicated formats that wrap content into specific data structures optimized for broadcasting video content. This video-to-frames conversion can be done in a dedicated preprocess step before publishing the event, directly in the camera, or simply via a SAS Event Stream Processing connector or adapter.

SAS Event Stream Processing currently provides out-of-the-box connectors for USB video class (UVC) cameras, as well as for Basler industrial cameras. These connectors stream video content directly into SAS Event Stream Processing events. Although images are binary data structures, it is often useful to be able to transmit and publish them using universal data structures. All SAS Event Stream Processing connectors and adapters, as well as the REST and the web socket API, support receiving images in Base64 format. Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format. These encodings are necessary for transmission of data when the channel does not allow binary data. When converting an image using Base64, the resulting Base64 data will only contain 64 different ASCII characters, all of which can reliably be transferred across systems that might corrupt the raw source bytes (for example, emails, HTTP, and text files). Once an image is converted to Base64, it can be published to SAS Event Stream Processing as text data, using (for example) a CSV data structure. If the destination field in the SAS Event Stream Processing source window schema is of type blob, the Base64 image data will be automatically and implicitly decoded and stored as a binary image in memory. Figure 2 presents the logical flow from the camera to the end user.

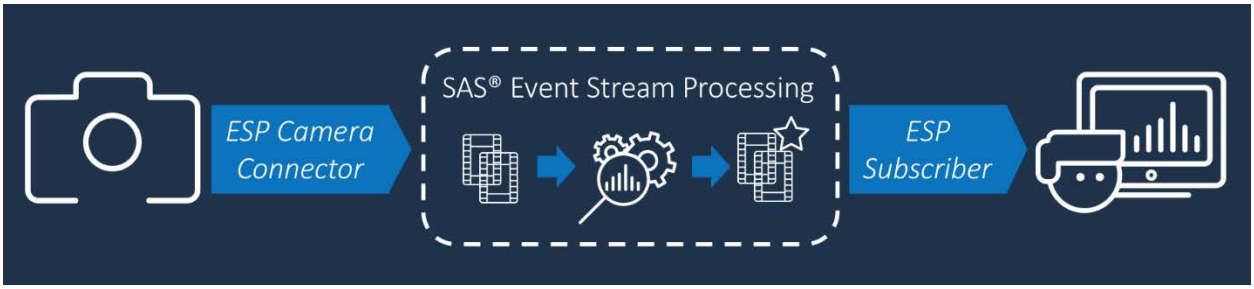


Figure 2. Logical Flow Schema

DEPLOYING ANALYTIC MODELS IN SAS® EVENT STREAM PROCESSING

Once a deep learning model has been developed, trained, and tested using SAS Visual Data Mining and Machine Learning or a SAS SWAT client, it must be stored in the SAS ASTORE file format in order to be deployed and used by SAS Event Stream Processing. The *SAS Visual Data Mining and Machine Learning User Guide* or the SAS SWAT client documentation provides more details about PROC ASTORE.

In order to run a deep learning model in SAS Event Stream Processing, two dedicated windows must be used, namely the Score window and the Model Reader window. In order to illustrate the deployment of such models, a SAS Event Stream Processing project that will receive images from a video camera and score these images using a deep learning model can be created. A very basic project will look like the one displayed in Figure 3.

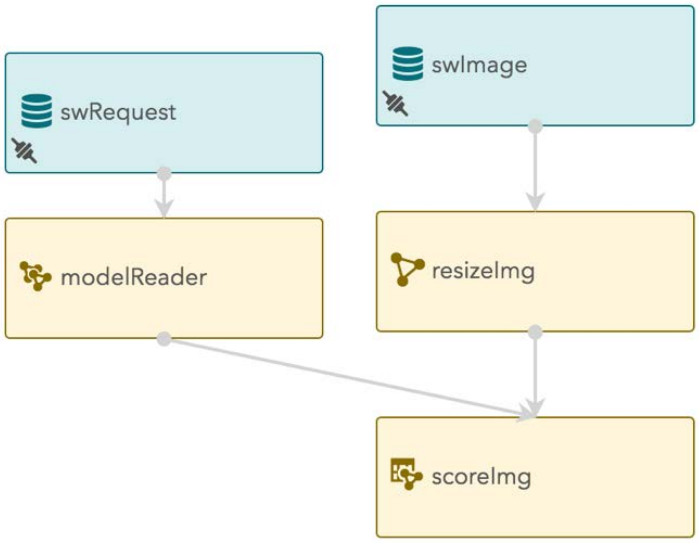


Figure 3. Sample SAS® Event Stream Processing Project for Deploying an Image Analytics Model

As shown in my example, the Source window **swImage** will receive the image events. A minimal schema is being used for this window, namely a key **`ID`** field of type int64, and an **`image`** field of type blob that will contain each of the images (Figure 4).

		Field
		id
		image

Figure 4. swImage Window Schema

The Score window **scoreImg** is responsible for running the ASTORE model and scoring each image that has been published. In order to set up this window accordingly with SAS Event Stream Processing Studio, the ASTORE model file is registered as an offline model source in the window's settings. When registering the ASTORE model using SAS Event Stream Processing Studio, the output schema model is automatically generated based on the ASTORE model output parameters. In this case, the model is a YOLO model that produces 589 output parameters, which, in addition to the event key field, represent 590 output fields in the window's schema. Figure 4 shows the first fields of the **swImage** window's schema.

		Field
🔑	#	id
	#	_nObjects_
	A	_Object0_
	#	_P_Object0_
	#	_Object0_x
	#	_Object0_y
	#	_Object0_width
	#	_Object0_height
	A	_Object1_
	#	_P_Object1_
	#	_Object1_x
	#	_Object1_y
	#	_Object1_width
	#	_Object1_height

Figure 4. scoreImg Window Schema (Extract)

Most deep learning neural networks models dedicated to process images require some preprocessing operations to be performed on the images before running the scoring. For example, models such as convolutional neural networks are trained on fixed image size and require input images to be resized to that size in order to provide consistent results. Other preprocessing steps might include cropping or inverting the image, changing colors, applying filters, converting the image to a specific binary format, and many other image manipulations that might be required for the model to operate and provide accurate results. In order to perform these preprocessing steps on the input image event, a Calculate window is being used in my example. In our project, the Calculate window **resizeImg** is using the **`ImageProcessing`** algorithm that executed a **`resize`** operation on the images, so 448 x 448 images (the size required by this model) can be provided to the scoring window. Figure 5 shows a screenshot of the main parameters.

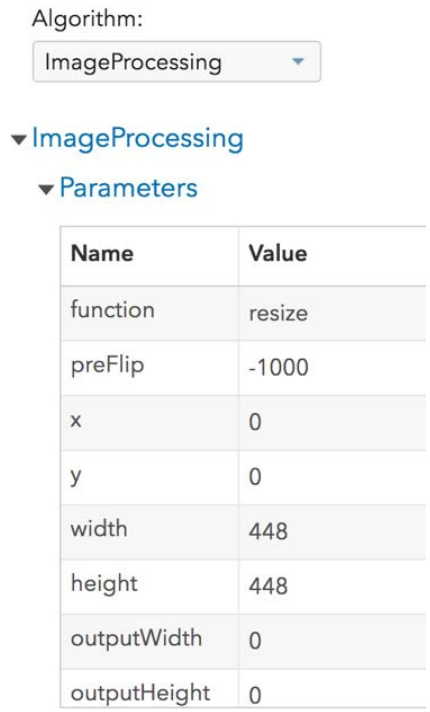


Figure 5. resizeImg Window Parameters

In SAS Event Stream Processing 5.1, two additional windows are required when running an ASTORE model, namely a Model Reader window and a second Source window. In my example, the Model Reader window **modelReader** is responsible for reading the ASTORE file and providing the binary model to the Score window **scoreImg**. This window has no parameters or schema to define. The details about the ASTORE filename and disk path are provided dynamically through request events.

Request events are standard events that are sent to SAS Event Stream Processing through source windows using usual publishers and are dedicated to request specific operations to be performed by the SAS Event Stream Processing analytics windows. Windows that support these requests are the Model Reader, Score, Train, and Model Supervisor windows. In my example, the Source window **swRequest** is responsible for receiving the request events for the **modelReader** window**.** This is the standard schema for all request events. The **swRequest** window schema is shown in Figure 6.

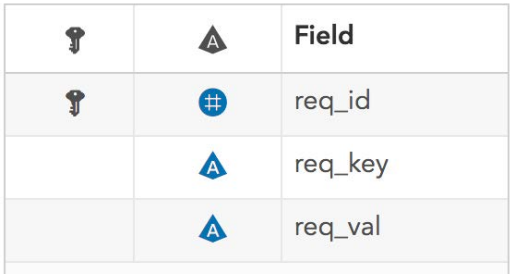


Figure 6. swRequest Window Schema

Because these request events must be sent dynamically, one of the simplest ways to publish the initial request is to use a file socket connector and put the request events in a CSV file. In my example, the following request events are being published to the **swRequest** window:

```
i,n,1,"action","load"
i,n,2,"type","astore"
i,n,3,"reference","/home/sas/astore/yolo_rectformat.ast
i,n,4,,
```

Using the content of these request events, the **modelReader** window will load the ASTORE file **`/home/sas/astore/yolo_rectformat.ast`** and send the binary internal model to the **scoreImg** window to perform the scoring on the images. One very interesting property of these request events is that they allow changing the analytic models that are used for the scoring process dynamically, without stopping the streaming flow. For example, if the analyst considers that the deployed model is drifting, he can dynamically deploy an updated model or change some parameters.

Once the SAS Event Stream Processing project has been fully created, it is ready to be started and the model can be executed on streaming images. In order to publish images to be scored by this model, a camera connector such as the UVC or the Basler Pylon connector connects to an external source that publishes some images as event streams to the SAS Event Stream Processing project, or eventually publishes images encoded as Base64 in CSV files. A connector can also be created using the C, Java, or Python Pub/Sub API in order to connect to a specific image source. In my example, the YOLO model is performing object detection. It outputs the coordinates of the bounding box of the objects it has detected on an image. These boxes can then easily be displayed on the image using a client such as Python ESPy that is subscribing to the Score window **scoreImg**, as shown in **Figure 7**.

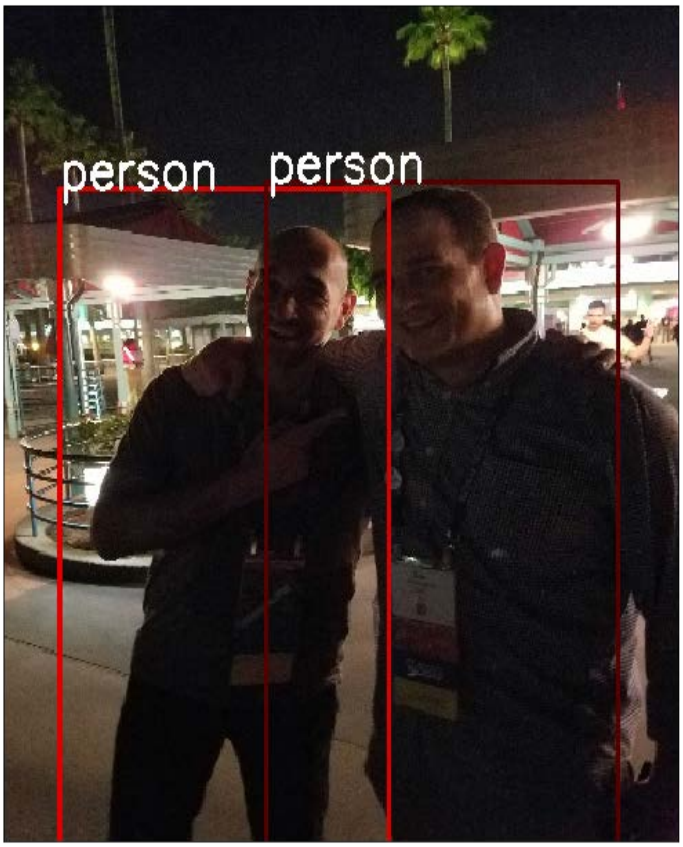


Figure 7. YOLO Model Image Output Showing Detection Boxes

PERFORMANCE CONSIDERATIONS

It is quite common for deep learning models to be process-intensive, as they are often using many layers and/or thousands or millions of parameters. While these processing resources can be easily managed when the model is deployed as a batch process, it can become a critical issue when streaming data is being processed flowing at high throughput and require low latency output. It is even more critical when

the model is deployed at the edge, on a gateway, or on a device or a camera, as these components offer limited processing capabilities and memory resources.

SAS Event Stream Processing 5.2 will provide support of GPUs for processing deep learning models. GPUs usually provide a very good acceleration for processing neural networks. However, even with GPU acceleration, this performance boost might not always be sufficient to meet the required throughput and latency of a streaming application. A model deployed on an edge device will have to comply with the device's available resources. Some edge devices are now equipped with GPU cards, which makes them very good targets for such deployments. Even then, however, it must be ensured that the model will be fast enough for the required task and will have the space to run on the device's memory. When the number of layers and the number of parameters of a neural network model are increased, memory size and the processing resources required are also increased, which is often directly related to accuracy. This means that deployment targets must be considered when developing such models, as there will most likely have to be compromise among performance, resources, and accuracy.

CONCLUSION

SAS Event Stream Processing 5.1 introduces new fantastic ways of performing analytics on image streams. Although executing neural networks or deep learning models on pictures or videos is not new and has been a very common practice for the last 20 years in many different domains such as social media, photo apps, or CCTV, the level of flexibility that SAS Event Stream Processing brings for deploying such models is currently unmatched. Analysts can now develop deep learning models in a fully integrated environment such as SAS Visual Data Mining and Machine Learning and then directly deploy the same models for scoring streaming events in the cloud, on premises, or at the edge thanks to SAS Event Stream Processing.

More and more industries and activities are starting to use images as one of their main data sources for driving business. I strongly believe that applying deep learning models on images in real time and extracting substantial insights from this data in order to reduce the time to decision, improve the accuracy of decisions, reduce workload, and develop new business models will be a key driver for the future and will drive most of the innovations that will happen in the next 10 years.

REFERENCES

- "Base64." Wikipedia. Available https://en.wikipedia.org/wiki/Base64.
- Lee, June-Goo, PhD, et al. 2017. "Deep Learning in Medical Imaging: General Overview." *Korean Journal of Radiology*. Available https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5447633/.
- Paiva, Omir Antunes and Luciano M. Prevedello. 2017. "The potential impact of artificial intelligence in radiology." *Radiologia Brasileira*. Available https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5656066/.
- SAS Institute Inc. 2018. *SAS® Event Stream Processing 5.1: What's New.* Cary, NC: SAS Institute Inc. Available http://pubshelpcenter.unx.sas.com:8080/test/?cdcId=espcdc&cdcVersion=5.1&docsetId=espwn&docsetTarget=p1adzsdqbmhlztn1mb3wssvv0ef0.htm&locale=en.
- "SAS Scripting Wrapper for Analytics Transfer (SWAT)." GitHub. Available https://github.com/sassoftware/python-swat/.
- *SciSports SAS Customer Story*. Cary, NC: SAS Institute Inc. Available https://www.sas.com/en_us/customers/scisports.html.
- Redmon, Joseph, and Ali Farhadi. "YOLO9000: Better, Faster, Stronger." *Cornell University Library Computer Science*, December 25, 2016. Available https://arxiv.org/abs/1612.08242.

RECOMMENDED READING

- *SAS® Event Stream Processing User Guide*
- *SAS® Visual Data Mining and Machine Learning User Guide*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Frederic Combaneyre
SAS Institute Inc.
frederic.combaneyre@sas.com