GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

Project Report

What is new in Tensorflow and Keras

Yuvraj Singh

MatrNr: 21621819

Supervisor: Prof. Dr. Julian Kunkel
Ali Doost Hosseini
Jonathan Decker

Georg-August-Universität Göttingen
Institute of Computer Science

September 29, 2024

Abstract

This seminar report discussed and practically presents advancements within the TensorFlow (TF) and Keras ecosystem, focusing mainly on functionality, performance improvements, and hardware compatibility. Highlights of this report are KerasComputer Vision (CV) and KerasNatural language processing (NLP), as they attempt to provide all required tools for computer-vision and NLP tasks, respectively. This report also includes practical tasks performed using tools from KerasCV and KerasNLP Application Programming Interface (API)s. Whether it is using state-of-the-art object detection models like You Only Look Once (YOLO) or Large Language Models (LLM)s like Generative Pre-trained Transformer (GPT)-2(small) or BARD, Keras makes it possible just by a few lines of code. In the end, this report discusses DTensor and its basic concepts for scalable and distributed computing using Single Program, Multiple Data (SPMD) parallel computing paradigm under the hood. DTensor provides one global development environment while managing the distribution of tensors among devices internally. The combination of these tools showcases the growing potential of TF and Keras in both research and industry applications.

Contents

List of Tables

List of Figures

Listings

List of Abbreviations

HPC High Performance Computing

GRO Göttingen Research Online

ML Machine Learning

DL Deep Learning

AI Artificial Intelligence

TF TensorFlow

TFX TensorFlow Extended

JAX Just After eXecution

CNN Convolutional Neural Networks

RNN Recurrent Neural Networks

NN Neural Network

API Application Programming Interface

OS Operating Systems

CPU Central Processing Unit

GPU Graphics Processing Unit

TPU Tensor Processing Unit

CUDA Compute Unified Device Architecture

YOLO You Only Look Once

AWS Amazon Web Services

CV Computer Vision

FPS Frames Per Second

NLP Natural language processing

GPT Generative Pre-Trained Transformers

LLM Large Language Models

GAN Generative Adversarial Networks

GPT Generative Pre-trained Transformer

XLA Accelerated Linear Algebra

RGB Red Green & Blue

CSRT Channel and Spatial Reliability Tracker

GB Gigabyte

SPMD Single Program, Multiple Data

MPS Metal Performance Shaders

1 Introduction to TensorFlow & Keras

1.1 TensorFlow

The TF has emerged as one of the most preferred frameworks for Machine Learning (ML) & Deep Learning (DL). TF is a flexible and comprehensive framework that has enabled students, researchers, engineers, scientists, analysts, and developers to try their hands on ML & DL. TF has made it easy to create ML & DL applications.

1.1.1 Some features and developments

TF was an initiative started by the Google Brains team in the year 2015. Its first version was released in February 2017 as TF 1.0. It has seen significant changes and additions since then. Including the integration of Keras, which is high-level API for building models, importing datasets & other libraries with convenience. TF 2.0 was released with major API changes. It was TF 2.0, where model making was made more simpler and user-friendly while maintaining performance. TF can be used using various programming languages like Python, C++ & JavaScript. But, Python is the programming language that is used most in TF framework, one of the reasons for that is Python's short and easy-to-understand syntax. The ML & DL models can be developed and deployed across a range of platforms in TF, meaning that a model or application can be made or deployed on Linux, macOS, and Microsoft Windows Operating Systems (OS). TF can also run on different types of hardware like Central Processing Unit (CPU), GPU, & Tensor Processing Unit (TPU). [Goo24]

TF Core: TF includes core API for building, training, and deploying ML models. TF Core offers to support both high and low-level operations, i.e. Keras(high-level).

TensorFlow Extended (TFX): In March of 2018, TFX was released to enable end-to-end ML pipelines. It provides all the necessary tools and libraries to manage workflows, model training, monitoring, and validation.

TF Lite: TF Lite, which is for edge or computationally resources-limited devices like Raspberry Pi. TF Lite models are lightweight making them suitable for deployment on edge devices.

TensorFlow.js: is a library for ML in JavaScript. To develop ML models in JavaScript, and to use ML directly in the browser or Node.js, etc. TF documentation and tutorials show how to use TensorFlow.js with end-to-end examples.

TF data: offers a collection of datasets that are ready to use. All these datasets are exposed as TF.js. TF datasets enable high performance and easy-to-use input pipeline. This makes development and experimentation with models faster and more convenient.

TF Hub: is component of TF ecosystem that allows to use trained ML models, with possibility to publish models. This promoted re-usability. Therefore, there is no need to train models from scratch every time.

There are more components in TF ecosystem like TF Serving, TF Model Garden, TF Probability, TF Cloud, TF Quantum, etc.

The version of TF was 2.16.1 during this project. To add, Keras 3 will be the default version of Keras from this version of TF, and onwards.

1.2 Keras

Keras is a high-level Neural Network (NN) API which is written in Python. It is designed in a way that can run on top of frameworks like TF, PyTorch, JAX. However, Keras is more integrated with TF ecosystem, which is now its default backend. Keras makes it possible to do fast experimentation with a simple and user-friendly interface. It is modular and extensible, which makes it easier to customize models and plug in and out different types of NN layers, and optimizers. This kind of flexibility makes Keras a valuable tool for both research and production-focused environments. Keras supports a wide range of NN architectures, which includes Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), transformers, etc. [ONE]

1.3 JAX & JAX2TF

JAX is a high-performance library from Google for numerical computation. It is specially designed to leverage the power of special hardware like GPU and TPU if available to provide faster computations. JAX is good for research and prototyping but lacks built-in tools for deployment.

Here comes JAX2TF, which is a lightweight API to fill the gap between TF and JAX. This enables JAX to use tools from TF ecosystem. As a result, it is also possible to build and train models with JAX and deploy on edge devices like Raspberry Pi using TF Lite from TF ecosystem. Furthermore, JAX2TF allows fine-tuning of models trained with JAX in TF.

This combination enables model experimentation with JAX and followed by TF for smooth deployment.

1.4 Comparison: TensorFlow vs. PyTorch

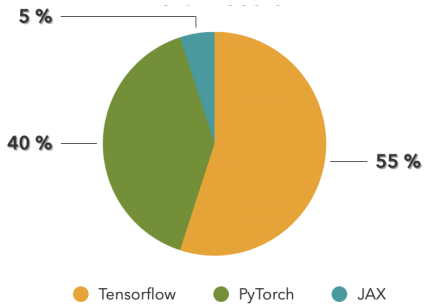

Figure 1: StackOverflow: Market share survey 2023

Both TF and PyTorch are renowned and powerful ML & DL frameworks. Both of these frameworks support GPU & TPU acceleration. Both consist of large and active

communities. Another highlight of both these frameworks is flexible APIs for ML&DL tasks.

Parameter	TensorFlow	PyTorch
Ecosystem	• More comprehensive ecosystem including TF Hub, TF Graphics, and Keras.	• Less comprehensive compared to TF, but a growing ecosystem.
Production Readiness	• TFX provides support for scalability, deployment, & management. • Offers a range of tools for production.	• Less focus on production & deployment. • PyTorch is gaining with tools like TorchServe but lags behind TensorFlow.
Prototyping and Research	• Slightly more complex to set up. • Suited for large-scale research and deployment.	• Favored by researchers due to its ease of use. • Faster for rapid prototyping and experimentation.
Training Time	• Efficient training may require more setup for custom experiments.	• Requires less training time, making it suitable for quicker experiments.
Customization	• Offers more tools for customization, especially through Keras and TF's APIs.	• TensorFlow offers more options for customization.

Table 1: Comparison between TensorFlow and PyTorch framwork [Vih24][Alv24]

2 TF's Hardware Compatibility

2.1 Apple Silicon:

TF 2.13 first provided Apple Silicon wheels. This TF native support for Apple Silicon chips enabled optimized performance on Apple's ML compute framework. This framework also includes automatic hardware switching between CPU and GPU, to be specific, Metal

Performance Shaders (MPS) in the case of Apple GPUs. Such features ensure efficient resource utilization of Apple's hardware for ML tasks.[TT23a][23]

2.2 NVIDIA GPUs:

TF now provides integrated NVIDIA Compute Unified Device Architecture (CUDA) libraries on Linux systems, this feature is available from TF 2.15. TF 2.15 also included an update to Clang 17.0.1 and CUDA 12.2 to boost NVIDIA's GPU performance. Furthermore, Clang 17 will be the default C++ compiler for TF. [TT23b]

2.3 Cross-Platform Compatibility:

TF is designed to support a wide range of hardware, which includes CPU, GPU, & TPU. TF also supports deployment on cloud platforms like Google, Amazon Web Services (AWS), Microsoft Azure, allowing flexible deployment on cloud-bases environments. For mobile and resource-constrained devices, TF provides TF Lite, which includes tools to convert TF models or weights to run on edge devices.[Boe23]

3 KerasCV

KerasCV is an extension of the Keras framework which is specifically designed for CV tasks. It provides a range of models, which includes state-of-art models like YOLO, and tools to develop CV applications. KerasCV provides all the required methods for image augmentation, image processing, model training, etc. KerasCV provides a wide range of pre-trained models for object detection, classification, segmentation, and even image generation using the Diffusion Model. This framework is built with the motive to provide an easy-to-use and highly customizable interface. KerasCV is well-documented, with relevant and well-explained examples, these documentations are actively maintained.[Ker24b]

3.1 What is new in KerasCV

3.1.1 Advanced Data Augmentation:

KerasCV offers API for using complex data augmentation layers with just a few lines of code. Examples of these layers could be RandAugment, CutMix, MixUp, and the list goes on. Using these layers effort for augmentation is decreased, following the contribution to improve the model's accuracy and robustness.

As mentioned, KerasCV provides augmentation layers like AutoAugment which automatically searches for the best augmentations, which helps the model to generalize well. [Ker22a]

3.1.2 Object Detection & Image Classification:

KerasCV provides variety of state-of-art models which includes YOLO object detection models with variety of weights like YOLOs(small) for edge devices. YOLOm(medium) and YOLOl(large) for more complex object detection tasks with the help of more parameters.[Ker23]

KerasCV includes state-of-art models for object classification like for example pre-trained EfficientNetV2B0 backbone.[luk23]

KerasCV also gives options to users to use these pre-trained models and further fine-tune them according to their specific needs. These models can be adapted by custom datasets, which saves a lot of time, effort, and computation resources.

3.1.3 High-performance image generation using Stable Diffusion in KerasCV:

KerasCV now offers to use the Stable Diffusion model with just a few lines of code. Stable Diffusion is a powerful open-source text-to-image generator. There are uncountable open-source implementations to create images from text, but KerasCV offers components like Accelerated Linear Algebra (XLA) compilation and mixed precision support, which help to achieve state-of-art generation speed. XLA is an open-source compiler for ML, it takes model from frameworks like PyTorch, TF and, JAX, and optimize their performance across different hardware platforms like GPUs. Whereas, Mixed Precision belongs to TF, which uses both 16- and 32-bit floating point during model training making it use less memory and run faster. Using Keras mixed precision API performance can increase by 3 times on modern GPUs, about 60% on TPUs, and more than 2 times on CPUs. [Ker22b][Ker24c][Ten24b]

3.2 KerasCV-Practical Task-I

This practical task demonstrates the practical application with KerasCV for object detection like drone flying in complex settings. By using tools from KerasCV model can adapt to tackle with situations like rapid camera or drone movement, low light, reflection, low resolution, complex background, etc.

3.2.1 Fine-tuning pre-trained YOLOv8 model for DJI 300 RTK drone

In this task, tools from KerasCV were used. The main components were the data augmentation layer and pre-trained object detection layer, from which the YOLOm v8 model was used.

3.2.2 Task Outline

This task focuses on fine-tuning of pre-trained YOLOm v8 model on a high-quality custom DJI 300 RTK drone dataset. With an emphasis on the impact of advanced data augmentation on model performance.

The outline of the task is following:-

1. Advanced data augmentation.
2. MODEL 1: Fine-tuning the pre-trained YOLOv8 model **without augmentation**.
3. MODEL 2: Fine-tuning the pre-trained YOLOv8 model **with augmentation**.
4. Evaluation of Trained Models.
5. Deployment of Model.
6. Challenges and possible improvements.

3.2.3 Dataset Preparation

The dataset for this task consisted of images of a DJI 300 RTK drone. The photos of the drone were taken both indoors and outdoors, in daylight and evening, at rest and while in the air, with and without natural occlusions, in reflection and under full light, to make the model as robust as possible to handle complex test situations. About 500-600 high-quality photographs of the drone were taken in the setting described. The cameras used were of iPhone 12, Raspberry Pi 12 MP HQ Camera, and Sony Alpha 7 III to get one of the best quality datasets with available resources. This dataset was labeled using *MAKE SENSE* image annotator[Sen].

3.2.4 Data augmentation

In the task, data augmentation was used to enhance the diversity in the dataset. Although image collection was taken in all possible settings, it is generally not possible to capture all the settings. There comes data augmentation, which helps the model to generalize by simulating different real-world situations using KerasCV data augmentation API. Various data augmentation were applied to datasets like basic random rotations with appropriate angles and horizontal flips. The advanced data augmentations included MixUp, Motion-Blur, Varying Red Green & Blue (RGB), RandomGridMask, RandomChannelShift, and RandomHue.

Figure 2: Random samples of augmentations performed

MixUp: It happens with the blending of two images, making the model generalize by learning from combined features.

MotionBlur: Simulates real-world motion or movement, making the model more robust to moving and blurred objects. This augmentation will help to detect the drone in a dynamic environment, or with rapid movements.

VaryingRGB: simulates different changes in environment light & color. This will give the model the ability to handle different lighting conditions in real-world scenarios.

RandomGridMask: In this augmentation part of the image is masked on random areas with random size. This model can detect objects with occlusion or missing features of the target object.

RandomChannelShift: Random color shifts, simulates environmental and camera sensor differences. This makes the detection model more resilient to varied images in terms of color consistency.

RandomHue: Adjusting Hue can simulate lighting and color conditions, it also provides the model the ability to generalize across different lighting and color shifts.

These augmentations collectively can improve the model's ability to handle a range of real-world conditions by simulating those conditions that were not in the original dataset, also increasing the size of the dataset at the same time. To note, these augmentation layers ask for arguments like upper and lower limits to rotate the image, or range of each RGB channel to perform augmentations.

3.2.5 Model Training

	Model-1	**Model-2**
GPU Used	NVIDIA A100	NVIDIA A100
Model Used	YOLOv8 m	YOLOv8 m
Dataset size	671 images	3.177 images
Background images	50 images	185 images
Training image size	1080 px	1080 px
Training time	26 mins	125 mins

Table 2: Training settings for Model-1 & Model-2

For training both models, YOLOm v8 backbone was used.

Model-1 is a basic model which was trained with 671 original images only, without any data augmentation. It consists of just 50 background images(the absence of a drone in the frame). This model was just trained in 26 minutes using NVIDIA A100 GPU from Google-Colab. This model was trained with an image size of 1080 pixels. This model was trained as a base model to compare Model-2 with it, to see the impact that data augmentation using KerasCV API makes.

Model-2 involved a more extensive image dataset, which included basic and advanced augmentations as mentioned above. This dataset consisted of 185 background images, making the total size of the dataset to 3.177 images. GPU used, image size, and backbone were the same as for Model-1. The training time taken was about 125 minutes.

3.2.6 Model Evaluation

The fine-tuned YOLOv8 models were evaluated using three different test videos acquired from the web. The objective was to measure the performance of both models with varied levels of difficulty, including resolution, lighting, flying distance, etc. in each of the test cases.

	Description
Test 1	Video recorded in a similar setting as training dataset
Test 2	Completely unseen, lower resolution, & distant flying
Test 3	Unseen, lower resolution, distant flying & complex background

Table 3: Description of tests performed

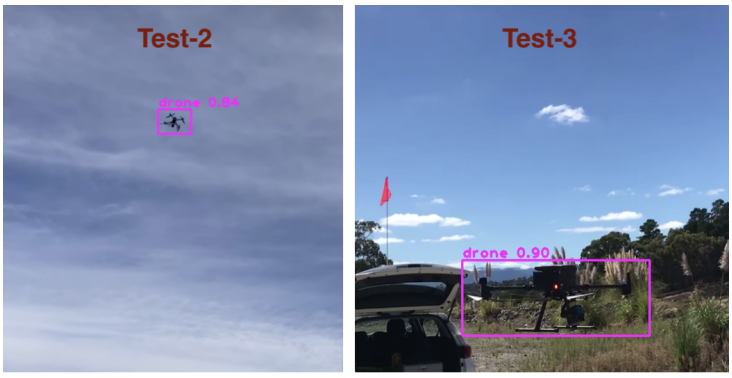

Figure 3: Shots from Test 2 & 3

Test-1: Is a video recorded in the same scenario as when the original training dataset was recorded. This test allowed how well the model performs on the resembling the training conditions.

Test-2: This is a completely unseen video with lower resolution and a DJI 300 RTK drone flying at a significant distance. This test pushed the model to perform on unseen data, and how it tacked challenging frames.

Test-3: Is also a completely unseen video with lower resolution, complex background, lighting, and DJI 300RTK drone flying at a significant distance. This test also pushed the model to perform on unseen data, and how it tacked challenging frames.

In Figure 3, example shots from test videos 2 & 3 can be seen.

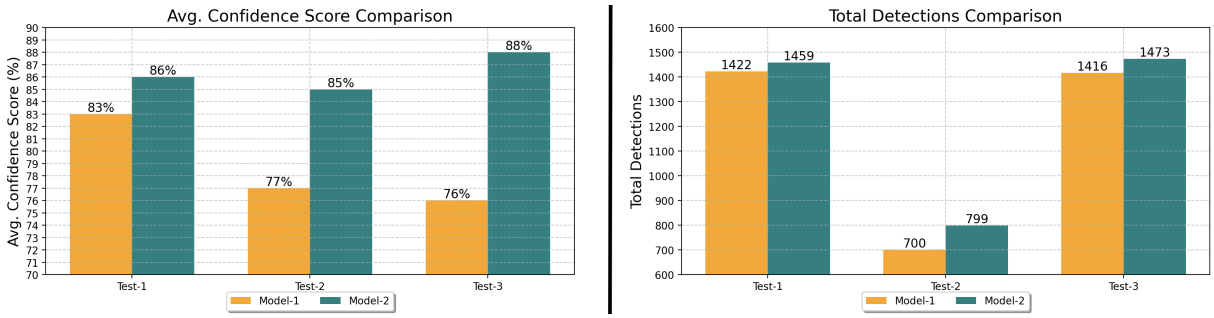

Figure 4: Model 1 & 2 performance comparison

By comparing the performance of both models with the help of the Average Confidence Score(%) and the number of total detections made by the models throughout each video.

Average Confidence Score(%) graph shows that both Model-1 and Model-2 performed equally well on Test-1. On the contrary, Model-2 performed significantly well compared to Model-1 in Test-2 and Test-3. This supports that augmentation enables the model to make detection with higher confidence.

Number of total detections graph shows that in all the test cases Model-2 detects DJI 300 RTK drone more than Model-1. Again this supports that augmentation improves model performance. To avoid any kind of bias, special care of Frames Per Second (FPS) and test video resolution was kept the same.

3.2.7 Challenges

Sometimes, even well-trained models can lose the track of target object. In that case, we can use Kalman filters, or Channel and Spatial Reliability Tracker (CSRT) trackers

provided by Python's computer vision library. The CSRT can hold the pixels from the last bounding box provided by the object detection model until the model can detect the target object again.

Model deployment is another crucial aspect of the ML/DL pipelines. TF and Keras provide a range of tools for smooth and effective deployment. An example of this could be converting this trained YOLOm v8 model as TFlite model, and to be deployed on resources-constrained devices like Raspberry Pi. But, embedded devices like Jetson Orin can be used for better inference, and speed in combination with high-resolution cameras with appropriate bandwidth to get the best out of the trained models.

3.3 KerasCV-Practical Task-II

3.3.1 High-performance image generation using Stable Diffusion

Figure 5: Astronaut riding horse by Stable Diffusion[Ker22b]

KerasCV provides support for Stable Diffusion enabling users to produce high-resolution, realistic images with XLA compilation and mixed precision support, which together achieve state-of-the-art generation speed. Stable Diffusion is a powerful deep-learning model for high-quality and realistic image generation. In just a few lines of code and description as an argument what to create, enabling possibilities in creative applications like designs, fantasy art, and entertainment. It is free to use, without any need for expensive proprietary software.[Ker22b]

3.3.2 Image generation using Stable Diffusion on Mac M1 Pro Chip

```
images = model.text_to_image(
    "A detailed airship floating above a steampunk city at sunset. The
    airship has a spherical, metal structure with Victorian technology.
    The city below has tall, gothic buildings with metalwork. The
    atmosphere is misty with warm, golden sunlight.",
    batch_size=2
)
def plot_images(images):
    plt.figure(figsize=(20, 20))
    for i in range(len(images)):
        ax = plt.subplot(1, len(images), i + 1)
        plt.imshow(images[i])
        plt.axis("off")
plot_images(images)
```

Listing 1: Example Prompt for generating image

It takes about 5 minutes to generate a single image using Stable diffusion on an Apple MacBook Pro with M1 Chip(10 Core CPU & 16 Core GPU).

Figure 6: Image generation using Stable Diffusion on Mac M1 Pro

3.3.3 Limitations of Stable Diffusion

Stable diffusion typically requires highly specified hardware, 10 to 30 Gigabyte (GB)s of VRAM on a GPU to generate images quickly and efficiently. The maximum image size can be 1024x1024 pixels, whereas the default size is 512x512 pixels.

Figure 7: Example of generating human figures using Stable Diffusion

Stable Diffusion has language and cultural biases to the English language and Western culture. Furthermore, Stable Diffusion is not the best to generate faces and limbs, especially human beings. This is evident in Figure 7.

4 What is KerasNLP

KerasNLP is a special extension of Keras DL framework designed to perform various NLP tasks with ease. Some of the highlights of KerasNLP are following:-

Text Preprocessing: KerasNLP contains tools for essential NLP tasks such as tokenization and text processing.

State-of-the-Art-Models: KerasNLP offers a range of pre-trained models for text generation, classification, translators, and more. KerasNLP provides many pre-trained models, such as Google Bard and GPT-2.

Easy integration: KerasNLP simplifies the process of integrating NLP functions into projects or applications, whether it is a custom solution or a pre-trained model.

4.1 KerasNLP-Practical Task

This practical task shows how KerasNLP can be used to generate text and further fine-tune a text generation model. This text generation can be achieved by using LLMs. LLMs are types of models that are trained on very large corpus text to generate output for NLP tasks, such as text generation, machine translation, etc. Generative LLMs are based on DL NN, such as Transformers which were invented by Google in 2007. [Ker]

4.1.1 Text Generation using GPT-2

This part shows GPT-2 model can used from KerasNLP API to generate human-like texts. It is fully integrated within the TF ecosystem.

```
preprocessor = keras_nlp.models.GPT2CausalLMPreprocessor.from_preset("
    gpt2_base_en",sequence_length=128,)
gpt2_lm = keras_nlp.models.GPT2CausalLM.from_preset("gpt2_base_en",
    preprocessor=preprocessor)
```

Listing 2: loading & initializing the GPT-2

GPT-2 model can be loaded and initialized using KerasNLP API as shown in Listing-2.

```
output = gpt2_lm.generate("Autonomous vehicles (AV)", max_length=200)
print(output)
print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")
```

Listing 3: Example prompt to generate text

After loading the model, text can be generated by simply calling method *generate()* from *gpt2_lm*. For example, the prompt is passed as *"Autonomous vehicles (AV)"* shown in Listing-3.

```
Autonomous vehicles (AV) and unmanned vehicles (UAVs) could become a
    major drivers of transportation in the future, said an industry group
    .
```

Listing 4: Output

Output can be seen after passing the prompt as shown in Listing-4. This output is based on information on which GPT-2 the model was trained.

4.1.2 Fine-tuning

KerasNLP makes it possible to fine-tune or train the model on the text of the user's choice. Fine-tuning allows GPT-2 model to adapt knowledge from provided text, enhancing its ability to generate outputs specific to the domain of interest. To demonstrate this process pre-trained GPT-2 model was used. [Ker24a]

In this practical task, GPT-2 model was trained with setting as shown in Table 4. The model was fine-tuned using a university seminar report as a custom dataset written on *Ethical Issues with Autonomous Vehicles*.

```
output = gpt2_lm.generate("Autonomous vehicles (AV)", max_length=200)
print(output)
print(f"TOTAL TIME ELAPSED: {end - start:.2f}s")
```

Listing 5: Example prompt to generate text after fine-tunning

GPT Backbone	gpt2_base_en
Description of Text Used	Ethical issues with Autonomous Vehicles
Trained for Epochs	10
Training Time	7.5 Minutes

Table 4: GPT-2 Fine-tuning details

```
Autonomous vehicles (AV) are autonomous vehicles designed to operate
   safely. Autonomous vehicles (AV) represent a remarkable technological
    development in road transportation. Autonomous vehicles (AV) are
   highly dependent on data coming from the sensors, AI, and software
   making decisions.
```

Listing 6: Fine-tuned Output

From the output in Listing-6, it is evident that the response this time was based on knowledge from paper *Ethical Issues with Autonomous Vehicles*.

5 DTensor

DTensor is a global programming model that allows developers to design an application that operates on Tensors globally while DTensor manages the distribution across devices automatically. DTensor scales up models using a procedure called SPMD expansion. SPMD is a parallel computing model where multiple processors collaborate in the execution of a program to solve computations faster. And DTensor uses SPMD to distribute programs and tensors according to sharding directives. This decoupling allows running the same application on a single device, multiple devices, or even on multiple clients while preserving global semantics internally. DTensor can be used on devices like CPUs, GPUs, or even TPUs, including virtual or simulated ones. [Ten24a][Ten23]

5.1 DTensor's Model of Distributed Tensors

DTensor has mainly two concepts, namely **Mesh** and **Layout**. Mesh defines the list of devices on which computations would be carried out. And Layout defines how to shard the Tensor's dimension on a Mesh.[Ten24a]

5.1.1 Mesh

Mesh is a concept that defines the group of devices, such as CPUs, GPUs, or even TPUs, in other words, it represents available hardware resources that are responsible for the distributed computations. Mesh has some conventions and features as follows:-

1. Logical Grid: Organizes devices into a grid with named dimensions, each dimension is called *Mesh dimension*.

2. Unique Names: Each of the dimensions in the same Mesh can not have the same name, must be unique by rule.

3. Reference by Layout: Names of Mesh are used by Layout to describe tensor's division.

4. Multi-Dimensional Array: Mesh can be laid out as a multi-dimensional array, where each element is a device.

Mesh Examples

```
configure_virtual_cpus(6)
DEVICES = [ f CPU :{i}    for i in range(6)]
```

Listing 7: Hardware: 6 virtual CPUs

```
mesh_1d = dtensor.create_mesh([('x', 6)], devices=DEVICES)
```

Listing 8: Example 1D Mesh: 6 devices along a mesh dimension 'x'

In Listing-7, six virtual CPUs have been created to configure them into a mesh. Following that in Listing-8, the mesh has been created, which expands across dimension 'x'. as shown in Figure-8.

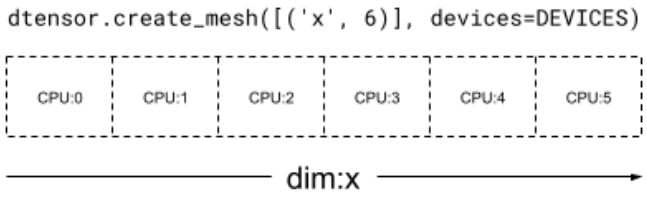

Figure 8: 1D Mesh expanding along 'x' dimension[Ten24a]

```
mesh_2d = dtensor.create_mesh([('x', 3), ('y', 2)], devices=DEVICES)
```

Listing 9: Example 2D Mesh

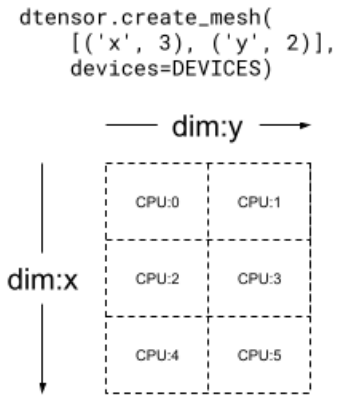

Figure 9: Example 2D Mesh[Ten24a]

In Listing-9, an example 2D mesh is been created, visual representation can be seen in Figure-9.

5.1.2 Layout

Layout describes how tensor or data will be distributed across the defined Mesh. The Layout ensures that tensors are distributed to appropriate devices, to enable parallel computing.

The layout has some conventions and features as follows:-

1. Terms: The terms *Axis and Rank* are used in the Layout context. Whereas, the term *Dimension* is used in the context of the Mesh.

2. Rank: Is the number of axis of the Layout, which must match the rank of the tensor.

3. Sharding: Each axis of the Tensor can be shared across the Mesh dimension, tensor also could remain *"unsharded"*.

4. Matching Dimensions and Axes: Number of Layout axes does not need to match the number of Mesh dimensions.

Layout Examples

Continuing with Listing-8, Listing-10 shows how a tensor's second axis can be shard across 6 devices or along the x-dimension of 1D Mesh. The visual representation can be seen in Figure 10.

```
layout = dtensor.Layout([dtensor.UNSHARDED, 'x'], mesh_1d)
```

Listing 10: Shard the second axis of tensor across 6 devices

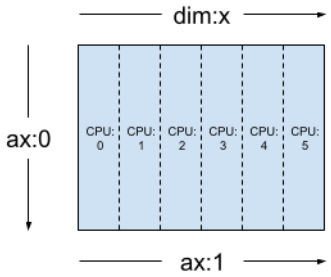

Figure 10: Shard the second axis of tensor across 6 devices[Ten24a]

```
layout = dtensor.Layout(['y', 'x'], mesh_2d)
```

Listing 11: First axis sharded across 'y', second axis across 'x' of rank-2 Tensor

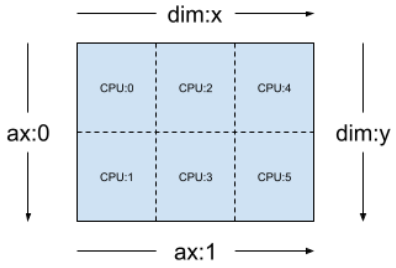

Figure 11: 1st axis sharded across 'y', second axis across 'x' of rank-2 Tensor[Ten24a]

Now continuing with Listing-9, Listing-11 shows how a tensor's first axis can be sharded across the "y"-dimension, and the second axis across the "x"-dimension of the rank-2 Tensor. The visual representation can be seen in Figure 11.

As models are getting bigger and bigger, fitting them into one device is getting harder. Therefore, DTensor is one of the options to fit these models and training environments on multiple devices while getting unified global semantics. [Ten23]

6 Conclusion

In this paper, the latest features of TensorFlow and Keras, and the growing ecosystem were explored. TF and Keras ecosystem are consistently evolving, including advancement in hardware compatibility and efficiency. TF support for Apple Silicon and NVIDIA GPUs is making it possible to work on a wide range of devices with ease, improving both power efficiency and training times.

KerasCV and KerasNLP provide dedicated tools for computer vision and natural language processing applications. In KerasCV projects 1 and 2, it was observed that it is convenient to use state-of-the-art models with just a few lines of Python code, and following that fine-tuning those models on custom datasets with little more effort. As data augmentation was part of this, it was also evident that Keras also provides augmentation layers to handle these tasks effectively. Last but not least KerasCV also makes it possible to use Stable Diffusion with just a few lines of code followed by a prompt to generate free Artificial Intelligence (AI) generated images.

With KerasNLP it is possible to use LLMs like GPT-2 and Bart with just a few lines of code, and it is even possible to fine-tune them on personal or choice of text corpus. KerasNLP provides all other required tools to perform NLP tasks requiring the least effort.

Last, the DTensor could be a major player when it comes to distributed training by efficiently enabling parallel computing, for example along multiple devices.

TF and Keras are continually growing in terms of performance and accessibility, by coming up with new tools that empower both research and industry applications.

References

[23] "TensorFlow 2.13 and Keras 2.13 Release Notes". In: (July 2023), p. 1. URL: https://www.exxactcorp.com/blog/deep-learning/tensorflow-2-13-and-keras-2-13-release-notes.

[Alv24] Farooq Alvi. "PyTorch vs TensorFlow in 2024: A Comparative Guide of AI Frameworks". In: (Jan. 2024), p. 1. URL: https://opencv.org/blog/pytorch-vs-tensorflow/.

[Boe23] Gaudenz Boesch. "TensorFlow Lite – Real-Time Computer Vision on Edge Devices (2024)". In: (Dec. 2023), p. 1. URL: https://viso.ai/edge-ai/tensorflow-lite/#:~:text=TensorFlow%20Lite%20(TFLite)%20is%20a,on%20mobile%20and%20edge%20devices..

[Goo24] Google. "An end-to-end platform for machine learning". In: (July 2024). URL: https://www.tensorflow.org.

[Ker] Keras-Team. "KerasNLP". In: (), p. 1. URL: https://keras.io/keras_nlp/.

[Ker22a] Keras-Team. "CutMix, MixUp, and RandAugment image augmentation with KerasCV". In: (2022), p. 1. URL: https://keras.io/guides/keras_cv/cut_mix_mix_up_and_rand_augment/.

[Ker22b] Keras-team. "High-performance image generation using Stable Diffusion in KerasCV". In: (2022), p. 1. URL: https://keras.io/guides/keras_cv/generate_images_with_stable_diffusion/.

[Ker23] Keras-Team. "Object Detection with KerasCV". In: (2023), p. 1. URL: https://keras.io/guides/keras_cv/object_detection_keras_cv/.

[Ker24a] Keras-Team. "GPT2 Text Generation with KerasNLP". In: (2024), p. 1. URL: https://keras.io/examples/generative/gpt2_text_generation_with_kerasnlp//.

[Ker24b] Keras-Team. "KerasCV". In: (2024), p. 1. URL: https://keras.io/keras_cv/.

[Ker24c] Keras-team. "XLA". In: (2024), p. 1. URL: https://openxla.org/xla.

[luk23] lukewood. "Classification with KerasCV". In: (2023), p. 1. URL: https://keras.io/guides/keras_cv/classification_with_keras_cv/.

[ONE] ONEIROS. "About Keras 3". In: (), p. 1. URL: https://keras.io/about/.

[Sen] Make Sense. "Make Sense". In: (), p. 1. URL: https://www.makesense.ai.

[Ten23] TensorFlow. "Google I/O 2023: What's new in TensorFlow and Keras?" In: (May 2023), p. 1. URL: https://blog.tensorflow.org/2023/05/google-io-2023-whats-new-in-tensorflow-and-keras.html.

[Ten24a] TensorFlow. "DTensor concepts". In: (2024), p. 1. URL: https://www.tensorflow.org/guide/dtensor_overview.

[Ten24b] Tensorflow. "Mixed precision". In: (2024), p. 1. URL: https://www.tensorflow.org/guide/mixed_precision.

[TT23a] TensorFlow and Keras Teams. "What's new in TensorFlow 2.13 and Keras 2.13?" In: (July 2023), p. 1. URL: https://blog.tensorflow.org/2023/07/whats-new-in-tensorflow-213-and-keras-213.html.

[TT23b] TensorFlow and Keras Teams. “What’s new in TensorFlow 2.15”. In: (Nov. 2023), p. 1. URL: https://blog.tensorflow.org/2023/11/whats-new-in-tensorflow-2-15.html.

[Vih24] rennan Whitfield Vihar Kurama Artem Oppermann. “PyTorch vs. TensorFlow: Key Differences to Know for Deep Learning”. In: (Mar. 2024), p. 1. URL: https://builtin.com/data-science/pytorch-vs-tensorflow#.