REFINITIV STREETEVENTS

EDITED TRANSCRIPT

NVDA.OQ - NVIDIA Corp - Accelerate AI Model Creation Using AutoML in NVIDIA TAO 4.0 Webinar

EVENT DATE/TIME: FEBRUARY 15, 2023 / 4:00PM GMT

REFINITIV

CONFERENCE CALL PARTICIPANTS

Chintan Shah

Debraj Sinha

Varun Praveen

PRESENTATION

Debraj Sinha

Hi, everyone. Thank you for joining us today for our webinar on TAO. I am Debraj Sinha and I'm part of the Metropolis Marketing team at NVIDIA, and I'll be the moderator for today's session.

Before we begin, I'd like to cover a few housekeeping items. All of the windows on your screen are resizable and movable. At the bottom of your screen are some icons that offer more information. We really want this to be as interactive as possible, so if you have any questions, comments, please submit them throughout the talk using the Q&A window locating to the right of the slide. We'll answer them after the presentation.

So let's get started. First, I would like to introduce you to our speaker, Chintan Shah, who is the Product Manager for TAO here at NVIDIA. And here are the agenda -- here are the topics for today.

We'll be covering our latest release, TAO 4.0, talking about its features and concentrating one of its main feature AutoML. AutoML automatically tunes your AI model to find the best model for your dataset. And we have a really cool demo to show you how you can deploy AutoML with TAO.

So let's get started. Chintan, over to you.

Chintan Shah

All right. Thank you, Debraj. Hello, everyone. Good morning, good afternoon, good evening. My name is Chintan Shah. I'm the Product Manager at NVIDIA, and I am responsible for TAO.

So today, I'm going to cover some of the key features in TAO as well as one of the key features that we released in 4.0, AutoML.

So before we begin, let's talk about some of the challenges that we've seen in our developers face when building AI. Well, first, there's multiple training frameworks. You have learn how to use TensorFlow, PyTorch, MXNet, and where do you start? Once you pick your training framework, there's literally hundreds or more model architectures to choose from.

And there's CNN models, there's Transformer model. There's all kinds of different models that the developers can choose from and that can be very overwhelming at time. And once you do that, you've got to pick a dataset, generally, a large -- you have to collect large dataset if you want to train something accurately.

And then once you're ready to deploy your model, there might be multiple deployment options, you might have to optimize it for Edge or you might have to optimize it for large data centers. So there's a lot of options that you will have to think about when you're creating this model.

And again, we're only talking about one use case. Once you multiply this by n-number of use cases, the challenges just grows exponentially. How can TAO help here?

Well, NVIDIA TAO is a low-code toolkit. It builds on top of TensorFlow and PyTorch, but kind of abstracts away the complexity of AI and deep learning framework.

The idea here is that you start with one of NVIDIA provided pre-trained models, and there are hundreds of model architectures to choose from classification, object detection, segmentation to other use cases like key point estimation or you can start with one of a task-based model. So things like people detection, vehicle detection, pose estimation, license plate recognition.

Or there's another option, and you can bring your own model right into TAO. With TAO, TAO provides with simple interface, 4 or 5 simple commands to go from data to a production quality model. You can augment your data with TAO and then you can train your data, train your model on your dataset.

And then for optimization, TAO provide several turnkey optimization, like model pruning and gaze quantization that can really reduce the size of the model, make your model run faster to inference. We can achieve up to 4x speed up by using all the different options that you have.

And then once the model comes out of TAO, it's a production model that can be deployed in various NVIDIA frameworks, like DeepStream or streaming video analytics job, for Triton, for inference serving use cases. And best of all, TAO, is fully supported by the NVIDIA AI Enterprise, so if you want enterprise support you can use NVIDIA AI Enterprise to get the enterprise support.

One of the things that we have at TAO are the pre-trained models. And these are just some of the pre-trained models that we have, various use cases from people detection, segmentation to gaze estimation, key point estimation. We have a very accurate city segmentation models, all the way to license plate recognition. And these are trained on a lot of our own internal data that we have collected and labeled over the years. And we have very comprehensive model reservoir, model cart that you can find on NGC to understand the use cases, the dataset that this was trained on.

And you have multiple options. You can use the model as is or you also have the option of transfer learning on our own dataset with TAO. And this opens up the possibility to improving the accuracy of the model for your use cases, for your camera angles, et cetera.

So with that said, let me talk about the other class of pre-trained models. So these are more general purpose pre-trained model. These are -- think of it as a pre-trained weights that developers can start it.

These are popular model architectures from image classification, the object detection, segmentation. And the option for backbones we have 15 plus different backbones to choose, from a small ResNet10 to larger ResNet101 model to the DarkNet model and also the newer EfficientNet models and the MiT, which is the new transformer model.

All these models are publicly available on NVC. They're trained on a public OpenImage datasets. And you can pick and choose the combination of backbone with a particular task that you're working with.

The 2 new ones that we have released in 4.0 are highlighted here in green. One is Deformable DETR, which is a state-of-the-art object detection model. And other is a SegFormer which is a transformer-based semantic segmentation model that's been trained on 24 classes for segmenting different objects, different process in a city use cases.

And with that said, one thing I want to talk about before I move to our TAO 4.0 feature is running TAO on the cloud. TAO is cloud-agnostic, can be run on any cloud and integrate with various cloud services.

We've tested on 3 cloud providers -- the 3 leading cloud providers. You can use TAO at the infrastructure level. Think of it as like the IAS stack where you spin up and manage your own VMs from CSP and you can use AWS, GCP or Azure. One level up are the PaaS or the platform as a service layer where you -- where TAO can be integrated with various managed CSP services.

On AWS, you can run TAO and then Amazon Kubernetes cluster, the EKS. And we provide a script, which deploys TAO services on EKS. For GCP, we integrated with Vertex AI. And Vertex AI is a fully end-to-end MM platform where you can use TAO notebooks directly on Vertex AI with -- directly deployed on Vertex AI. And Colab is more for quick experimentation where this is one of the new features that we have in 4.0 where you can run TAO directly in Colab.

And then on Azure, you can integrate it with the Azure's Kubernetes Service, AKS. We provide instruction to run TAO on AKS. And recently, we also integrated with Azure Machine Learning, and the idea here is that you can run TAO notebooks with the Azure Machine Learning services. Recently, we published a very detailed blog and a video on deploying TAO on Azure Machine Learning.

One of the core features of TAO is REST API. This is something that we have released sometime middle of last year. And the way to use the API is through our API services. TAO API service, again, it's a fully managed -- sorry, it's a self-managed TAO service, you deploy it on your own Kubernetes server, and this could be on-prem, cloud or any cloud manager Kubernetes services. And Kubernetes bring the resiliency and scalability, so you can scale up or down based on demand.

The idea here is that IT teams can spin up their own service on their own infrastructure. Think of it like IT teams that enterprises can manage their TAO services and make it easy for their AI practitioners to quickly get started with TAO. And then once the service has been deployed, users can interface it with standard REST APIs and -- which allows them to innovate into their existing platform or create a UI that AI teams can use without any learning curve.

There are also benefits for someone like an AI service provider, to run the map start-up or data allocation provider. They can quickly build on top of TAO to provide additional training services to their customers.

The way it works is, as a user, you interface with the API services, which will automatically manage and orchestrate their compute, it will pull the right container from NGC. And I'll show you this in a demo later on. And we feel like this will really open up TAO to make it integrate into other platforms.

So let me walk you through the entire API workflow. So given REST API is very easy with TAO, and once the service is up and running, there are a few simple steps. So starting from the left, you start by creating -- first, you create a dataset object and then you create an object for your training data, validation data, and you upload and convert your dataset to the format that is suitable for that model. And again, these are just simple API commands for each of these.

In parallel, you start by creating your model object. In this case, you create, let's say, you want to train YOLO model, you create a YOLOV4 model. Next, you will assign a pre-trained weight. So let's say, if you want to use your own pre-trained weights or if you want to use one of the pre-trained weights from NGC, you assign your pre-trained weight, then you assign the dataset. So the dataset that you created, you have sign that to this particular model object. Once all of that is done, then the next step is to kick off your training job, evaluate it and export it where you can use it for your -- in your application.

The table at the bottom kind of shows a few simple tasks that we have in the simple API call. More information can be found in our API guide in our documentation.

So with that said, let me now quickly jump into some of the key features that we have in TAO 4.0. I'll talk briefly about Google Colab. Google Colab is a free platform from Google where you can run your lightweight AI jobs in the cloud. They provide free compute resources in the cloud. They have free GPU resources in the cloud. And the idea here is that if you want to quickly experiment, Google Colab. And we have several notebooks that with the one click the use and click directly opens it in Colab and they can go through and step through the training notebook.

In the past, if you had to do -- if you were to just quickly try it, you would to select the VM, you have to configure the VM. It's all a prerequisite and then you can do your training. But now with Colab, the simple one-click notebook, you can experiment faster, you can try out TAO without having to do any setup and you can use your own -- and it's free to use resources in the cloud.

Another feature that we have is we are working with a lot of MLOPS providers for various things. So here is kind of the entire into end to end ML workflow, from data to creating your AI, to creating -- to validating your model and then deploying your model.

And TAO kind of provides this training parts from building models to training and optimization. So it's very important that TAO fits well with the larger ecosystem of MLOPS. With our 4.0 release, we're partnering with Weights & Biases and ClearML for experiment tracking, module management.

So the idea is that users can use TAO like they've done it in the past, provide few configuration, provide the API fee for the respective MLOPS platform. And automatically, they can view their progress in one of the platforms and see how their runs, how their experiments are doing. It allows them to quickly create, run. It allows them to automate and create -- and reproduce results in the future.

I talked briefly about our transformer models, but let me kind of provide why transformer are the -- are becoming the next big thing. Few things that we see with the transformer-based model that where we could not feed with the traditional CNN model is our robustness.

Here, if you see this very noisy image, you can see that this image is extremely difficult to process with the CNN model. With the CNN model, this would not be able to segment very well.

But with the SegFormer model that we released, we can at least see a very good segmentation map. Again, this is extremely difficult image to process. So with the noisy image like this, it does a reasonable job of segmenting person, road, vehicles on the road, et cetera.

Another thing with the segmentation model -- sorry, another thing with the transformer model is, in generalizable is much better than a CNN model due to various things. So if I look at this image, right? So let's say you have an image of a duck that's likely what you used to train your model.

But as we start moving away from an image to just a sketch of duck, a CNN model, if you look at the activation -- heatmap activation, you can see that it's kind of all over the place. It does reasonably well when it's a real image. But as you start getting into the sketch and a quickdraw the heatmap are all over the place. So it's not able to predict -- able to accurately predict what this object is.

If you compare that with the transformer model, you can kind of see that the heatmap activation is kind of around the duck. So it's able to generalize that -- from an picture of a duck that this likely a sketch or a quickdraw of a duck.

Another thing is accuracy. I mean, as you provide more and more data to your model for a CNN model, like ResNet50, the accuracy starts to cap off after a certain level. But with the transformer based model, what you're seeing is more samples you provide, the accuracy improves even further. The tail is much further along than a CNN model. And this makes the model more generalized and you can use it for fine-tuning it on your own dataset. So it makes for better fine-tuning than CNN models. As I mentioned, we're releasing the Deformable DETR for object detection and SegFormer for semantic segmentation.

Now with that said, let me now jump into our core feature of 4.0 which is AutoML. So creating an accurate model is very tedious and time-consuming task. It requires some amount of expertise. A developer needs to choose from what models to pick, what hyper parameters to choose. And again, there's literally hundreds or more options that the developer has to choose from.

Question is, can this be automated? Absolutely. AutoML in the process of automating the manual task with data scientists and developers must compete as they build their AI models. And this includes things like choosing the model, right algorithm, creating models, running experiments, choosing the right set of hyper parameters and all of this is extremely time-consuming.

And AutoML can help developers accelerate this process of hyper parameter tuning without really needing deep learning expertise. It frees them from the broad task in building a model and it allows them to focus on extracting insight to solve important business problems. And how can you do this with TAO?

So in TAO 4.0, we enable AutoML and we enable AutoML in 4.0, and what we provide is smart hyper parameters suite and optimization. We support AutoML on large selection of models. As the table shown here, we support on various object detection model, classification, segmentation and also our OCR model, the license recognition. The value of AutoML is it reduces the manual task of hyper parameter optimization. It reduces time and you can get your model faster.

So the graph on the right kind of shows the accuracy improvement that you might get for some of these models from like the baseline spec to renewed AutoML. So the dark green bar shows the baseline accuracy, which is the default config file that comes with TAO and. And now by using AutoML on a particular dataset, it can improve it by good amount. And the improvements can be anywhere from like 5% to all the way to like 20%.

And the model, the accuracy is very dependent -- the accuracy is really dependent on the size of the dataset, the type of objects you have and then there crosses. So it's very difficult to have one hyper parameter for all models. So in this case, by using AutoML, we can quickly get a better models faster.

Best of all, it's very easy to use. You don't really need any expertise. We provide Jupyter Notebooks, you can quickly step through each of the steps in your notebook and I'll show you in a demo what we mean.

And on the other side, it's also fully configured. So if you're an expert, if you know what you're doing, we provide all the hooks to select your own hyper parameters or select different set of hyper parameters to suite, you can select different types of AutoML algorithms for your experimentation. So it's really catered for all developers.

So let me kind of walk you through different AutoML algorithms that we have in TAO. So TAO support kind of 2 main algorithms. One is Bayesian, and Bayesian will use this adaptive, selects the hyper parameter based on past experiments. It runs serially one experiment after another. So let's say, in an example, you had this red line, which is kind of your default model. And what Bayesian will do is it will -- when it starts, it randomly fix a set of hyper parameters that you start with.

And then, let's say, you start with one experiment. And your first experiment gives you this accurately. And eventually, based on this, it will start learning. So it kicks off another set of experiments. So let's say, your next sort of experiment your accuracy improves even more.

And then after that, based on the learnings of the previous, it kicks off another experiment that improves it even more until you reach the best model which will have gotten the -- which will have been learned from all the previous experiments before that.

If you're using a multi-GPU setup, each experiment will run parallel across all the GPUs. Bayesian is more accurate than the other algorithm which is Hyperband. But again, it's slightly slower because you need to run all experiment to completion.

On the other side, we have Hyperband, which selects hyper parameter based on this concept of successive halving. It allocates the resources based on the number of runs it had. So kind of let me show you an example here.

So when you start, it starts with a large number of recommendations. So in this case, again, in this example, I'm showing it starts with, let's say, 8 set of recommendations. After each iteration, what it does is it kind of halves the number of iteration going forward. So instead of 8, for iteration 1, you will go down to 4, and it will drop the other -- drop the worst 4 selection in this case. And here, we're looking at the loss. So it will drop all the 4 which are higher lost than the other one.

And then eventually, iteration 2, it runs for a longer period of time than iteration 1, but now we only have 2 runs going on. And then the last iteration is where you have a single run, or single recommendation, where you will -- your -- out of other things is your best model.

And again, this example kind of shows 8, but you likely start with a larger number of initial recommendations, probably 80 to 100 and that's one thing that we have. It's fully configurable, you can configure the number of runs. You can also configure the ratio of halving.

So here I'm kind of showing that it reduces by half every iteration or you can -- like that you can choose to reduce it by 1/3, 1/4, et cetera. It's slightly faster than the Bayesian because you're not running -- because you're not running all the iteration to completion. You are kind of only -- you're really only running one to completion.

So how do you get started with AutoML? It's fairly simple, a few simple steps and all of this is documented in a blog that we published when we released 4.0. Step 1, you can download all the TAO resources from NGC. And this package contains the Jupyter Notebooks. This contains the scripts that are needed to deploy TAO API, either on cloud or Bare Metal. And then once you download it, next you deploy your API server, and this happens via either the script or you can do this manually yourselves. Once you deploy it, the last step is walking through the Jupyter Notebook to do all your training.

So you will start from like selecting the model topology, treating your dataset, uploading your dataset, converting it and then configuring your AutoML parameters, training it and then lastly, comparing the models. And again, I'll show all this entire workflow in more details in demo coming up.

Before I kind of go into demo, I want to quickly talk about our architecture for our AutoML orchestration. So one of the requirements to run AutoML with TAO, is you need to use our APIs. And the reason that we need the API is because it needs to allocate all the resources and the way we allocate it is to manage it to Kubernetes.

The kind of 3 key components, you have the API server, then you have the shared NFS storage and one lastly, you have the compute cluster. And this is where all your training jobs will be run.

As a user, you can interface it with 1 of 3 days, you can use our Jupyter Notebook to go through the training process. You can also use our CLI. CLI is our lightweight client CLI application, and it allows you to interface with the API remotely. And third is our HTTPS or directly via REST API. And the idea here is that you can use this called it API from your own application or you can build a UI on top of it.

All of this information -- all of this API connects to your API gateway and API gateway kind of manages the NGC registry. And what it does is, it's responsible for pulling models, pulling containers from NGC. It can also -- it also handles the data upload from local space. It also sends the experiment data models back to the local space.

And then underneath this API gateway there's kind of this AutoML handler. And what this handler does is, it defines a job, attaches the dataset that you want to use to the job, create the config for the model that you are training and then it initiates this AutoML controller.

And AutoML controller, it really does the hyper parameters search. It evaluates the results, evaluates the output from completed run, gets the data and then recommends the next set of hyper parameter.

And then lastly, there's the workflow manager. This is kind of responsible for queuing and managing job on a Kubernetes cluster. This works more at a system level. So what this means is, there's one virtual manager, and this could be shared across many different runs. So you could have multiple jobs, multiple AutoML jobs, multiple training jobs, but you will have one single workflow manager.

Again, as a user, you only have to interface with the API and the rest will be handled by the API server and other components. Now let's see the entire AutoML flow in action.

(presentation)

Chintan Shah

All right. Hopefully, you guys were able to follow the demo. So before I wrap up, I want to talk about -- you want to kind of summarize with some of the key features that we have in TAO. I talked a lot about our AutoML and how you can use AutoML to improve accuracy and get to your models as quickly as possible, along with the APIs needed to run AutoML.

There are several data augmentation techniques in TAO, spatial augmentation as well as color space augmentation. There is augmentation available offline, meaning that you can take a dataset, augment it and then you get a new dataset, which could be same size or larger size. Or we also have online augmentation. So basically, every time it runs through training every e path the datasets will look different. Lots of option in augmentation.

Another feature that we have is being able to bring your own model weights in ONNX format into TAO. And the idea here is that you can bring classification or segmentation weights into TAO and then use the TAO capability to train, transform as well as optimize being able to kind of prune and quantize your model before deploying.

You can view all your training metrics with TensorBoard. TensorBoard, which is freely available as visualization toolkit from TensorFlow, allows you to look at your training loss, validation loss as well as a lot of other -- a lot of other statistics on your model.

And then lastly, improvement via pruning and quantization. Here, we're plotting it on our RestNet34, our PeopleNet model. Our pre-optimization and after pruning and quantization, we can see roughly 4x increase in inference performance across our different line of GPUs.

How do you get started with TAO? There are multiple ways to get started? You can use our launcher, which is our lightweight application -- lightweight CLI application. And the idea here is that you download this and it automatically pulls the right TensorFlow and PyTorch container. So users, in this case, would just interface with this launcher tool and not have to worry about what the containers to pick and, et cetera.

There are several containers, so this becomes slightly difficult, have different versions of TensorFlow and PyTorch. So if your launcher makes it slightly easier. But, obviously, there might be certain use cases where you need access to the container. You'll also have that option. You can directly pull on a container and run docker run command directly at container level.

And last but not least, is our APIs. I talked a lot about our APIs and how you can use to automate your task, you can to integrate into larger workflows or you can use it to build your own UI on top of our API. Multiple options depending on use case. You can use either that -- the link at the bottom, TAO you getting started kind of provide more information on each of these different workflows.

And before I jump into our Q&A, I do want to talk about our LaunchPad lab. It's a free lab on NVIDIA LaunchPad, which is a playground provided by NVIDIA AI Enterprise. And here you can run TAO, you can experiment with TAO and quickly learn how to use TAO. There are several curated notebooks to create a model with TAO and then deploying their model with DeepStream.

It's free to use and you get tools to access onto GPUs in the cloud, and you can use this time to kind of -- use this opportunity to learn a different software from NVIDIA AI. And it's -- as I mentioned earlier, TAO is pretty supported by NVIDIA AI Enterprise. If you're interested in enterprise support reach out to us for enterprise support.

And with that said, let's quickly jump into our Q&A section.

Before I go into Q&A, I do want to talk about our GTC coming up. Our GTC is on March 20 to 23. Please register with GTC. We'll have a talk about TAO. We will talk about our upcoming release TAO 5.0, which will have lots of newer models, new transformer models along with other things. So do you check out or do register for our section at GTC.

And with that, I will pass to Debraj to you for Q&A. Thank you.

QUESTIONS AND ANSWERS

Debraj Sinha

Thanks, Chintan for the great presentation. Let's begin the Q&A part of it. (Operator Instructions). And also joining us we have Varun Praveen, who is the Engineering Lead for TAO to help us with the Q&A also. So let's jump into it.

So first question to you, Chintan. Like, are TAO models supported on the Jetson AI platform then Edge AI Platform?

Chintan Shah

Yes. Good question. So TAO model can run on Jetson. If you're looking to train, training needs to still be on our logger GPUs, datacenter GPUs. But once the model is trained, you can deploy it on any Jetsons. So yes, the models can be deployed on Jetson.

Debraj Sinha

Got it. And does TAO support training visualization?

Chintan Shah

Sorry, can you put the question?

Debraj Sinha

Does TAO with AutoML, support training digitalization?

Chintan Shah

Yes. So with the AutoML, you can view the logs in TensorBoard. So we integrate with TensorBoard so you can view all your logs in TensorBoard. So when you set up your AutoML run, you configure or you turn on option for TensorBoard and it will dump TensorBoard logs and then you can open those logs into TensorBoard.

Debraj Sinha

Got it. So this question is to Varun. So, Varun, can we add custom model architectures to TAO?

Varun Praveen

At the moment, we can't bring -- entire custom model architecture to TAO. You can bring pre-trained model weights into TOA through the TAO, bring your own model weights converter which supports image classification and semantic segmentation. For other use cases like object detection and city segmentation we currently don't support bring in custom model architectures.

Debraj Sinha

Got it. And to you, Varun, will AutoML be able to be used with pre-trained model like PeopleNet?

Varun Praveen

Yes. So all our pre-trained models that -- that are released with TAO have a trainable version available with it on NGC. So you can use this as the pre-trained models in your AutoML job and then the rest is like any AutoML job.

Chintan Shah

I'll add a couple of things to that question, Debraj. So our pre-trained models like the PeopleNet and the TrafficNet, they're based on the TechNet and some of the other model architectures. So if you remember the table that I showed in the slide earlier, which had bunch of models for objection detection. So as long as the model -- the pre-trained model is one of those architecture, you can fine-tune it with using AutoML.

Debraj Sinha

Nice. And is DeepStream supported with TOA?

Chintan Shah

Is DeepStream supported with TAO? Yes. And what this means is, all TAO models or all the VM models that we have can be deployed into the DeepStream application. And we provide a turnkey application -- example application on GitHub to be able to take a module from TAO to -- directly into DeepStream.

Debraj Sinha

Got it. And how can I install TAO on a PC? And if I have several GPUs, does TAO uses all the GPUs?

Chintan Shah

Yes. So I'll take the first question. How do I install TAO on PC? I'm guessing the few requirements that we have is, one, we need a GPU. Second, we need a Linux. We currently do not support it on a Windows PC. But if it's a standard you go to Linux, then you can install on that PC.

As far as can I have several GPUs and TOA does use all GPU? That is correct. We do support multi-GPUs. We have one model with multiple GPUs, we support that. If you have a larger cluster with multi-nodes and each have larger number of GPUs, we also support that as well. So we support both multi-GPU as well as multi-nodes.

Debraj Sinha

Got it. So this question is to you, Chintan. Like does it mean TAO is an alternative to other deep learning applications we have? What are the main advantages of TAO compared to others?

Chintan Shah

Does it -- can you repeat your first question?

Debraj Sinha

Yes. So basically, asking like, what are the main advantages we have for TAO compared to other deep learning applications?

Chintan Shah

Yes. So the key advantages that you have with TAO is optimization, inference optimization, being able to not only get the best model, but also the most performant model on NVIDIA GPUs. And that can be the like model pruning, quantization and also being able to directly convert it into TensorRT. So when you use TAO you basically get all of those for free or without any effort. It's a single command that you have to use to convert it to TensorRT versus if you were to use other framework you would have to do a lot of optimization yourself.

The other advantage that we have is, we do provide a large selection of models, and you actually -- all you need is to know how to use config file to go from tiny model all the way to larger model. So it provide a lot of flexibility without having to be an expert, without having to be an AI expert or data scientists or researchers.

Debraj Sinha

Nice. And you touched upon TensorRT. So we have this question, like how does TAO differ from TensorRT? Can you elaborate on that?

Chintan Shah

Yes. Yes, definitely. So TAO is more of a training model application toolkit. TensorRT is more for inference runtime. So TAO, the model -- TAO is used to create a model that uses TensorFlow and PyTorch. But now to optimize the model to be able to run on TensorRT, TAO does leverage TensorRT, but again, it's only to generate a TensorRT engine. Now a user will have to take that engine file and then deploy it on their inference platform where they are running TensorRT. So TensorRT is more of optimize the runtime -- inference runtime for NVIDIA GPU.

Debraj Sinha

Got it. So this question is on how can you run -- like what is the minimum hardware/software setup that you need to run TAO in an in-house setup?

Chintan Shah

Minimum setup, Varun, do you want to take that one?

Varun Praveen

Yes, I can. So minimum setup for us would be any -- like in terms of hardware, we want any 8 core CPU with at least a single NVIDIA GPU, and that GPU should be a Pascal generation GPU or greater. We don't support the older generations with MACs, et cetera. So that's your minimum hardware requirement.

As per -- and software requirements, we need -- the biggest requirement that we need is, depending on the GPU that you're using, if you use a GeForce or an RTX machines, we would need to have at least in -- you need to have a 520 and above. For DGX machines or like our Tesla GPUs, you can go down to 470. TAO supports could have our compatibility so you should be able to run the training on GPUs with 470 drivers. And that's again only for Tesla GPUs.

Debraj Sinha

Got it. Thanks, Varun. And other questions we have for -- on deploying on Jetson. So if the model is going to run on the Jetson Xavier, do I need to apply for any different configuration to optimize for that hardware?

Chintan Shah

Direct question for me and Varun?

Varun Praveen

I'm sorry, can you repeat that again, Debraj you cut off in between?

Debraj Sinha

Sorry. If the model is running on the Jetson Xavier, do I need to make any like additional configuration changes to optimize for that hardware?

Varun Praveen

No, you don't -- not in terms of the training side, you can -- almost all of our models are deployable on Jetson Xavier so you should be able to run the training model. And once you have a trained model run, model export, which will generate a .eprt file that you can capture and generate that an RT engine for. So in terms of configuring a training time, there's no specific configurations that we limit to Jetson or any other deployment platform.

Debraj Sinha

Got it. And this question is to you, Varun. Can I have multiple validation set, save the best model on every validation set, the best model overall and get some validation metrics?

Varun Praveen

At the moment, as part of the training pipeline evaluate, we don't support multiple validation datasets and choosing fine-grain models based on the MAPs of different datasets. We allow you to combine and run it as a single validation dataset, but the fine-grain validation and saving per validation dataset is not supported.

Debraj Sinha

Got it. And this question to you, Chintan. Do we have any tools to expand the model with new data?

Chintan Shah

Can you repeat that question?

Debraj Sinha

Do we have any tools to expand the model with new data?

Chintan Shah

Do we have any tools to expand the model with new data. Sorry, I don't know if I understand the question.

Debraj Sinha

I feel like this is more in combined with the pre-trained models and using your own data to build your customized demos.

Chintan Shah

Yes, with TAO, I mean that's one of the things that we provide. You can use your own data and customize the model to your own dataset. And you can use one of our -- opt-in of our pre-trained model, the use case-specific pre-trained model to transfer learn on or you can just use a general-purpose model architecture and then you transfer learn on that and that will be customizable to your own data.

Debraj Sinha

Got it. So I'll take a couple of questions more. This question is to you, Praveen. Can I customize the training loop like Keras callbacks, for example?

Varun Praveen

So we -- any -- all the customizations that we provide are available as considerable parameters in TAO. So in terms of Keras -- in terms of call backs, for example, you can choose to use different learning rate to schedule a callback, so different optimizers or even the exponential roaming average callback, et cetera. We provide those features -- different kind of call backs available to you. But if you want to customize and bring your own callback that currently is not supported.

Debraj Sinha

Got it. And this question is for TensorRT. Like does TensorRT support stream data type as input? Do you know it -- do you know about it, Varun?

Varun Praveen

From what I understand yes, we do have some of the -- some stream data types allowed as input, but I'd have to get back on the details.

Debraj Sinha

Got it. And this is to you, again, Varun. Like, will newer versions of YOLO be available in near future for TAO?

Chintan Shah

I can take that. I can take that. Yes, I can take that. So one of the things I mean, we looked at adding YOLO models, especially the YOLO5 and V7. Unfortunately, we are not able to add due to licensing reasons, which is one of the reasons why we have not added. But we are adding in our upcoming release and -- in 5.0 and beyond, we are adding lots of newer transformer models -- capability on transform model from NVIDIA Research. I think those will have comparable accuracy and performance to a lot of the -- even better after the accuracy than a lot of the YOLO model. But yes, it's something that we looked at it, but unfortunately, we're not able to add for various licensing reasons.

Debraj Sinha

Got it. So that's all the time we have for Q&A. Thank you, Varun. Thank you, Chintan. And thank you all of you for joining us for the webinar. And I just like to remind you that we have the GTC Conference coming up on March 21 to 23. We have some exciting sessions on AI. We also have this session on NVIDIA TAO, where we're going to announce some new game-changing features for our next version 5.0. So stay tuned, join us for that session.

So again, thank you so much for joining us, and on-demand version for the webcast will be available very soon. Have a great day. Thank you. Thank you. Bye.

Chintan Shah

Bye. Thank you, guys. Thank you, Debraj.