European IT Certification Curriculum
Self-Learning Preparatory Materials

EITC/AI/GCML

Google Cloud Machine Learning

This document constitutes European IT Certification curriculum self-learning preparatory material for the EITC/AI/GCML Google Cloud Machine Learning programme.

This self-learning preparatory material covers requirements of the corresponding EITC certification programme examination. It is intended to facilitate certification programme's participant learning and preparation towards the EITC/AI/GCML Google Cloud Machine Learning programme examination. The knowledge contained within the material is sufficient to pass the corresponding EITC certification examination in regard to relevant curriculum parts. The document specifies the knowledge and skills that participants of the EITC/AI/GCML Google Cloud Machine Learning certification programme should have in order to attain the corresponding EITC certificate.

Disclaimer

This document has been automatically generated and published based on the most recent updates of the EITC/AI/GCML Google Cloud Machine Learning certification programme curriculum as published on its relevant webpage, accessible at:

https://eitca.org/certification/eitc-ai-gcml-google-cloud-machine-learning/

As such, despite every effort to make it complete and corresponding with the current EITC curriculum it may contain inaccuracies and incomplete sections, subject to ongoing updates and corrections directly on the EITC webpage. No warranty is given by EITCI as a publisher in regard to completeness of the information contained within the document and neither shall EITCI be responsible or liable for any errors, omissions, inaccuracies, losses or damages whatsoever arising by virtue of such information or any instructions or advice contained within this publication. Changes in the document may be made by EITCI at its own discretion and at any time without notice, to maintain relevance of the self-learning material with the most current EITC curriculum. The self-learning preparatory material is provided by EITCI free of charge and does not constitute the paid certification service, the costs of which cover examination, certification and verification procedures, as well as related infrastructures.

TABLE OF CONTENTS

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: INTRODUCTION
TOPIC: WHAT IS MACHINE LEARNING

Machine learning is a powerful tool that allows us to derive meaning from the vast amount of data in the world today. It is not magic, but rather a set of tools and technologies that can be utilized to answer questions using data. In this Cloud AI Adventures series, we will explore the art, science, and tools of machine learning, and discover how easy it is to create amazing experiences and gain valuable insights.

The value of machine learning is just beginning to show itself. With the increasing volume of data generated by people, computers, phones, and other devices, traditional human analysis and manual rule-writing are no longer sufficient. We need automated systems that can learn from the data and adapt to a shifting landscape. Machine learning is already present in many products we use today, although it may not always be apparent. From tagging objects in photos to recommending the next material to watch, machine learning is behind it all. Google search is a prime example, with multiple machine learning systems at its core, from understanding your query to personalizing the search results based on your interests.

Machine learning has a wide range of applications, including image recognition, fraud detection, recommendation systems, and text and speech systems. These capabilities can be applied to fields such as healthcare, retail, and transportation. As machine learning becomes more prevalent, it is no longer considered a novelty but an expected feature in products. It has the potential to make human tasks better, faster, and easier, and even enable us to accomplish tasks that were previously impossible.

Taking advantage of machine learning is easier than ever. The tooling has improved significantly, and all you need is data, developers, and a willingness to explore. Machine learning can be defined as using data to answer questions. This definition can be split into two parts: using data for training and answering questions through predictions or inference. Training involves using data to create and fine-tune a predictive model, which can then be used to make predictions on new data. As more data is gathered, the model can be improved and new models can be deployed.

Data is the key to unlocking the potential of machine learning. It is the foundation on which the entire process relies. In future materials, we will delve deeper into the different techniques and approaches in machine learning, and provide the tools to help you accomplish your goals with your data.

Machine learning is a powerful tool that allows computers to learn from data and make predictions or decisions without being explicitly programmed. In this didactic material, we will delve into the concrete process of doing machine learning, providing a step-by-step formula for approaching machine learning problems.

The first step in the machine learning process is to define the problem and gather relevant data. This involves understanding the problem at hand and determining what kind of data is needed to solve it. Once the problem and data requirements are clear, the next step is to collect and preprocess the data. This may involve cleaning the data, handling missing values, and transforming the data into a suitable format for machine learning algorithms.

After the data is prepared, the next step is to choose an appropriate machine learning algorithm. There are various types of algorithms, such as regression, classification, clustering, and reinforcement learning, each suited for different types of problems. The choice of algorithm depends on the nature of the problem and the available data.

Once the algorithm is selected, the next step is to train the model. Training involves feeding the algorithm with labeled data, where the input features are paired with their corresponding target values. The algorithm learns from this labeled data to make predictions or decisions. The training process aims to find the best set of parameters or weights that minimize the error between the predicted and actual values.

After the model is trained, it needs to be evaluated to assess its performance. This is done using a separate set of data called the test set, which was not used during the training phase. Evaluation metrics such as accuracy, precision, recall, and F1 score are used to measure the model's performance. If the model's performance is satisfactory, it can be deployed and used for making predictions on new, unseen data.

It is important to note that machine learning is an iterative process. If the model's performance is not satisfactory, it may be necessary to revisit previous steps, such as collecting more data, choosing a different algorithm, or tuning the model's parameters. This iterative process continues until a satisfactory model is obtained.

Machine learning is a step-by-step process that involves defining the problem, gathering and preprocessing data, choosing an appropriate algorithm, training the model, evaluating its performance, and iterating if necessary. By following this formula, one can effectively approach machine learning problems and harness the power of artificial intelligence.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FIRST STEPS IN MACHINE LEARNING
TOPIC: THE 7 STEPS OF MACHINE LEARNING

Machine learning is a powerful technology that has enabled computer systems to perform tasks such as detecting skin cancer, sorting cucumbers, and identifying escalators in need of repair. But how does it work? In this didactic material, we will walk through the process of machine learning using a basic example of building a system that can determine whether a drink is wine or beer.

The first step in machine learning is to collect data to train our model. In our example, we will collect data on the color and alcohol content of glasses of wine and beer. These two factors, color and alcohol, will be our features. We will use a spectrometer to measure the color and a hydrometer to measure the alcohol content. Once we have gathered our data, we will have a table of color, alcohol content, and whether it's beer or wine. This will be our training data.

The next step is data preparation. We will load our data into a suitable place and prepare it for training. We will combine all our data and randomize the ordering to ensure that the order of the data does not affect our learning process. We will also visualize our data to identify any relevant relationships between variables and to check for data imbalances. Additionally, we will split our data into two parts - the majority will be used for training our model, and the rest will be used for evaluating the model's performance.

After data preparation, we move on to choosing a model. There are various models available for different types of data. In our case, with just two features, we can use a small linear model. This model is simple but effective for our task.

The final step is training the model. In this step, we will use our training data to improve the model's ability to predict whether a given drink is wine or beer. This process is similar to someone learning to drive - at first, they are unfamiliar with the controls, but with practice, they become more skilled. Similarly, our model will gradually improve its predictions as it learns from the training data.

It is important to note that the quality and quantity of the training data directly impact the accuracy of the model. Gathering a diverse and representative dataset is crucial for achieving good results.

The process of machine learning involves collecting and preparing data, choosing a suitable model, and training the model to make accurate predictions. By following these steps, we can create models that have the ability to answer questions and solve problems based on data.

Machine learning is a powerful tool that allows us to use data to answer questions and make predictions. In the context of Google Cloud Machine Learning, there are seven steps involved in the process of machine learning.

The first step is to gather data. In order to train a machine learning model, we need a dataset that contains examples of inputs and their corresponding outputs. In this case, we are using a dataset of drinks, with their color and alcohol percentage as inputs, and whether they are wine or beer as the output.

Once we have our dataset, the next step is to prepare the data. This involves cleaning the data, handling missing values, and transforming the data into a format that can be used by the machine learning model. In this case, we are using a simple formula for a straight line to represent our model.

The third step is to choose a model. In machine learning, a model is a mathematical representation of the relationship between the inputs and outputs in our dataset. In this case, we are using a straight line to separate wine from beer based on their color and alcohol percentage.

After choosing a model, the next step is to train the model. This involves initializing random values for the parameters of the model, and then adjusting those values based on the comparison between the model's predictions and the actual outputs in the dataset. This process is repeated multiple times, with each iteration called a training step.

Once the training is complete, the next step is evaluation. This involves testing the trained model on a separate

dataset that was not used for training. This allows us to assess how well the model will perform on new, unseen data. It is important to set aside a portion of the original dataset for evaluation purposes.

After evaluation, we may choose to further improve our model by tuning hyperparameters. Hyperparameters are parameters that are not learned from the data, but rather set by the user before training. Examples of hyperparameters include the number of training iterations and the learning rate. Finding the optimal values for these hyperparameters can improve the accuracy of the model.

Finally, once we are satisfied with our trained model and hyperparameters, we can use the model to make predictions or inferences. This is the step where we can answer questions based on the data we have. In this case, we can use our model to predict whether a given drink is wine or beer based on its color and alcohol percentage.

The seven steps of machine learning in the context of Google Cloud Machine Learning are: gathering data, preparing the data, choosing a model, training the model, evaluating the model, tuning hyperparameters, and using the model for prediction or inference.

Machine learning is a powerful tool in the field of artificial intelligence that allows computers to learn and make predictions without being explicitly programmed. In this didactic material, we will explore the seven steps of machine learning, which serve as a foundational framework for understanding the process.

The first step in machine learning is to gather and prepare the data. This involves collecting relevant data that will be used to train the model. The data should be representative of the problem we are trying to solve and should be properly formatted and cleaned to ensure accurate results.

Once the data is gathered, the next step is to select a model. There are various models available, each with its own strengths and weaknesses. The choice of model depends on the specific problem and the type of data we are working with.

After selecting a model, the next step is to train it. Training involves feeding the model with the prepared data and allowing it to learn from the patterns and relationships in the data. During this process, the model adjusts its internal parameters to minimize the error between its predictions and the actual values in the training data.

Once the model is trained, the next step is to evaluate its performance. This is done by testing the model on a separate set of data, called the test set, that was not used during the training phase. The evaluation metrics, such as accuracy or mean squared error, are used to assess how well the model generalizes to new, unseen data.

After evaluating the model, the next step is hyperparameter tuning. Hyperparameters are settings that are not learned by the model during training but need to be set manually. They control the behavior of the model and can greatly impact its performance. Hyperparameter tuning involves trying different combinations of values to find the optimal settings for the model.

Finally, the last step is prediction. Once the model is trained and evaluated, it can be used to make predictions on new, unseen data. This is the ultimate goal of machine learning - to use the learned patterns and relationships to make accurate predictions or classifications.

To further explore machine learning and experiment with different parameters, the TensorFlow Playground is a valuable resource. It is a browser-based machine learning sandbox that allows users to try different parameters and run training against mock datasets. This interactive tool provides a hands-on experience in understanding the impact of various parameters on the model's performance.

While this material provides a basic understanding of the seven steps of machine learning, there are more advanced concepts and nuances that will be covered in future materials. However, this foundational framework gives us a common language to think through the problem and serves as a starting point for further exploration.

In the next material, we will dive into building our first real machine learning model using code, moving beyond drawing lines and algebra. Stay tuned for an exciting hands-on experience in creating and training machine learning models.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FIRST STEPS IN MACHINE LEARNING
TOPIC: PLAIN AND SIMPLE ESTIMATORS

Machine learning is a powerful tool that allows us to build models and gain insights from data without requiring advanced mathematical knowledge. In this material, we will explore the process of training a simple classifier using Google's open-source machine learning library, TensorFlow.

To begin, we will import TensorFlow and numpy, and print out the version number of TensorFlow to confirm the version we are using. We will be building a model to classify different species of iris flowers based on their measurements. The dataset consists of measurements of the height and width of the flowers' petals and sepals.

We will load the dataset using TensorFlow's Load CSV with Header function. The data is presented as floating-point numbers, and the labels are recorded as integers corresponding to the species of flowers. We can access the data and labels using named attributes.

Next, we will set up the feature columns, which define the types of data coming into the model. In this case, we will use a four-dimensional feature column to represent our features. We will instantiate the model using tf.estimators.LinearClassifier, passing in the feature columns, the number of different outputs the model predicts (in this case, 3), and a directory to store the model's training progress and output files.

To connect our model to the training data, we need to create an input function. The input function generates the data for the model by mapping the raw data using the feature columns. We use the same name for the features as we did in defining the feature column to associate the data.

Now we are ready to train our model. We will use the classifier.train function, passing in the input function as an argument. This function handles the training loop and iterates over the dataset, improving the model's performance with each step. After completing the training steps, we can evaluate our model's performance using the classifier.evaluate function and our test dataset. We can extract the accuracy from the metrics returned.

In this material, we have learned how to train a simple classifier using TensorFlow's high-level API called estimators. This API simplifies the training process by providing a pre-defined training loop and allowing us to focus on the interesting parts of machine learning. We have successfully trained a model to classify different species of iris flowers with an accuracy of 96.6%.

In the process of working with machine learning, there are several key steps that need to be followed. These steps include obtaining the raw data, setting up the input function, defining the feature columns and model structure, running the training, and finally evaluating the results. By following this framework, we can focus on understanding the data and its properties, rather than getting bogged down in the underlying mathematical details.

In this material, we also explore a simplified version of TensorFlow's high-level API using a canned estimator. This approach provides a straightforward way to implement machine learning models without delving into complex programming or mathematical concepts. It is a great starting point for beginners in the field of artificial intelligence.

In subsequent materials, we will delve deeper into this topic and explore how to enhance our models with additional details, handle more complex datasets, and utilize more advanced features. This will allow us to create more sophisticated and accurate machine learning models.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FIRST STEPS IN MACHINE LEARNING
TOPIC: SERVERLESS PREDICTIONS AT SCALE

Once we have a trained machine learning model, we need to serve our predictions at scale. Google's Cloud Machine Learning Engine provides a solution for creating a prediction service for TensorFlow models without any operational work. This allows us to go from a trained model to a deployed auto-scaling prediction service in just a few minutes.

When serving predictions, it is ideal to deploy a purpose-built model that is fast and lightweight. We want a static model that doesn't require any updates while serving predictions. Additionally, if we want our prediction server to scale with demand, it adds another layer of complexity to the problem.

Fortunately, TensorFlow has a built-in function called "export_savedmodel" that can generate an optimized model for serving predictions. This function handles all the necessary adjustments, saving us a lot of work. We can run this function directly from our classifier object once we are satisfied with the trained model's state.

The "export_savedmodel" function creates a snapshot of our model and exports it as a set of files. These exported files consist of a file called "saved_model.pb" which defines the model structure, and a variables folder that contains the trained weights in our model. As our model improves over time, we can continue to produce exported models to provide multiple versions.

Once we have an exported model, we have two primary options for serving it in production. One option is to use TensorFlow serving, which is a part of TensorFlow and available on GitHub. The other option is to use Google Cloud Machine Learning Engine's prediction service. Both options have very similar file interfaces, but for this discussion, we will focus on Cloud Machine Learning Engine's prediction service.

Cloud Machine Learning Engine allows us to take the exported model and turn it into a prediction service with a built-in endpoint and auto-scaling capabilities. It provides a feature-rich command line tool, API, and UI, allowing us to interact with it in multiple ways based on our preferences.

To use Cloud Machine Learning Engine's prediction service, we can follow these steps:

1. Run the "export_savedmodel" function on our trained classifier to generate an exported model.
2. Upload the exported model files to Google Cloud storage, ensuring that the compute and storage are in the same region.
3. In the Cloud Machine Learning Engine UI, create a new model, which serves as a wrapper for our released versions.
4. Create a version by providing a name for the model version and pointing it to the Cloud storage directory that holds the exported files.

By following these steps, we can easily create a prediction service for our model. The service handles all the operational aspects of setting up and securing the endpoint, and it automatically scales based on demand. This elasticity means that we only pay for compute resources when they are needed, reducing costs when demand is low.

Google Cloud Machine Learning Engine's prediction service allows us to serve our machine learning predictions at scale without the need for extensive operational work. By leveraging the "export_savedmodel" function and the Cloud Machine Learning Engine's features, we can quickly go from a trained model to a deployed auto-scaling prediction service.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FIRST STEPS IN MACHINE LEARNING
TOPIC: TENSORBOARD FOR MODEL VISUALIZATION

In this material, we will explore how to use TensorBoard to visualize and debug machine learning models. TensorBoard is a powerful tool that allows us to track metrics and analyze the structure of our models, making it easier to identify and solve problems.

Machine learning can be unpredictable, and the success or failure of the training process depends on various factors such as data quality, model nuances, and parameter choices. TensorBoard helps us track these metrics and visualize the model structure, enabling us to fine-tune the model and debug any issues that arise.

TensorFlow, the underlying framework for TensorBoard, uses computational graphs to represent models. Instead of directly adding numbers, TensorFlow constructs graph operators and passes inputs to them. When training a model, TensorFlow executes operations within the graph. TensorBoard can visualize these models, allowing us to examine their structure and ensure everything is connected as intended.

TensorBoard offers various features for model visualization. We can zoom, pan, and expand elements to explore different layers of abstraction, reducing visual complexity. Additionally, TensorBoard can plot metrics such as accuracy, loss, and cross-entropy on graphs, providing insights into the model's performance. TensorFlow's pre-configured values make it easy to start visualizing these metrics.

Furthermore, TensorBoard supports other types of information, including histograms, distributions, embeddings, audio, pictures, and text data associated with the model. However, these features will be covered in future materials.

To demonstrate TensorBoard, let's consider an example with a linear model. After starting TensorBoard and specifying the directory where the model files are saved, we can observe scalar metrics such as loss. By expanding and zooming into the graphs, we can analyze the training progress. In our example, the loss consistently decreases, indicating ongoing improvement. This insight suggests that extending the training process might yield even better results.

Moving to the graphs tab, we can explore the model's structure. Initially, the graph appears simple, but we can expand each block to examine its components in detail. By naming graph components, we can enhance debugging and understand how the graph is connected. TensorFlow allows most operations to be named, facilitating model clarification.

TensorBoard is a valuable tool for visualizing machine learning models and analyzing metrics. It simplifies debugging and allows us to optimize the training process. By using TensorBoard, we gain a deeper understanding of our models and can make informed decisions to improve their performance.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FIRST STEPS IN MACHINE LEARNING
TOPIC: DEEP NEURAL NETWORKS AND ESTIMATORS

In this material, we will explore the concept of deep neural networks and estimators in the context of machine learning. As datasets become increasingly complex, it can be challenging to achieve high accuracies using linear models. Deep neural networks offer a solution to this problem by adapting to complex datasets and generalizing better to unseen data.

Deep neural networks, as the name suggests, consist of multiple layers that allow them to fit more complex datasets compared to linear models. However, there are trade-offs associated with using deep neural networks. They generally take longer to train, are larger in size, and have less interpretability compared to linear models. Despite these drawbacks, deep neural networks can lead to higher final accuracies on complicated datasets.

One of the challenges in deep learning is finding the right set of parameters for the model. The configurations can seem limitless, depending on the dataset. To address this, TensorFlow provides built-in deep classifier and deep regressor classes that offer a reasonable set of defaults. These defaults allow data scientists to get started quickly and easily.

To illustrate the transition from a linear model to a deep neural network, let's consider an example using the Iris dataset. Although we won't showcase a 2,000-column model in this material, we will use the four columns that have been used throughout this series. The main change involves replacing the linear classifier class with the DNN classifier, which creates a deep neural network. Most of the remaining code remains the same.

When working with deep neural networks, an additional parameter is required: the hidden units argument. Since deep neural networks have multiple layers, each layer can have a different number of nodes. The hidden units argument allows you to specify the number of nodes for each layer using an array. This way, you can create a neural network by considering its size and shape instead of writing the entire network from scratch.

The estimators framework in TensorFlow simplifies the process of converting a linear model to a deep neural network. By leveraging the framework, you can organize your data, training, evaluation, and model exporting in a common way. This approach provides flexibility to experiment with different models and parameters while reducing the need for extensive code changes.

The DNN classifier also offers additional parameters that can be customized, such as the optimizer, activation function, and dropout ratios. These parameters allow for further fine-tuning of the deep neural network.

Deep neural networks offer a powerful solution for training on complex datasets. By leveraging the estimators framework in TensorFlow, you can easily transition from a linear model to a deep neural network with minimal code changes. While deep neural networks have certain drawbacks, such as longer training times and decreased interpretability, they can achieve higher accuracies on challenging datasets.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: BIG DATA FOR TRAINING MODELS IN THE CLOUD

When training machine learning models, it is common to encounter situations where the training data is too large to fit on a local machine or the training process takes a significant amount of time. In such cases, moving the training to the cloud can be a viable solution. By shifting the machine learning training to another computer with access to more storage, you can free up your local hard drive space and continue working on other tasks while the training is being conducted.

To understand how to move the training to the cloud, it is important to consider two primary resources: compute and storage. These resources can be decoupled, meaning they can be handled separately, which can lead to more efficient systems when dealing with big data. While compute load can be easily moved around, transferring large datasets can be more involved. However, the benefits are significant, as it allows the data to be accessed by multiple machines in parallel, thereby accelerating the machine learning training process.

Google Cloud Platform offers convenient ways to connect these resources. Firstly, it is recommended to ensure that the data is stored in Google Cloud Storage (GCS). This can be achieved using tools like gsutil, a command line tool specifically designed for interacting with Google Cloud Storage. For smaller to medium-sized datasets, gsutil is a suitable option, as it supports parallel processing, enabling faster transfer jobs.

In cases where the dataset is too large to be transferred over the network, Google provides the option of using the Google Transfer Appliance. This physical device is shipped to your data center and securely captures and transfers a whole petabyte of data. Compared to transferring data over the network, which could take several years or months depending on the network bandwidth, the transfer appliance can complete the task in just 25 hours.

Once the data is in the cloud, you are ready to run machine learning training at scale. However, the details of training machine learning models on large datasets will be covered in the next material. It is worth noting that training on large datasets no longer needs to be challenging with limited compute and storage resources. By leveraging the cloud and tools like gsutil or the transfer appliance, you can train on large datasets without any issues.

In the subsequent material, we will explore the compute side of running machine learning in the cloud, including options for utilizing GPUs. For now, remember that when working with big data sets for machine learning, putting your data in the cloud is the way to go.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: NATURAL LANGUAGE GENERATION

Natural language generation (NLG) is a field within natural language processing (NLP) that focuses on teaching computers to generate human-like language based on structured data. In the context of conversational user interfaces, NLG plays a crucial role in enabling computers to respond to users in a natural and intelligent manner.

One of the key challenges in NLG is ensuring that the generated language makes sense in the context of the conversation. This involves determining whether the response is appropriate and relevant to the user's input. For example, suggesting a coffee shop when asked about dinner plans would be considered an inappropriate response. Additionally, it is important to ensure that the language used is grammatically correct and unambiguous.

Structured data plays a significant role in addressing the first requirement of generating appropriate responses. By analyzing structured data, such as weather forecasts or search results, computers can extract relevant information to answer user queries. For instance, if a user asks about the weather in Santa Clara next week, the NLG system can utilize structured data to generate a response that provides accurate and specific information.

The ultimate goal of NLG is to enable computers to transform structured data into natural language that can be used to engage in conversations with users. By leveraging NLG techniques, conversational user interfaces, like the Google Assistant, can provide intelligent and contextually relevant responses to user queries.

NLG is a vital component of NLP that focuses on teaching computers to generate human-like language based on structured data. It involves addressing the challenges of generating appropriate and grammatically correct responses, while also leveraging structured data to provide accurate and contextually relevant information.

Natural language generation is a crucial aspect of machine learning, particularly when it comes to generating responses to user queries. The goal is to create a conversational and interactive experience for users, rather than simply providing them with static information. In this context, the challenge lies in transforming structured data into natural language responses.

One possible solution is to use a template-based approach, where predefined templates are filled in with relevant data to generate responses. However, this approach often lacks conversational flow and can sound robotic. To illustrate this, let's consider an example where we generate a weather forecast using a template. We could iterate through the days of the week, filling in the blanks with temperature and weather conditions. While this approach is straightforward, it lacks the conversational aspect we desire.

To address this limitation, we need to find a way to generate natural language from structured data. One approach is to consider how humans would answer the question and try to replicate that process using a computer system. As humans, we would summarize the information, identify repetitive patterns, and provide a more contextually aware response. For example, instead of listing the weather conditions for each day, we might summarize it as "cloudy until Thursday with showers the rest of the week. Temperatures range from the high 40s to the mid-60s."

However, achieving this level of natural language generation is non-trivial. It requires the use of machine learning techniques to automate the process. Without machine learning, we would have to rely on writing numerous rules, which are often specific, require significant engineering effort, and are not easily scalable to new inputs and outputs. For instance, if we wanted to discuss finance instead of weather or support a different language, we would need to create an entirely new set of rules.

Machine learning offers a more scalable and creative solution to natural language generation. By providing the model with examples of data and the desired language output, we can enable the model to learn its own rules. This eliminates the need for manual rule-writing and allows the computer to be more creative in generating responses.

In our research, we have explored various machine learning architectures, with recurrent neural networks

(RNNs) showing promising results. RNNs are a type of neural network that can process sequential data, making them suitable for generating natural language responses. However, RNNs are just one of the architectures we are exploring, and there may be other approaches that prove equally effective.

Natural language generation plays a crucial role in creating conversational and interactive experiences. Machine learning, particularly using architectures like recurrent neural networks, offers a scalable and creative solution to this problem. By providing the model with examples of data and language output, we can enable it to generate natural language responses without relying on manual rule-writing.

Recurrent Neural Networks (RNNs) are a type of deep neural network that have the ability to make decisions over several time steps. Unlike traditional neural networks, RNNs have a feedback loop that allows the outputs of the network to feed back into itself. This makes RNNs especially good at remembering what it saw earlier, as it enforces a sequential policy over the data.

Language, in general, is extremely sequential, and RNNs are particularly useful for natural language generation. The order of words in a sentence matters, and RNNs excel at capturing this sequential nature of language. They are able to generate language by outputting one character at a time, allowing for fine-grained control over the output granularity. In the case of character-based RNNs, the model generates output at the character level.

One interesting aspect of RNNs is their ability to learn to pay attention to specific pieces of structured data at different time steps. This is illustrated in a visualization where each row represents different pieces of structured data, and each column represents a single step in the model. The shading of the squares indicates how much the model cares about that particular piece of data at a given time step. By examining this visualization, we can see how the model has learned to pay attention to different parts of the data as it generates output character by character.

One notable result in the visualization is the presence of a diagonal line in the middle. This line represents the model's ability to read specific characters for a specific location in the sentence. The model has learned to spell out the specific location by looking at the data character by character. Remarkably, no one explicitly taught the model to do this; it learned to reference the data and make decisions based on examples.

RNNs are powerful tools for natural language generation due to their ability to capture the sequential nature of language. They can generate language character by character, allowing for fine-grained control over the output granularity. RNNs also have the ability to learn to pay attention to specific pieces of structured data, enabling them to make informed decisions during the generation process.

Artificial Intelligence (AI) has made significant advancements in recent years, particularly in the field of natural language generation. Natural language generation refers to the ability of AI systems to generate human-like text or speech. In this didactic material, we will explore the concepts discussed in a previous material on AI adventures, focusing on the use of machine learning for natural language generation and its role in conversational user interfaces.

Machine learning is a branch of AI that involves training algorithms to learn patterns and make predictions or decisions without being explicitly programmed. Natural language generation can be achieved using various machine learning techniques, such as recurrent neural networks (RNNs) and transformers. These models are trained on large datasets of text, allowing them to learn the statistical patterns and structures of language.

One of the key applications of natural language generation is in conversational user interfaces, such as chatbots or virtual assistants. These systems rely on AI to understand and generate human-like responses in natural language. By leveraging machine learning, these interfaces can provide more engaging and interactive experiences for users.

During the conversation, the speaker mentioned the excitement about future research and the potential for writing research papers using these networks. This highlights the ongoing exploration and development in the field of natural language generation. As AI continues to advance, there are still many unexplored possibilities and opportunities for further research.

Natural language generation is a fascinating field within AI that has the potential to revolutionize how we interact with technology. By leveraging machine learning techniques, AI systems can generate human-like text

or speech, enabling more natural and engaging conversations. The development of conversational user interfaces is just one example of how natural language generation can be applied. As researchers and developers continue to push the boundaries of AI, we can expect to see even more exciting advancements in the future.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: DISTRIBUTED TRAINING IN THE CLOUD

In this didactic material, we will explore the concept of distributed training in the cloud using Google Cloud Machine Learning. Training big models in the cloud requires scaling compute power from machine learning. In the previous material, we discussed the challenges of handling large datasets and how to leverage scalable storage in the cloud. Now, we will focus on the second part of the problem, which involves managing compute resources for training larger models.

The current approach to training larger models involves parallel training. This means that the data is split and sent to multiple worker machines. The model then combines the information received from these workers to create the fully trained model. Setting up virtual machines, installing network libraries, configuring them for distributed machine learning, and managing compatibility issues can be challenging for those unfamiliar with the process.

To simplify this process, we can use Cloud Machine Learning Engine's training functionality. With this service, we can go from Python code to a trained model without the need for infrastructure work. The service automatically acquires and configures the necessary resources and shuts them down when training is complete.

There are three main steps to using Cloud Machine Learning Engine for distributed training. First, we need to package our Python code. This involves placing our code in a separate script and wrapping it inside a Python package. The package contains the module and an empty file called "__init__.py". This structure allows us to call the module from other files.

Next, we create a configuration file that specifies the desired machines for training. We can choose to run training with a small number of machines or scale up to many machines with GPUs attached. There are predefined specifications available, and we can also configure a custom architecture if needed.

Once our code is packaged and the configuration file is ready, we can submit the training job. This can be done using the gcloud command line tool or the equivalent REST API call. We provide a unique job name, the package path and module name, the region for the job to run in, and a cloud storage directory to store the training outputs. It is important to choose the same region as where the data is stored for optimal performance.

After running the command, the Python package is zipped and uploaded to the specified directory. The package will then be executed in the cloud on the machines specified in the configuration. We can monitor the training job in the Cloud Console by navigating to ML Engine and selecting Jobs. Here, we can see a list of all the jobs, including the current one, along with a timer indicating the job's progress and a link to the logs.

Once the training is complete, the trained model is exported to the specified cloud storage path. From there, we can easily create a prediction service by pointing to the outputs of the model. This allows us to make predictions at scale using Cloud Machine Learning Engine's serverless predictions.

By using Cloud Machine Learning Engine, we can achieve distributed training without the need to manage infrastructure ourselves. This frees up more time to focus on working with our data. Simply package the code, add the configuration file, and submit the training job to train your model efficiently.

Distributed Training in the Cloud

We will now explore the concept of distributed training in the cloud. In this material, we will discuss the benefits and techniques of distributed training, which is a crucial aspect of machine learning.

Distributed training involves training machine learning models on multiple machines simultaneously. This approach offers several advantages, including increased speed and the ability to handle larger datasets. By distributing the training process across multiple machines, we can significantly reduce the time required for model training.

One popular platform for distributed training is Google Cloud Machine Learning. Google Cloud provides a

powerful infrastructure that allows users to train their models efficiently and effectively. With Google Cloud, you can easily scale your training jobs to thousands of machines, enabling you to tackle large-scale machine learning tasks.

To perform distributed training in Google Cloud, you need to leverage the power of TensorFlow, an open-source machine learning framework. TensorFlow provides a high-level API called tf.distribute.Strategy, which simplifies the process of distributing your training across multiple machines.

There are several strategies available in TensorFlow for distributed training. One common approach is data parallelism, where each machine trains on a different subset of the training data. Another technique is model parallelism, where different machines train on different parts of the model.

When using data parallelism, the training data is divided into smaller batches, and each machine processes a different batch. The gradients computed by each machine are then averaged to update the model parameters. This approach allows for parallel processing of the training data, resulting in faster training times.

In model parallelism, the model is divided into smaller parts, and each machine trains on a different part. The outputs from each machine are then combined to obtain the final model. This technique is particularly useful for training large models that cannot fit into the memory of a single machine.

By leveraging the power of distributed training in the cloud, you can train complex machine learning models more efficiently. Distributed training allows you to process larger datasets and reduce training times, enabling you to iterate on your models more quickly.

Distributed training in the cloud is a powerful technique that can significantly enhance the training process for machine learning models. By leveraging platforms like Google Cloud Machine Learning and TensorFlow, you can take advantage of distributed training to achieve faster and more efficient model training.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: MACHINE LEARNING USE CASE IN FASHION

In this educational material, we will explore a machine learning use case in the fashion industry using Google Cloud Machine Learning. We will walk through the steps of training a model to detect different types of clothing using the Fashion-MNIST dataset.

The Fashion-MNIST dataset is similar to the classic MNIST dataset, but instead of handwritten digits, it consists of pictures of various clothing types such as shoes, bags, and different categories of clothing. The images in the dataset are still 28-by-28 pixels, and there are 10 different categories.

To start, we will build a linear classifier using TensorFlow's Estimator Framework. We will flatten the images from 28-by-28 to 1-by-784 pixels and create a feature column called "pixels." Next, we will create our linear classifier with 10 possible classes for labeling.

To train our model, we will set up our dataset and input function using TensorFlow's built-in utility that accepts a NumPy array. We will load the Fashion-MNIST dataset and point to the folder where it is stored. Then, we can call the classifier.train function to bring together our classifier, input function, and dataset.

After training, we will run an evaluation step to see how our model performed. Compared to the classic MNIST dataset, the Fashion-MNIST dataset is more complex, and we can typically achieve an accuracy in the low 80's or even lower.

To improve our model's performance, we can switch to a deep neural network (DNN) classifier. This is a one-line change, and we can rerun the training and evaluation to see if the DNN classifier performs better than the linear one. TensorBoard can be used to compare the performance of these two models side by side.

If the DNN model is not performing better, we can experiment with hyperparameters such as model size and learning rate to achieve higher accuracy. Deep networks often take longer to train compared to linear models due to their complexity.

Once we are satisfied with our model, we can export it and create a scalable Fashion-MNIST classifier API. This allows us to make predictions using our trained model.

To make predictions with estimators, we can use a similar approach to calling the Train and Evaluate functions. However, we specify a batch size of 1, num_epochs of 1, and shuffle as False to preserve the order of predictions. We can extract a subset of images from the dataset and make predictions on each image individually.

In the evaluation data set, we have selected five images for prediction. Two of these images were misclassified by the model, with the third example being predicted as a bag and the fifth example as a coat, incorrectly. These examples highlight the challenge of classifying clothing images due to their graininess compared to handwritten numbers.

This educational material demonstrated the process of training a machine learning model using the Fashion-MNIST dataset in the fashion industry. We started with a linear classifier and then switched to a deep neural network for better performance. We also explored how to make predictions using estimators and discussed the challenges of classifying clothing images.

After training your model, it is important to evaluate its performance. You can achieve this by assessing the accuracy of the model using specific parameters. The code used to train the model and generate the images can be found in the provided links. Additionally, there are more resources available in the links for further exploration.

In the upcoming materials, we will focus on the tools available in the machine learning ecosystem. These tools will assist you in constructing your workflow and tool chain. Furthermore, we will showcase various architectures that can be utilized to solve specific machine learning problems.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: DATA WRANGLING WITH PANDAS (PYTHON DATA ANALYSIS LIBRARY)

Pandas is a powerful and popular tool in data wrangling, which is an important step in machine learning. It is an open-source Python library that provides easy-to-use, high-performance data structures and data analysis tools. The name "Pandas" comes from the term "panel data," which refers to multi-dimensional data sets encountered in statistics and econometrics.

To install Pandas, you can run the command "pip install pandas" inside your Python environment. Once installed, you can import Pandas using the alias "pd".

One of the most common uses of Pandas is reading in CSV files using the "read_csv" function. This function loads the data into a DataFrame, which can be thought of as a table or a spreadsheet. You can quickly glimpse your data by calling the "head" function on the DataFrame.

DataFrames have rows of data with named columns, which are called series in Pandas. You can access a particular column of a DataFrame by using bracket notation and passing in the name of the column. If you want to access a particular row of a DataFrame, you can use the "iloc" function and specify the index of the row.

Pandas provides a variety of functions for data manipulation. For example, you can use the "describe" function to display a table of statistics about your DataFrame, which is useful for sanity checking your data set. You can also shuffle your data using Pandas, which can be helpful in situations where you want to shuffle the entire data set.

To access a range of columns or rows, you can use colon notation inside the brackets that follow the "iloc" function. The starting index is included, while the ending index is excluded. If you want to select both a subset of columns and a subset of rows, you can combine the techniques we've learned so far.

This is just an overview of what Pandas can do. The Pandas ecosystem offers many more features, such as efficient file storage in the form of PyTables and HDF5 format, as well as running certain statistical analyses.

Pandas is a powerful tool for data wrangling in Python. It provides easy-to-use data structures and analysis tools, making it a popular choice for manipulating and preparing data for machine learning.

Pandas is a Python library that provides powerful data manipulation and analysis capabilities. In this didactic material, we will provide a brief overview of Pandas and its functionalities.

Pandas allows users to work with structured data, such as tabular data, in an efficient and intuitive manner. It provides data structures and functions that make data cleaning, transformation, and analysis tasks easier.

One of the main data structures in Pandas is the DataFrame, which is a two-dimensional table-like structure. It consists of rows and columns, where each column can have a different data type. The DataFrame is particularly useful for handling structured data and performing operations on it.

With Pandas, you can easily load data from various sources, such as CSV files, Excel spreadsheets, or databases, into a DataFrame. Once the data is loaded, you can perform a wide range of operations on it, such as filtering rows, selecting specific columns, sorting data, and aggregating values.

Data cleaning is an important step in any data analysis process. Pandas provides functions to handle missing values, remove duplicates, and perform other data cleaning tasks. You can also apply transformations to the data, such as changing data types, creating new columns, or applying mathematical operations.

Pandas also offers powerful data visualization capabilities. You can create various types of plots, such as line plots, scatter plots, or bar plots, to explore and visualize your data. These visualizations can help you gain insights and communicate your findings effectively.

To get started with Pandas, you can refer to the multitude of Internet resources. The official documentation

provides detailed explanations of Pandas' functionalities, as well as good examples and materials to help you learn and apply them in practice.

Documentation links:
- Pandas Documentation: https://pandas.pydata.org/docs/
- Pandas User Guide: https://pandas.pydata.org/docs/user_guide/index.html
- Pandas API Reference: https://pandas.pydata.org/docs/reference/index.html

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: INTRODUCTION TO KAGGLE KERNELS

Kaggle Kernels are a feature of the Kaggle platform that allows users to run Jupyter Notebooks directly in their browser. This means that users can access a Jupyter Notebook environment anywhere in the world with an internet connection, without the need to set up a local environment. Additionally, the processing power for the notebooks comes from servers in the cloud, rather than the user's local machine, allowing for more intensive data science and machine learning tasks without overheating the user's device.

To get started with Kaggle Kernels, users need to create an account on Kaggle.com. Once logged in, users can choose a dataset they want to work with and create a new kernel or notebook with just a few clicks. The selected dataset is preloaded in the kernel environment, eliminating the need to transfer large datasets over a network.

In the provided example, the dataset used is a collection of grayscale images of clothing and accessories, with 10 categories and a total of 60,000 samples. The images were originally 28 by 28 pixels but have been flattened to 784 distinct columns in a CSV file. The CSV file also contains a column representing the index of each fashion item.

After loading the data into a pandas data frame, users can take advantage of the features provided by pandas, such as displaying the first few rows and using the describe function to explore the dataset's structure. Additionally, users can visualize the images using the matplotlib.pyplot library, which allows them to see the clothing and accessory items represented by the pixel values.

Kaggle Kernels provide a fully interactive notebook environment in the browser, with no need for Python environment configuration or library installations. This makes it easy for users to start working on data science projects without any setup hassle.

Kaggle Kernels are a powerful tool for data science and machine learning, allowing users to run Jupyter Notebooks in their browser without the need for local setup. With the ability to access datasets, perform data analysis, and visualize data, Kaggle Kernels provide a convenient and efficient environment for practicing and learning data science.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: WORKING WITH JUPYTER

Jupyter notebooks are a powerful tool for working with Python code in an interactive and visual manner. They allow you to write and run code right in your browser, providing a convenient way to experiment and explore data. In this didactic material, we will explore the features and functionalities of Jupyter notebooks.

To begin, Jupyter is built on the IPython project and offers more than just running Python code. It supports bash commands, special magics, and plugins, enhancing the Python coding experience. If you haven't used Jupyter before, it is recommended to install it by running the command "pip install jupyter". However, if you are using a packaged Python distribution like Anaconda, it may already be installed.

When running Jupyter locally, you connect to a locally-running web server through your browser, typically on port 8888. To start a notebook, navigate to your working directory and run the command "jupyter notebook". The notebook will automatically open in your browser, but if not, you can access it by going to "localhost:8888". If you don't have a notebook to open, you can create a new one by clicking on "New" and selecting the Python version you are using.

Once you have a notebook open, you can start writing Python code in the empty cells. To run a cell, simply press "Control-Enter". You can run typical Python code and see the results immediately. When a cell is running, an asterisk appears in the brackets on the left side of the cell. Once it finishes, a number representing the order of execution is displayed. The results of the final line of a code cell are printed as the output of that cell, unless the value is stored to a variable.

Jupyter notebooks also offer a convenient way to access function documentation. By pressing "Shift-Tab" while calling a function, you can view its docstring, which provides information about the function's arguments and usage. This feature works not only with built-in functions but also with your own custom functions if you have written good docstrings.

To manage output, you can reduce the space it takes up by clicking on the left-hand panel of the output, which turns it into a scrolling window. Double-clicking collapses the output entirely.

Adding new cells is easy. You can click the plus icon in the toolbar to add a cell below the current one. Additionally, there are cell execution commands that can create new cells for you. Pressing "Shift-Enter" runs the current cell and highlights the next one. If there is no next cell, a new one is created. On the other hand, if you want to insert a new cell immediately after a given cell, you can use "Alt-Enter" to execute the cell and then insert a new one.

One of the most powerful features of Jupyter notebooks is the markdown support. Markdown allows you to write formatted text, including headings, lists, and links, in addition to code. This makes it easy to document your code and provide explanations for your work. To use markdown, simply change the cell type to "Markdown" and start writing.

Jupyter notebooks also offer some additional functionalities. For example, you can time your code execution by starting a cell with "%%time". After running the cell, it will display the time it took to execute. This is useful for quick checks on performance. Additionally, you can run command line commands in your notebook by prefixing the command with an exclamation point. This is useful for running one-off commands.

Jupyter notebooks provide an interactive and visual environment for working with Python code. They allow you to write and run code in your browser, view function documentation, manage output, add new cells, and write formatted text using markdown. These features make Jupyter notebooks a powerful tool for data exploration, experimentation, and documentation.

In Jupyter notebooks, you can start a cell with "%%bash" to interpret the entire cell as a bash script. This is particularly useful for running TensorBoard. Normally, you would have to open a new terminal window and run TensorBoard from the command line, but having it within a Jupyter notebook cell can be convenient if you just want to quickly start it, take a look, and then close it. However, please note that while TensorBoard is running, it

will occupy your notebook and you won't be able to run anything else. To stop TensorBoard, simply click "Interrupt Kernel" and you will regain control.

These are just some of the features and capabilities of Jupyter that are particularly useful. There are many more features waiting for you to explore. In the meantime try Jupyter notebooks and discover all that they have to offer.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: FURTHER STEPS IN MACHINE LEARNING
TOPIC: CHOOSING PYTHON PACKAGE MANAGER

Every data scientist has different preferences when it comes to their programming environment. Today we will discuss the Python package manager, Pip, and two popular options for managing different Pip packages: virtualenv and Anaconda.

Pip is Python's package manager and comes built into Python. It installs packages like tensorflow, numpy, pandas, and Jupyter, along with their dependencies. Many Python resources are delivered in the form of Pip packages, and you can easily install everything needed by using the command "pip install -r requirements.txt", where requirements.txt is a file that outlines all the Pip packages used in a project.

When working on different projects, you may need to use different versions of a library. To organize groups of packages into isolated environments, you can use virtualenv or Anaconda. Virtualenv allows you to create named virtual environments where you can install Pip packages in an isolated manner. This tool is great if you want detailed control over which packages you install for each environment. For example, you could create one environment for web development and another for data science, ensuring that unrelated libraries do not interact with each other.

Anaconda, created by Continuum Analytics, is a Python distribution that comes preinstalled with many useful Python libraries for data science. It is popular because it brings together the tools used in data science and machine learning with just one install, making the setup process simple. Like virtualenv, Anaconda also uses the concept of creating environments to isolate different libraries and versions. Anaconda introduces its own package manager called conda, from where you can install libraries. Additionally, Anaconda still allows you to use Pip to install any additional libraries not available in the Anaconda package manager.

If you find yourself needing to test different libraries on both virtualenv and Anaconda, you can use a library called pyenv. Pyenv sits atop both virtualenv and Anaconda and allows you to control which environment is in use, as well as whether you're running Python 2 or Python 3. Pyenv also has the ability to set a default environment for a given directory, automatically activating it when you enter that directory. This can be helpful in managing your environments and simplifying your workflow.

Ultimately, the choice between virtualenv and Anaconda depends on your workflow and preferences. If you primarily use core data science tools and don't need extra libraries, Anaconda can provide a simpler workflow. However, if you enjoy customizing your environment, virtualenv or pyenv may be more suitable. There is no one right way to manage Python libraries, and new tools and options are constantly emerging. It's important to choose the tools that best serve your needs and preferences.

Managing Python libraries involves making decisions about package managers and creating isolated environments. Pip is Python's package manager and installs packages and their dependencies. Virtualenv and Anaconda are popular options for managing different Pip packages. Virtualenv allows for detailed control over package installation, while Anaconda simplifies the setup process by providing preinstalled libraries for data science. Pyenv can be used to manage both virtualenv and Anaconda environments, providing control over Python versions and default environments for directories.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE TOOLS FOR MACHINE LEARNING
TOPIC: GOOGLE CLOUD DATALAB - NOTEBOOK IN THE CLOUD

Streaming data down to a local environment can be slow and costly. In this material, we will explore how to bring the notebook environment to the data using Google Cloud Machine Learning tools, specifically Google Cloud Datalab.

Google Cloud Datalab is built on top of the Jupyter notebook and offers additional functionalities such as easy authentication with BigQuery datasets, fast operations to Google Cloud Storage, and SQL query support. To get started, we need to install the Datalab component using the command "gcloud components install datalab". Once installed, we can start Datalab with a single command. This command sets up a virtual machine in the cloud, configures the network, and installs necessary libraries like TensorFlow, pandas, NumPy, and more.

Once Datalab is running, it opens a notebook environment that is similar to Jupyter notebooks. However, this environment runs on a virtual machine in the cloud. Datalab comes with some default samples that provide a great starting point for exploration. For example, the "Hello World" notebook in the Docs folder allows us to run cells and experiment without the need to manage different Python libraries or configurations.

Datalab also provides additional tools. By clicking on the Account icon in the upper right corner, we can access options and information. The notebook runs as a service account, already authenticated with the project's assets. If we want to access resources from another project, we need to grant access to the service account rather than the user account.

The Datalab notebook is hosted on a virtual machine called "AI Adventures" running on Google Compute Engine. We can shut down the VM by clicking a button, and Datalab has a feature that automatically shuts down the virtual machine after 30 minutes of inactivity. This timeout can be customized or disabled in the Settings.

In this material, we will run an example that analyzes the correlation between programming languages used on GitHub. The notebook uses NumPy and pandas to analyze BigQuery data and visualize it using Matplotlib. The data is pulled from the GitHub timeline table in the public dataset samples provided on BigQuery.

To interact with BigQuery, we can use the "%%BQ" command in a cell to send operations to BigQuery. This allows us to query the data and perform analysis directly within the notebook environment.

Google Cloud Datalab provides an interactive notebook environment in the cloud, allowing us to bring our compute to the data. It offers seamless integration with Google Cloud services, authentication with BigQuery datasets, and support for SQL queries. With Datalab, we can analyze large datasets and run powerful tools without the need for local machine resources.

Artificial Intelligence (AI) has become an integral part of various industries, including the field of Machine Learning (ML). Google Cloud offers a range of tools and services that enable developers and data scientists to leverage AI and ML capabilities effectively. One such tool is Google Cloud Machine Learning, which provides a platform for building and deploying ML models at scale.

Another useful tool in the Google Cloud ecosystem is Google Cloud Datalab, a notebook environment that runs in the cloud. Datalab allows users to analyze data, visualize results, and collaborate with others seamlessly. It provides a convenient interface to interact with Google Cloud services, including BigQuery, a fully-managed, serverless data warehouse.

With BigQuery, users can analyze large datasets without the need for additional scripting or dealing with REST API responses. By running queries, users can retrieve metadata about the dataset, such as the number of rows. Additionally, BigQuery offers a sampling feature, allowing users to retrieve a smaller chunk of data for quick analysis and data format validation.

In the context of analyzing GitHub commit data, Datalab provides the capability to write SQL queries directly using the BQ magic command. This allows for syntax highlighting and a more intuitive query writing experience. By executing these queries, users can obtain insights into the languages associated with GitHub commits. For

example, JavaScript, Java, and Ruby are among the most frequently used languages, while R, Matlab, and Puppet have fewer commits in the sample dataset.

To further analyze the data, Datalab offers seamless integration with pandas, a powerful data analysis library in Python. Users can easily load query results into pandas data frames, enabling them to apply various analysis techniques and explore interesting statistics. For instance, users can identify contributors who have contributed to the most distinct languages.

To correlate different languages, users can reshape the data using pandas' pivot function, creating a table with user names as columns and 25 different languages as rows. This table can then be used to compute correlations between language pairs, revealing notable correlations using the core function. However, visualizing these correlations in a decimal table might not be intuitive. To address this, Datalab leverages Matplotlib, a popular data visualization library, to create a colorful plot that highlights positive and negative correlations between language pairs.

The resulting plot allows users to identify interesting language pairs and their correlations. For example, it is observed that Java has limited association with languages like JavaScript, PHP, and Ruby. On the other hand, C and C++ exhibit a strong correlation. CoffeeScript and JavaScript also show a positive correlation. Moreover, Ruby and PHP exhibit a significant negative correlation across their rows, indicating that they are less likely to be used together.

It is important to note that the analysis presented here is based on a small sample of the larger GitHub public dataset. For those interested in working with the full GitHub commit history, a link to the public dataset is provided for further exploration. Google Cloud Datalab serves as an excellent tool for running cloud-connected notebooks close to the data, with convenient connectors to services like BigQuery and easy authentication.

Google Cloud provides a comprehensive set of tools, including Google Cloud Machine Learning and Google Cloud Datalab, that empower users to leverage AI and ML capabilities effectively. These tools enable data scientists and developers to analyze large datasets, visualize results, and gain valuable insights. By combining the power of BigQuery, pandas, and Matplotlib, users can perform advanced data analysis and explore correlations between different programming languages.

Google Cloud Machine Learning offers a range of tools and services to help developers and data scientists build and deploy machine learning models. One of these tools is Google Cloud Datalab, which provides a notebook environment in the cloud for interactive data exploration, analysis, and visualization.

With Google Cloud Datalab, you can easily access and analyze your datasets stored in the cloud. It allows you to write and execute code in a notebook format, making it easy to iterate and experiment with your data. You can also leverage the power of Google Cloud's infrastructure to scale your computations and handle large datasets.

Using Datalab, you can import and export data from various sources, such as Google Cloud Storage, BigQuery, or even your local machine. This flexibility allows you to work with data from different locations and formats seamlessly. Datalab also provides built-in support for popular machine learning libraries, such as TensorFlow and scikit-learn, enabling you to train and evaluate models directly within the notebook.

In addition to data analysis and model development, Datalab offers powerful visualization capabilities. You can create interactive charts, plots, and dashboards to gain insights from your data and communicate your findings effectively. Datalab integrates with tools like Google Charts and Matplotlib, making it easy to generate visualizations that enhance your analysis.

To get started with Datalab, you can create a new notebook instance in the Google Cloud Console. Once your instance is ready, you can open the notebook interface in your web browser and start working with your data. Datalab provides a rich set of features, including code autocompletion, inline documentation, and integrated help, to support your data exploration and analysis workflow.

Google Cloud Datalab is a powerful tool for data scientists and developers working with machine learning. It provides a notebook environment in the cloud, enabling interactive data exploration, analysis, and visualization. With its seamless integration with Google Cloud services and popular machine learning libraries, Datalab simplifies the process of building and deploying machine learning models.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE TOOLS FOR MACHINE LEARNING
TOPIC: PRINTING STATEMENTS IN TENSORFLOW

When working with TensorFlow, it is important to understand how print statements work in the context of graph computations. Unlike typical print statements in Python, TensorFlow's print statement does not immediately display the values of operations. Instead, it shows a description of the operation and, if known, the expected dimensions of the node. To print the values flowing through a specific part of the graph during execution, we need to use TensorFlow's tf.Print function.

To incorporate the print statement into the graph, we need to assign the output of the print call to a variable that will serve as an input in a later node. This effectively links the print statement into the flow of operations. While the print statement itself does not perform any computation, it prints the desired node as a side effect.

One important thing to note is that only the nodes of the graph that need to be executed will be executed. If there is a dangling print node in the graph, it will not be executed. To ensure that the print statement is executed, we need to splice the print call into the graph by assigning its output to a variable that is used as an input in a subsequent node.

In addition to printing a single node, tf.Print allows us to print multiple nodes by passing an array of nodes as the second argument. This can be useful for debugging and understanding the values of multiple nodes in the graph. We can also include a message as the third argument to prepend a string before printing the nodes, making it easier to identify specific print statements in the logs.

One common use case for tf.Print is in input functions, where it can be used to debug and understand the data being passed into the training process. Unlike using the Python print function, tf.Print can be integrated into the data pipeline of the input function, allowing us to print the values of nodes as they are passed through the pipeline.

However, it is important to set limits on the amount of data being passed into tf.Print to avoid generating excessively long log files.

TensorFlow's print statement, tf.Print, is a powerful tool for understanding the values flowing through a computational graph. By splicing the print call into the graph and using it strategically, we can gain valuable insights into the execution of our machine learning code.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE TOOLS FOR MACHINE LEARNING
TOPIC: TENSORFLOW OBJECT DETECTION ON IOS

To build a custom object recognition mobile app starting from raw data, there are several steps involved. In this didactic material, we will discuss the process of using Google Cloud Machine Learning tools, specifically the TensorFlow Object Detection API, to train a model for object detection on iOS devices.

The first step is to gather and preprocess the data. In this case, the developer, Sara Robinson, wanted to train a model to recognize Taylor Swift in images. She used her Google Photos account to find all the pictures of herself, which served as the training data. The Google Photos search function was particularly helpful in this process.

Next, Sara trained the model locally using the TensorFlow Object Detection API. This API provides utilities that make it easier to recognize objects in an image. Instead of just classifying an image, the API returns bounding boxes that indicate the location of the object in the image. The Object Detection API utilizes transfer learning, which involves using a pre-trained model on a similar classification task and updating it with new data. In this case, Sara used a pre-trained model that had been trained on millions of images and updated it with her own data.

To use the Object Detection API, the images need to be in a specific format called TFRecord. Sara converted her images to the Pascal VOC format, which is an XML file that contains the coordinates of the bounding boxes and the associated labels. In this case, the label was "Taylor Swift."

After preparing the data, Sara ran the training process, which was much faster than starting from scratch due to the benefits of transfer learning. Instead of taking days, the training only took a couple of hours.

Once the model was trained, Sara wanted to build an end-to-end demo by creating an iOS client app. She used Swift, a programming language for iOS development, to build the app. The app sends a prediction request to the trained model to detect Taylor Swift in images.

This project showcases the power of Google Cloud Machine Learning tools, such as the TensorFlow Object Detection API, in building custom object recognition applications. By leveraging pre-trained models and transfer learning, developers can save time and resources while achieving accurate results.

Building a custom object recognition mobile app involves gathering and preprocessing the data, training the model using the TensorFlow Object Detection API, and creating a client app to make predictions. The use of transfer learning and pre-trained models simplifies the training process and improves efficiency.

To train a machine learning model for object detection using TensorFlow on iOS, several steps need to be followed. First, the images need to be labeled with bounding boxes and converted into the Pascal format. Once in this format, a script can be used to convert the labeled images into TFRecord files, which are used to feed into the model.

To train the model using Google Cloud Machine Learning Engine, the data needs to be stored in Google Cloud Storage. This includes the model checkpoints from the pre-trained model, the training and test TFRecord files, and a config file that specifies the location of the data. ML Engine is then used to run the training process, which took approximately 30 minutes in this case.

After training, the result is a model checkpoint file that has been updated with the training data. To serve the model, the gcloud command with ML Engine is used to deploy the model. The checkpoint files of the final trained model are specified, and within a few minutes, the model is available for serving. The model can then be used to generate predictions.

To call the endpoint and make predictions, a Swift app written in iOS is used as the client. The client uploads an image to Firebase Storage. A Firebase Cloud Function is triggered whenever an image is uploaded to a specific storage bucket. The function, written in Node.js, downloads the image, resizes it, and base64 encodes it to prepare it for the ML Engine online prediction request. The prediction request is made, and if the confidence of

the prediction is greater than 70%, a box is drawn around the image using ImageMagick. The annotated image is then written back to Firebase Storage, along with metadata such as the confidence of the detection and the path of the image in Firebase Storage. This information is also stored in Cloud Firestore, a database.

This process involves labeling and converting images, training the model on ML Engine, deploying the model for serving, and making predictions using the client app and Firebase Cloud Functions.

Firestore is a useful tool in this context because it allows the creation of a listener on the ID of an image path. This means that whenever new data is added to the collection in Firestore, the client will receive updates without having to continuously check if the detection process is finished. As soon as the detection is completed, the client will be notified, and the new image with the bounding box and confidence detection score can be downloaded. This architecture enables real-time updates and efficient communication between the cloud and the mobile client.

The combination of Cloud Storage, Cloud Functions, and Firestore allows for a seamless flow of data and triggers. When an image is uploaded to Cloud Storage, it triggers the Cloud Function, which initiates the detection process. The response from ML Engine is then received, and Firestore is used to provide real-time updates to the mobile client. This tight loop ensures that the communication is quick and efficient, while also maintaining a thin client.

Firebase offers the advantage of real-time updates whenever new data is added to a specific path. This feature is particularly useful in this scenario, where real-time updates are crucial for the mobile app.

The process involved in building a custom machine learning model for object detection on iOS using Google Cloud Machine Learning and TensorFlow includes collecting and annotating data, training and serving the model at scale, and integrating it with the mobile client. Various tools and technologies are used, such as Cloud Storage, Cloud Functions, TensorFlow Object Detection, Cloud Machine Learning Engine, Swift SDKs, and Firebase SDK for Cloud Functions. The challenge lies in organizing and arranging the data in Google Cloud Storage, while the most interesting part is tying together all the different tools and technologies.

The journey of building this app involves working with different programming languages, including Python, Node, Swift, JavaScript, and even some bash commands with gcloud.

The process also involves labeling the images manually, which can be a time-consuming task. However, the satisfaction comes from seeing the final result of the prediction with the bounding box appearing in real time on the iOS app.

It is worth mentioning that the code for this project has been open-sourced on the creator's personal GitHub repository, allowing others to recreate and extend the model with their own data or client app.

This didactic material provides an overview of the process of building a custom machine learning model for object detection on iOS using Google Cloud Machine Learning and TensorFlow. It highlights the use of various tools and technologies, the challenges faced, and the satisfaction of seeing the final result. The material also emphasizes the availability of the code on GitHub for others to explore and build upon.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE TOOLS FOR MACHINE LEARNING
TOPIC: VISUALIZING DATA WITH FACETS

Data cleaning is an essential step in preparing datasets for machine learning. Messy data, with unbalanced or mislabeled values, can negatively impact the accuracy and reliability of machine learning models. In order to effectively clean and understand your data, Google Research has developed an open source project called Facets.

Facets is a tool that allows users to visualize and analyze their data in a generalizable manner. By using Facets, you can gain insights into how your dataset is structured and identify areas that require cleaning. The tool provides two main components: Facets Overview and Facets Deep Dive.

Facets Overview provides an overview of your dataset, splitting the columns into rows of salient information. This includes statistics such as the percentage of missing values, minimum and maximum values, as well as mean, median, and standard deviation. Additionally, it displays the percentage of values that are zeros, which can be helpful in identifying cases where most of the values are zeros. Facets Overview also allows you to compare the distributions of your test and training datasets for each feature, ensuring that they have similar distributions.

Facets Deep Dive offers a more detailed analysis of your dataset, allowing you to zoom in and examine individual data points. You can facet the data by row and column across any feature, similar to filtering by different categories when shopping online. The interface is divided into four main sections: a zoomable display of your data, a panel for changing the arrangement of your data, a legend for the display, and a detailed view of a specific row of data. By clicking on any data point in the center visualization, you can view a detailed readout of that particular data point.

To demonstrate the capabilities of Facets, let's consider an example using the "Census Dataset" from the 1994 US census. The dataset aims to predict whether a household's annual income is above or below $50,000 based on census statistics. Using Facets, we can split the data by age range and color the data points based on the target value. By faceting columns by hours per week, we can observe how different age ranges yield different results. For instance, we can see that the age 17-26 population has a larger chunk of individuals working 15 to 29 hours per week, which may be due to summer jobs. Additionally, we can observe that fewer people work 29 to 43 hours per week as they get older, while the 43 to 57 hour segment remains relatively constant.

To gain a more detailed look, we can change the positioning of the plot to "Scatter" and facet only by age. This allows us to easily see the working hours across different age groups. By doing so, we can observe that the hours per week rise in the middle of the chart and decrease on either side.

Facets is a powerful tool for visualizing and analyzing data in machine learning. It helps users understand the structure of their datasets, identify areas that require cleaning, and gain insights into relationships and trends within the data. By utilizing Facets, data scientists and machine learning practitioners can ensure the accuracy and reliability of their models.

Facets is a powerful tool for visualizing data in machine learning. It allows users to gain insights into their datasets by examining the relationships between different features and identifying any missing or unexpected values.

To load your dataset into Facets, you have two options. The first option is to use the web interface, where you can simply upload your data and explore it directly in your browser. The second option is to install the Facets library as a Jupyter notebook extension. You can find the instructions for installation on the project's GitHub page.

By visualizing your data with Facets, you can identify patterns and trends that may not be apparent from just looking at the raw data. This can help you make informed decisions when building machine learning models and improve their performance.

If you notice that your dataset is imbalanced, meaning that certain classes or categories have more data points

than others, Facets can highlight this imbalance. This information can guide you in collecting more data points to create a more balanced dataset.

Facets is a valuable tool for exploring and understanding your data in machine learning. By using its visualizations, you can uncover insights, identify issues, and make data-driven decisions to enhance your machine learning models.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE TOOLS FOR MACHINE LEARNING
TOPIC: GOOGLE QUICK DRAW - DOODLE DATASET

A team at Google has created the world's largest doodling dataset and a powerful machine learning model by turning the game of Pictionary into an interesting experiment. The game, called "Quick, Draw!", was initially featured at Google I/O in 2016. It involved one player drawing an object while the other player tried to guess what it was. In 2017, the team took this game a step further by using the Sketch-RNN model from the Magenta team at Google Research to predict what the player was drawing in real-time, eliminating the need for a second player.

"Quick, Draw!" is now available online and has collected millions of hand-drawn doodles. The dataset contains a wide variety of drawings from different players all over the world. It is a lot of fun to browse through the dataset and explore the different drawings. If you come across something that seems out of place, you can fix it right there on the page and make the data better for everyone.

The team has open-sourced the dataset, making it available in a variety of formats. The raw files are stored in .ndjson format and contain timing information for every stroke of every picture drawn. There is also a simplified version of the data available in the same format, which has undergone some pre-processing to help normalize the data. This simplified version is also available as a binary format for more efficient storage and transfer. Additionally, the simplified data can be rendered as a 28-by-28 grayscale bitmap in NumPy .npy format, which is compatible with existing code for processing MNIST data.

If you are interested in working with this dataset and training your own recurrent neural net, there is a material linked in the description below the material. You can also explore the dataset further by visualizing it using Facets. The Facets team has hosted the data online and provided presets for different views, allowing you to see how different players drew specific objects and which drawings were recognized as the intended objects.

Finally, you can extend the dataset by creating your own custom image class. Whether it's your company logo, school emblem, or something else entirely, you can train your model to recognize not only the existing classes in the dataset but also your own custom class.

Explore the "Quick, Draw!" dataset and see what insights you can gain from it.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE TOOLS FOR MACHINE LEARNING
TOPIC: GOOGLE MACHINE LEARNING OVERVIEW

Machine learning is a powerful tool used to solve complex problems by programming with data. In this session, we will explore the seven steps involved in the machine learning workflow and how Google Cloud Machine Learning tools can be utilized.

The first step in machine learning is gathering data. Without data, a model cannot be trained. It is important to identify where the data is located, in what format, and how to access it. Sometimes, the required data may not be readily available, and in such cases, data collection systems need to be developed. An interesting example of this is Quickdraw, an online game that collects hand-drawn images from users around the world. This dataset is open source and can be used for various machine learning tasks.

Once the data is gathered, it needs to be prepared before it can be used for machine learning. Data preparation involves exploring the data to gain a better understanding of its characteristics. This step helps identify any gaps or inconsistencies in the data and can save time and effort in the later stages of the process. Tools like Facets, an open-source data visualization tool from Google Research, can be used to analyze the data distribution and gain insights into the dataset.

After data preparation, the next step is to choose a model and train it using the prepared data. The model is trained to perform a specific task based on the available data. Evaluating the performance of the trained model is crucial to ensure its effectiveness. Fine-tuning can be done to improve the model's performance, and once satisfied with the results, the model can be used to make predictions or perform the desired tasks.

To illustrate these steps, let's consider a simple example of categorizing animals based on given data. The model will take in structured data about an animal and predict its type, such as bird, mammal, reptile, fish, amphibian, bug, or invertebrate. Each step of the machine learning workflow will be demonstrated using this example.

Machine learning is a process that involves gathering, preparing, and utilizing data to train models for solving various problems. Google Cloud Machine Learning provides a range of tools and resources to facilitate this process. By following the steps outlined in this session, you can effectively apply machine learning techniques to address your own problems.

In machine learning, one of the crucial steps is choosing the right model for the task at hand. Model selection involves finding the model that best fits the given data set and problem. Different models can handle different types and complexities of data. In this didactic material, we will explore how Google Cloud Machine Learning tools, specifically TensorFlow, can be used for model selection.

Before diving into model selection, it is important to understand how a machine learning model works. A model attempts to divide a region based on the given data points. For example, it may try to separate blue and orange dots by drawing an enclosing circle around the blue dots. The goal is to find a model that accurately represents the underlying patterns in the data.

To illustrate the process of model selection, we will use an example of a linear classifier. A linear classifier draws a straight line between two spaces. This is a simple model that can be implemented using TensorFlow. However, if the data is more complex, we may need to use a more sophisticated model, such as a deep neural network.

To switch from a linear classifier to a deep neural network in TensorFlow, we only need to make a few adjustments. One of these adjustments is adding an argument called "hidden units" to define the structure of the network. For example, we can have three layers with 30, 20, and 10 neurons respectively. The ease of making such changes allows us to experiment with different models without starting from scratch.

Once we have chosen a model, the next step is training. Training involves passing the data to the model and updating it based on the predictions it makes. Initially, the model's predictions will be inaccurate due to random initialization. However, as the model is updated with each piece of data in the training set, it improves over time. In practice, training a model in TensorFlow is as simple as calling the "train" function on the model object.

After the training process is complete, we move on to evaluation. Evaluation is done to assess the accuracy of the model in performing its primary task. By showing the model examples with known correct answers, we can determine how well it performs. Evaluation in TensorFlow involves using the "evaluate" function on the trained model.

Model selection is a crucial step in machine learning. Google Cloud Machine Learning tools, such as TensorFlow, provide a user-friendly environment for experimenting with different models. By choosing the right model and training it on the data, we can create accurate and effective machine learning models.

During the evaluation phase of machine learning, we use a separate set of data called evaluation data to measure the performance of our model. This evaluation data is not used to update the model, but rather reserved for evaluating its performance. In code, this is done by calling the dot evaluate function and passing in the evaluation data. It's important to note that using the training data for evaluation would be a mistake, as the model has already optimized itself for this data. Evaluating using the same data would misrepresent the true performance of the model and likely result in poor performance on real-world data.

To improve our model, we can utilize hyper-parameter tuning. Hyper-parameters are parameters of the model itself, and we can experiment with different values to find the best combination. This process involves training multiple variants of the same underlying model with different parameters and evaluating their performance. By comparing the accuracy of these variants, we can inform our parameter choice and improve the model. Hyper-parameter tuning is still an active area of research, but it can be effectively used to optimize the performance of machine learning models.

Once we have trained and fine-tuned our model, the final step is to make predictions. This is the ultimate goal of training the model - to use it to make useful predictions on new, unseen data. We can deploy the model and show it data that it hasn't been trained on, and it will provide predictions based on its learned patterns.

In practice, there are various tools and platforms available to assist in the machine learning process. One such tool is Co Lab, which is a notebook environment that allows running Python code in the browser. It provides a convenient way to write and execute code, as well as share it with others. Co Lab is hosted on Google Drive and supports features like markdown and code execution.

The machine learning process involves gathering and preparing data, choosing and training a model, evaluating its performance, tuning hyper-parameters, and finally making predictions. Evaluation data is used to measure the performance of the model, while hyper-parameter tuning helps optimize its parameters. The ultimate goal is to make accurate predictions on new data. Tools like Co Lab provide a convenient environment for executing machine learning code and experimenting with different models and parameters.

Google Cloud Machine Learning is a powerful tool for implementing machine learning models. It provides various tools and services that enable users to train and deploy models in the cloud. In this overview, we will explore some of the key features and functionalities of Google Cloud Machine Learning.

One of the main advantages of Google Cloud Machine Learning is its seamless integration with other Google tools and services. For example, it can connect to the internet and authenticate with Google Drive, allowing users to easily access and download datasets. This eliminates the need to set up additional machines in the background.

Google Cloud Machine Learning also offers collaborative features, such as the ability to share work with others and engage in collaborative research. Users can add comments and collaborate on projects, making it a convenient platform for teamwork.

When working with datasets, it is important to follow best practices. This includes shuffling and splitting the data into training and evaluation sets. By using a fraction of the dataset for evaluation, users can assess the performance of their models accurately. The ratio of the split can be adjusted based on specific requirements.

Preprocessing the data is another crucial step in machine learning. In this case, the last column of the dataset represents the label, indicating the correct type of animal. However, the labels are indexed from one to seven, which needs to be adjusted to zero through six. This simple preprocessing step involves shifting the labels by

subtracting one.

To feed the data into the training process, an input function is used. This function takes the raw data and can shuffle and batch it as needed. It ensures that the data is properly prepared for training the model.

Google Cloud Machine Learning utilizes the concept of feature columns to represent incoming data. These feature columns allow users to customize the model for their specific dataset. In this case, all 17 columns of the dataset are set as numeric, indicating that they contain numerical data.

Creating a model in Google Cloud Machine Learning involves specifying the type of model and the number of classes. In this example, a linear classifier is created with seven different classes, corresponding to the different types of animals. The training and evaluation processes are combined into a single function for convenience and to avoid introducing bugs.

After running the model, the evaluation metric is displayed. In this case, the model achieved an accuracy of 90%, which is reasonable considering the small size of the dataset. However, it is important to note that this metric may not be reliable due to the limited number of samples.

Google Cloud Machine Learning provides a comprehensive set of tools and services for implementing machine learning models. Its integration with other Google tools, collaborative features, and support for preprocessing and training make it a powerful platform for machine learning projects.

Artificial Intelligence (AI) has revolutionized various industries, including machine learning. Google Cloud offers powerful tools for machine learning, making it easier for developers and data scientists to build and deploy AI models. In this didactic material, we will provide an overview of Google's machine learning tools and discuss how they can be used.

One of the key tools offered by Google is Google Cloud Machine Learning. This tool allows users to build and train machine learning models using TensorFlow, an open-source library for numerical computation and machine learning. TensorFlow provides a flexible and efficient framework for building deep neural networks, which are widely used in AI applications.

To demonstrate the use of Google Cloud Machine Learning, let's consider an example. In this example, we have a dataset and we want to train a deep neural network model to make predictions based on this data. The first step is to preprocess the data and convert it into a format that can be used by the deep neural network. We can achieve this by wrapping the existing feature columns in an indicator column, which allows the deep neural network to represent the data effectively.

Once the data is preprocessed, we can proceed to train the deep neural network model. TensorFlow provides functions such as `train` and `evaluate` to facilitate this process. After training the model, we can evaluate its performance using evaluation data. In this case, the evaluation data consists of 40 values, and the model's performance is measured by the percentage of correct predictions. It is important to note that the evaluation results may vary due to the limited size of the evaluation data.

To further analyze the model's performance, we can make predictions on specific examples from the evaluation data. TensorFlow provides a `predict` function for this purpose. By passing in the evaluation data and selecting a subset of examples, we can obtain the model's predictions and compare them with the correct answers. This allows us to identify the examples that were incorrectly predicted by the model.

In addition to Google Cloud Machine Learning, Google offers another tool called Kaggle. Kaggle is a platform known for its competitions, discussion forums, and data sets. It also provides a feature called Kernels, which is essentially a notebook for running machine learning code. Kernels in Kaggle are similar to notebooks in Google Cloud Machine Learning, but with some subtle differences.

In Kaggle Kernels, users can upload and run notebooks written in Python or R. The data sets used in Kaggle Kernels are stored in a directory called "input." Users can download and upload notebooks, making it easy to share and collaborate on machine learning projects. Kaggle Kernels also allows users to commit their notebooks, which generates a view-only version with all the outputs, ensuring reproducibility and easy sharing.

Google Cloud offers powerful tools for machine learning, including Google Cloud Machine Learning and Kaggle Kernels. These tools provide a user-friendly environment for building, training, and deploying machine learning models. With these tools, developers and data scientists can harness the power of AI and unlock new possibilities in various industries.

Google Cloud Machine Learning provides a range of tools and services for implementing machine learning models and deploying them at scale. One such tool is Kaggle Kernels, which allows users to create and run Jupyter notebooks in the cloud. Notebooks can be versioned and shared with collaborators, making it easy to reproduce and iterate on your work. Additionally, Kaggle Kernels supports GPU acceleration, enabling faster computation for deep learning models.

If you have larger workloads or need to run long-running jobs, Google Cloud Machine Learning Engine is a powerful option. This service allows you to kick off jobs that can run for hours across distributed machines, potentially with GPUs attached. Once your model is trained, you can use the auto-scaling prediction service to serve predictions at scale. This service is easy to use and supports TensorFlow models as well as other popular machine learning libraries like scikit-learn and XGBoost.

For those looking for pre-trained models, Google Cloud Machine Learning offers a range of APIs that can perform tasks such as image recognition and speech-to-text conversion. These APIs are easy to use and require minimal setup, making them ideal for applications that require quick and accurate results. However, customization options are limited, and you may need to wait for future updates to tailor the APIs to your specific use case.

If you want to dive deeper into machine learning concepts, Google offers a machine learning crash course that covers a wide range of topics. This course includes materials and assignments to help you build your machine learning knowledge from the ground up. Additionally, if you are interested in using Google Cloud for machine learning, you can visit the Cloud dome or the website cloud.Google.com/machinelearning for more information.

Google Cloud Machine Learning provides a comprehensive set of tools and services for implementing and deploying machine learning models. Whether you prefer working with notebooks, need to run long-running jobs, or want to leverage pre-trained models, Google Cloud has you covered.

Welcome to AI Adventures, a series dedicated to exploring various aspects of machine learning. In each episode, we delve into interesting topics and provide hands-on demonstrations to enhance your understanding. Additionally, we conduct interviews with experts in the field. We encourage you to subscribe to our series and stay updated with the latest content.

In this session, we will be discussing Google Cloud Machine Learning and the tools it offers for machine learning tasks. Google provides a comprehensive set of tools and services that facilitate the development and deployment of machine learning models.

Google Cloud Machine Learning allows you to build and train your own custom models using popular frameworks such as TensorFlow and scikit-learn. These frameworks provide a rich set of functionalities for creating and training models on large datasets.

One of the key tools offered by Google is the Cloud Machine Learning Engine. This tool enables you to easily deploy and manage your machine learning models on the cloud. It provides a scalable and reliable infrastructure for running your models in production, allowing you to serve predictions at scale.

Another important tool is the Cloud AutoML, which simplifies the process of building custom machine learning models. AutoML provides a user-friendly interface that automates many of the complex tasks involved in model development, such as feature engineering and hyperparameter tuning.

Google also offers pre-trained machine learning models through the Cloud AI Platform. These models are trained on vast amounts of data and can be used for various tasks, such as image and speech recognition, natural language processing, and sentiment analysis. Leveraging these pre-trained models can save you time and resources in developing your own models from scratch.

To further enhance your machine learning workflow, Google provides tools for data preparation and exploration. For example, BigQuery allows you to analyze and query large datasets efficiently, while Dataflow enables you to

process and transform data in real-time.

Google Cloud Machine Learning offers a comprehensive suite of tools and services that facilitate the development, deployment, and management of machine learning models. Whether you are a beginner or an experienced practitioner, these tools can greatly enhance your productivity and enable you to leverage the power of machine learning in your applications.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: GCP BIGQUERY AND OPEN DATASETS

Large data sets can present challenges for data scientists, from storage options to running analytics tools efficiently. However, Google Cloud offers a solution with BigQuery public datasets. These datasets are hosted by Google and can be easily accessed and integrated into applications. The best part is that Google covers the storage costs, and users only pay for the queries they perform on the data. Additionally, there is a one-terabyte per month free tier, making it accessible for beginners.

Currently, there are nearly 40 public datasets available on BigQuery. Each dataset contains multiple tables, and they are being utilized by numerous projects worldwide. What makes these datasets even more useful is the accompanying explanatory text, which helps users understand the data structure and get started with querying. One example is the New York City tree census dataset, which allows users to answer questions about tree species in the city and how they have changed over time.

Another remarkable dataset is the Open Images dataset, which includes approximately nine million URLs of annotated images across 6,000 categories. This dataset enables users to find answers to various image-related questions, such as the number of images featuring a specific object.

By combining BigQuery public datasets with tools like Data Lab, Facets, and TensorFlow, users can enhance their data analysis skills and perform impactful data science. To get started, users can visit the Public Datasets page on BigQuery and explore the available datasets.

Experience the power of BigQuery's public datasets and unleash your data analysis potential. Start querying away and discover the vast possibilities of open data.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: DATA SCIENCE PROJECT WITH KAGGLE

Kaggle is a platform that aims to be the best place for data scientists to find, share, and analyze data. It provides tools for building data science portfolios, conducting data analysis work, and sharing research. With over 1.7 million data scientists in its community, Kaggle offers a one-stop shop for working with data.

Kaggle has recently introduced new features that enhance its functionality. One such feature is the ability to publish and work with private datasets and kernels. Kernels are like laptops in the cloud, offering powerful computing resources such as 16GB of RAM, four CPUs, and six hours of compute time. These kernels come pre-installed with popular data science packages, providing a convenient one-click environment. Additionally, users can customize their kernels by installing additional packages or even adding a GPU.

One of the advantages of Kaggle is its support for collaboration. Data scientists can work together on datasets and kernels, making it easier to collaborate on data science projects. Kaggle also supports private mode, allowing users to work on their projects privately before sharing them with the world.

In this material we explore a dataset from the city of Los Angeles. The dataset contains information about environmental health code violations from restaurants and markets in Los Angeles. We downloaded the dataset to a local machine and will upload it to Kaggle to serve as the foundation of their project.

Some people may have concerns about distributed computing and the need for massive compute resources. However, Kaggle's vibrant community and powerful computing resources demonstrate that it can support a wide range of use cases, even with just one powerful machine.

By working together and leveraging the tools and resources provided by Kaggle, Yufeng and Megan aim to create a collaborative project that includes a public dataset and notebook with insightful analysis. This project will showcase the capabilities of Kaggle and provide valuable insights into the environmental health code violations in Los Angeles.

Artificial Intelligence (AI) is revolutionizing various industries, and one area where it has made significant advancements is in machine learning. In this educational material, we will explore how to create a data science project using Google Cloud Machine Learning and Kaggle.

Kaggle is a platform where users can upload and access thousands of datasets. To create a new dataset on Kaggle, we start by visiting the datasets page on Kaggle's website. Here, we have access to all publicly published datasets. To add our own dataset, we click on "New Dataset" and then proceed to drag and drop the files we want to upload. In this example, the files contain inspections of restaurants and markets in Los Angeles, as well as violations.

After uploading the files, we need to provide some metadata for the dataset. We choose to keep it private initially to ensure proper documentation and exploration before sharing it publicly. Documentation for datasets is crucial to make data accessible and understandable. It goes beyond just making the data machine-readable; it involves providing context, details about the dataset's contents, and possible questions the data can answer.

Once we click on "Create Dataset," our private dataset is successfully created. We can now begin analyzing the dataset and adding collaborators. Kaggle provides a quality checklist to help document the dataset effectively, ensuring its success when shared. The checklist includes providing a description, which explains the dataset's context and contents. Additionally, we can add tags to make the dataset more discoverable, such as "Public Health" and "Food and Drink." A subtitle and banner image can also be added to enhance the dataset's appearance and understanding.

Collaborators play a vital role in dataset development. In this example, the collaborator is granted edit access to the dataset, allowing them to view and make changes. Once the dataset is ready, it can be shared with the community, and other users can view it on Kaggle.

Creating a data science project with Kaggle and Google Cloud Machine Learning involves uploading datasets,

adding metadata, documenting the dataset effectively, and collaborating with others. By following these steps, we can contribute to the growing field of AI and machine learning.

In this didactic material, we will explore how to document datasets through code using Kaggle. Documenting datasets through code allows users to demonstrate what can be done with the data and share it with the community. We will specifically focus on publishing a kernel on a dataset to showcase its potential.

When working with a new dataset, it is common to start with a blank slate. By publishing a kernel on Kaggle, we can provide code that shows how to read in the data and work with it. This is especially useful for others who want to explore the dataset and understand its characteristics.

To get started, we will click on the "New Kernel" button, which gives us the option to create a script or a notebook. We will choose notebooks because it allows us to seamlessly mix markdown and code. Once the kernel is created, we will have access to the dataset in our environment.

Next, we will select the language we prefer for our analysis. In this case, the speaker prefers R, so they change the language to R. They have already prepared the code for analysis and will upload it to the kernel. They will then walk us through the code and explain their analysis.

In the first cell of the code, the speaker reads in the inspections CSV file and the violations CSV file. They join the two datasets together based on a common identifier and provide a glimpse of the resulting data frame. The dataset contains almost 900,000 health code violations spanning two years.

The analysis focuses on the number of violations reported over time, specifically by month. They generate a visualization that shows the trends in violations over time. The visualization reveals a significant number of health code violations, with some months having over 30,000 violations.

To further understand the dataset, we also explore the top 10 violations. They color-code the violations to indicate their frequency, with lighter colors representing more violations. For example, they highlight a violation related to the code for floors, walls, and ceilings being properly built, maintained, in good repair, and clean.

Finally, there is a future project idea to analyze violations by zip code. We have information about the facilities' addresses and plan to perform a geospatial analysis to uncover any patterns in violations across different areas. We save the relevant data to a CSV file for future use.

By publishing a kernel on Kaggle and documenting the dataset through code, we have not only provided insights into the dataset but also inspired new questions and potential analyses.

In this didactic material, we will discuss the process of creating and sharing a data science project using Kaggle, focusing on the topic of Artificial Intelligence and Google Cloud Machine Learning.

Once you have completed your work on a notebook in Kaggle, it is important to save and share it with others. To save your work, you need to click on the "Commit and Run" button. This action not only saves your code but also executes it from top to bottom. It is worth noting that if you close the tab before clicking "Commit and Run," your work will be saved as a draft.

After saving your notebook, you can view it by clicking on "View Snapshot." This will take you to the Notebook Viewer, where you can share your work with others. The Notebook Viewer allows people to access and explore your dataset. Sharing your work in a team environment can be useful for code review scenarios.

To share your notebook with others, you can provide them with the link to the Notebook Viewer. They can access it by clicking on "Kernels" in the dataset and selecting your work. From there, they have the option to either edit or fork your notebook. Forking a notebook is similar to forking a repository on GitHub. It creates a copy of the notebook, including the code, data, and environment used.

When you fork a notebook, you have your own copy that you can modify without affecting the original. You can make changes to your forked notebook and run it to ensure everything works as expected. It is also possible to share your forked notebook with others.

If you choose to make your work public, you can do so by going to the dataset settings and clicking on "Make Public." This action will make your dataset accessible to the community of data scientists on Kaggle. Additionally, you can make your kernel public separately from the dataset. This allows for flexibility in collaborating with others and releasing different versions of your work.

Kaggle provides a seamless platform for creating, sharing, and collaborating on data science projects. By following the steps outlined in this didactic material, you can save, share, and make your work public on Kaggle, enabling others to explore and build upon your analysis.

Artificial Intelligence (AI) has revolutionized various industries, including data science. In this didactic material, we will explore the topic of advancing in machine learning through a data science project with Kaggle, specifically focusing on Google Cloud Machine Learning.

In this project, the goal was to create a reproducible and documented dataset that can be shared with the world. The speakers discuss how they obtained data files from a local machine and transformed them into a publicly accessible dataset. This dataset can be used for various purposes, such as school projects or sharing research.

What should be emphasized is the collaborative nature of the project, highlighting how viewers can access the dataset and interact with it on Kaggle. They provide links to the notebooks in the material description, allowing users to explore the dataset, post comments, make edits, and even create their own notebooks based on the project.

This didactic material introduces the concept of advancing in machine learning through a data science project with Kaggle. It highlights the process of creating a reproducible and publicly shared dataset using Google Cloud Machine Learning. The material encourages participants to engage with the project and provides resources for further exploration.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: AUTOML VISION - PART 1

AutoML Vision is a powerful tool provided by Google Cloud Machine Learning that allows users to build and deploy custom machine learning models. In this two-part material, we will explore how to use AutoML Vision to create a model that can recognize different types of chairs.

To begin, we need to collect a large amount of labeled photos. Ideally, we should have hundreds of pictures per object. These photos will serve as the training data for our model. To make the data collection process easier, we can use tools like ffmpeg to extract frames from videos. By capturing materials of chairs and then extracting frames, we can quickly gather a large number of labeled images.

Once we have our labeled images, we can upload them to Google Cloud Storage using gsutil. It is recommended to replicate the folder structure of our data, with each folder representing a different label. This organization method simplifies the management of our images and data.

Next, we need to create a CSV file that lists the path and label for each image in our dataset. There are multiple ways to accomplish this, but in this example, we will use a Jupyter notebook and the Pandas library. We can create a dictionary that maps each folder name to an array of file names. By zipping the folder names and file names together, we can create a dictionary that represents the full path and label for each image. Finally, we can export this dictionary as a CSV file.

In part 2 of this material, we will use the data we have prepared to build a model that can detect the style of chair in a picture. Stay tuned for the next part of this series to learn more about AutoML Vision and its capabilities.

To prepare our data for training a machine learning model, we need to create a data frame that contains the file paths and corresponding labels. We start by looping over the dictionary that holds the file names and their labels. For each file, we combine the base Google Cloud Storage path, folder name, file name, and label. This results in an array of pairs, where each pair represents a file path and its label.

Next, we pass this array directly into a data frame using the Pandas library. Pandas accepts this format and allows us to create new data frames from scratch. The resulting data frame contains all the file paths and labels in the expected format.

Before proceeding, it is important to export the data frame to a CSV file. In this case, we want to export the data without including the index or header. If we leave the defaults, the outputted CSV file will have a header row and an index column that we don't need. To avoid this, we set the index parameter to false and the header parameter to false.

If your folder structure is similar to mine, you can adapt this notebook for your own use case with minimal changes. I will publish this notebook and provide a link in the description below the material.

Now that we have a CSV file that describes the location and label for all the images in our dataset, we are ready to train our model. Join me in part 2 of this material, where we will see how well our model performs.

Thank you for watching this material on Cloud AI Adventures. If you found it helpful, please like and subscribe to stay updated with the latest content. While you proceed to part two of this material, I'll enjoy this delicious apple.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: AUTOML VISION - PART 2

AutoML Vision is a powerful tool offered by Google Cloud Machine Learning that allows users to train and deploy machine learning models for computer vision tasks without the need for writing code. In this episode, we will focus on the second part of using AutoML Vision.

In the first part, we discussed the purpose of AutoML Vision and the requirements for the training data. Now, let's explore how to train our model using the gathered data. The process is straightforward: simply click on the "Train" button. It is recommended to start with the simple model option and evaluate its performance. Training may take some time, so you can take a break and return later.

Once the training is completed, you will receive various statistics about your model's performance. These statistics can help you identify mislabeled images or other aspects that need correction. You can then click on the "Retrain" button to address these issues. In this particular case, the model achieved high metrics due to the specific and clean data used for training.

However, the true test of a machine learning model lies in its performance on new, unseen data. The presenter challenged the model by presenting it with various images. The model correctly recognized the primary objects in the images but also identified other objects present in the background. It performed well overall, but some deviations were observed, such as the unexpected identification of a yellow chair in a photo of mostly blue chairs.

Despite these minor imperfections, the top option of this model proved to be quite good. It is worth noting that at this stage, the model is already available for use through its REST API. The training process also deploys the model, leveraging Cloud ML Engine's online prediction capabilities. This provides a customized prediction service for the model trained on our dataset.

One of the advantages of using AutoML Vision is its hands-free approach to training and deploying machine learning models. Once you have a well-formed data pipeline, the process becomes as simple as clicking a few buttons and waiting. This allows users to focus on preparing their data and offload the challenges of constructing a suitable computer vision machine learning model to the service.

AutoML Vision is a user-friendly tool that enables users to train and deploy machine learning models for computer vision tasks without the need for coding. By following a few simple steps, users can train their models, evaluate their performance, and leverage the power of online prediction capabilities. This tool simplifies the process of creating computer vision models, allowing users to concentrate on their data and its preparation.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: SCIKIT-LEARN

Scikit-learn is a widely-used machine learning library that provides a rich suite of tools for various aspects of machine learning. It was originally called scikits.learn and started as a Google Summer of Code project. The name "scikit" comes from it being a sciPy Toolkit. Over time, scikit-learn has gained popularity and is now a well-documented and well-loved Python machine learning library.

One of the remarkable features of scikit-learn is its extensive collection of machine learning algorithms. These algorithms cover a wide range of models, and most of them can be easily tried out with minimal code adjustments. This makes scikit-learn an excellent tool for understanding different types of models and gaining intuition about their performance with different parameter settings.

Scikit-learn offers a comprehensive set of tools for tasks that are "around" machine learning. This includes dataset loading and manipulation, preprocessing pipelines, and metrics. By using scikit-learn, you can conveniently handle these tasks within a single library.

To demonstrate scikit-learn in action, let's consider a simple example using a Kaggle kernel. In this example, we have a dataset of zoo animals, where the task is to classify each animal into one of seven different classes. We can load the dataset using pandas, and the class type field is the column we are interested in predicting.

In the past, we would shuffle and split the data manually using pandas. However, scikit-learn provides a function called train_test_split, which consolidates these tasks into one. By calling this function, we can easily create training and test data for our features and labels.

In this example, we will use a Support Vector Classifier (SVC) from scikit-learn. However, you can easily swap out this line of code for a different machine learning algorithm. After fitting the model using the fit method, we can evaluate its performance using the score method. Finally, we can make predictions using the predict method.

Scikit-learn has an API that closely aligns with the conceptual workflow of machine learning, making it easy to use and integrate into existing work. By exploring the materials and documentation, you can delve deeper into scikit-learn and create even more impressive models.

In the next material, we will discuss the other side of machine learning with scikit-learn, focusing on making predictions and scaling up the models.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: SCIKIT-LEARN MODELS AT SCALE

Welcome to this educational material on deploying scikit-learn models on Google Cloud ML Engine. In this material, we will guide you through the process of training a scikit-learn model and deploying it on Google Cloud ML Engine.

To begin, let's assume you have trained a scikit-learn model and now want to set up a prediction server. First, you need to export the model using the joblib library from sklearn.externals. The joblib.dump function can be used to export the model to a file.

Once the model is exported, you can retrieve the output from your Kaggle kernel and download it. With the trained scikit-learn model in hand, you are now ready to head over to Google Cloud Machine Learning Engine and load up the model to serve predictions.

Cloud ML Engine traditionally worked only for TensorFlow models, but now it also supports scikit-learn models as well as XGBoost. This means you can easily transition between scikit-learn, TensorFlow, and XGBoost without reworking your plumbing or architecture.

The first step in getting your model on the cloud is to upload the joblib file to Google Cloud Storage. It is important to name the file as "model.joblib" and place it inside a folder with a name you'll remember. This will help you locate the model later.

Next, you need to create your model and version on Cloud ML Engine. Make sure to specify that you are loading a scikit-learn model and select the appropriate runtime version of Cloud ML Engine and Python version used to export the model.

Once the model is uploaded and set up, you have a scikit-learn model serving in the cloud. However, a scalable model is of no use without the ability to call predictions. To do this, you can present the data to Cloud ML Engine as a simple array encoded in a JSON file.

In this material, we have provided an example of how to call predictions using a sample row of data. By following the steps outlined in this material, you can deploy your scikit-learn model to production and even automate the deployment process for future models.

You can now head over to Cloud ML Engine and start deploying your scikit-learn models for auto scaling predictions.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: INTRODUCTION TO KERAS

Keras is a powerful tool for creating machine learning models. In this didactic material, we will explore what Keras is and how to use it to get started with machine learning.

Keras is a high-level neural networks API written in Python. It is designed to be user-friendly, modular, and extensible, allowing users to build and experiment with different machine learning models easily. Keras is built on top of TensorFlow, a popular open-source machine learning framework.

To get started with Keras, you can use it directly within TensorFlow via the `tensorflow.keras` package. Alternatively, Keras also exists as a standalone library, but the TensorFlow version has the same APIs and additional features.

To use Keras, you don't need to install or configure anything if you use a tool like Kaggle Kernels. Simply create a Kaggle account if needed, sign in, and you will have access to all that Keras has to offer.

In this example, we will demonstrate how to use Keras to perform a machine learning analysis on the Fashion-MNIST dataset. This dataset contains 10 different types of fashionable items, such as pants, shirts, shoes, and handbags, represented as 28x28 pixel grayscale images.

To begin, we import TensorFlow, numpy, pandas, and matplotlib, just as we would in any machine learning project. We then load the training and test datasets into pandas dataframes and examine their structure. The training dataset consists of 60,000 examples, each with 784 columns representing the grayscale pixel values. The labels are represented as numbers from zero to nine.

Before training our model, we preprocess the data. We divide the pixel values by 255 to normalize them between zero and one. We also perform one-hot encoding on the labels, converting them into arrays of length 10. Each array has all zeros except for a one at the index corresponding to the original label value. This encoding is necessary for training our model.

Now comes the exciting part - creating our model using Keras. We build a sequential model, which allows us to add layers on top of each other. The first layer has 30 nodes and uses the rectified linear unit (ReLU) activation function. We then add another fully-connected layer with 20 neurons, also using ReLU activation. Finally, we add a layer that maps the inputs to the 10 output values representing the possible labels (zero through nine). This layer uses the softmax activation function, which distributes probabilities across the 10 categories.

After defining our model, we compile it using the `compile` function in Keras. We specify the loss function, optimizer, and metrics we want to use. The loss function measures how well the model performs during training, the optimizer determines how the model is updated based on the data, and the metrics provide evaluation metrics to monitor during training.

By following these steps, we can create a machine learning model using Keras. This is just a basic introduction to Keras, and there are many more advanced concepts and techniques to explore. But with Keras, you have a powerful tool at your disposal for building and experimenting with machine learning models.

In this tutorial, we will introduce Keras, a powerful deep learning framework, and explore how to use it for training and evaluating machine learning models. Keras is particularly useful for its simplicity and ease of use.

To begin, we need to define the loss function that will measure the error or loss of our model. In this case, we are using categorical cross entropy since our outputs are categorical. Cross entropy is a suitable choice for measuring the loss in this scenario.

Once our model is created, we can proceed with training. Training with Keras is straightforward and can be done by calling the ".fit" function. The training data, consisting of the training features and labels, needs to be provided as input. Additionally, it is recommended to specify the number of epochs and the batch size to have more control over the training process.

In this example, we have set the number of epochs to two, meaning that the entire dataset will be iterated over twice. The batch size is set to 128, which means that during each training step, the model will process 128 examples to adjust its parameters.

During the training process, Keras provides helpful progress updates, displaying the loss and accuracy at the end of each epoch. However, evaluating the model's accuracy against a separate test dataset is more informative. To accomplish this, we use the "model.evaluate" function and pass in the test features and labels. This will give us an accuracy score that we can print and analyze.

In this case, the model achieved an accuracy of 84.7%. Of course, there is room for improvement by increasing the number of epochs, using a more sophisticated model, or trying alternative approaches. This material serves as an introduction to Keras, providing a solid starting point for further exploration.

Keras benefits from a vibrant community and numerous code samples. Combining Keras with Kaggle's community resources offers an extensive set of tools and knowledge to facilitate your journey into deep learning.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: SCALING UP KERAS WITH ESTIMATORS

Artificial Intelligence - Google Cloud Machine Learning - Advancing in Machine Learning - Scaling up Keras with estimators

In this educational material, we will explore how to convert a Keras model to a TensorFlow estimator, which allows for distributed training and scaling. By converting our Keras model to a TensorFlow estimator, we can overcome the challenges of scaling up to larger datasets and running across multiple machines. Additionally, this conversion makes it easier to perform model serving once training is complete.

To convert a Keras model to a TensorFlow estimator, we can use the function called model_to_estimator. This function provides interoperability between Keras and TensorFlow and comes with built-in distributed training capabilities. By utilizing this function, we can solve our scaling challenges and simplify the process of model serving.

In the provided Kaggle notebook, we start by loading the necessary libraries such as NumPy, pandas, TensorFlow, and plotlib. We then load our dataset using pandas and preprocess the data by extracting the features and labels. Afterward, we perform one-hot encoding on the labels to convert them into arrays of length 10, representing the different possible labels.

Next, we define our training parameters and create our Keras model. This model is defined using the same code as in previous materials. Once the model is created, we can train it using the model.fit function, as seen before.

Now, let's focus on the conversion process. To convert our Keras model to a TensorFlow estimator, we use the model_to_estimator function. This function takes the Keras model as an argument and converts it into a TensorFlow estimator. Once the conversion is complete, we can perform all the usual operations with a TensorFlow estimator.

To train our TensorFlow estimator, we need to architect our input function. One important consideration is naming the label for our features. In Keras, we don't have control over naming it ourselves. Therefore, we can use the model.input_names attribute to obtain the input name of the Keras model. In this case, it is called dense_3_input. We can use this name as the key for our features in the input function.

After setting up the input function, we can proceed with training the estimator. We define our feature columns and train the estimator using the correct input names obtained from the Keras model. It is crucial to use the right input names to avoid errors during training.

Once the training is complete, we can evaluate the performance of our estimator. To do this, we set up the evaluate_input function and use the input name obtained from the Keras model. It is worth noting that we need to use the one-hot encoded labels in both the TensorFlow training and evaluation, just like in Keras.

By following these steps, we can successfully convert a Keras model to a TensorFlow estimator, enabling us to scale up our machine learning tasks and perform model serving efficiently.

In order to scale up Keras with estimators in Google Cloud Machine Learning, it is important to use the same labels and variables as in Keras. This is convenient because it allows us to use the same arrays and variables that were used before. The accuracy of the models may vary depending on factors such as random initialization state and the performance of the computer.

Once the models are trained in both Keras and TensorFlow, the next step is to export these models into the file system. Exporting Keras models is straightforward, as we can simply call the "model.save" function and provide a name for the exported model. This will export the model in the HDF5 format.

On the other hand, exporting TensorFlow models requires the use of the "export_savedmodel" function. This function takes two arguments: the path to export the model to, and a serving input receiver function (also known as a serving function). TensorFlow provides a utility to help build this serving function, which includes a

mapping of the input shape and type to help the model know what kind of inputs to expect.

After running the "export_savedmodel" function on the TensorFlow model, we can see where it was exported to. TensorFlow exports models as a file and a folder called "variables". For convenience, it is recommended to zip or tar the exported model so that it is easy to download. Once downloaded, the model can be unzipped or untarred for further use.

By following these steps, we can take a Keras model, convert it to a TensorFlow estimator, and export it for future use. This process allows us to leverage the advantages of both Keras and TensorFlow, combining the easy model creation syntax of Keras with the distributed training capabilities of TensorFlow estimators.

Scaling up Keras with estimators in Google Cloud Machine Learning involves using the same labels and variables as in Keras, exporting the models using the appropriate functions in Keras and TensorFlow, and utilizing the exported models for further analysis and tasks.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: INTRODUCTION TO TENSORFLOW.JS

TensorFlow.js is a JavaScript library that allows you to run TensorFlow, an open-source machine learning platform, on the most widely used programming language in the world. With TensorFlow.js, you can perform various machine learning tasks directly in the browser. In this didactic material, we will explore the features and capabilities of TensorFlow.js.

To get started with TensorFlow.js, visit js.tensorflow.org. This website provides comprehensive documentation, tutorials, and links to examples that will help you develop your skills in TensorFlow.js. The materials are designed to build on each other, allowing you to gradually enhance your understanding and proficiency. We recommend starting at the top of the page and working your way down, completing each material one by one.

One of the exciting things you can do with TensorFlow.js is train a convolutional neural network in your browser. This allows you to work with deep learning models and datasets directly in the browser environment. Additionally, you can create interactive applications, such as a Pac-Man game, using TensorFlow.js. Furthermore, TensorFlow.js enables you to play with data recorded from your webcam, opening up new possibilities for experimentation and creativity.

TensorFlow.js also supports the import of TensorFlow and Keras models into the browser. This means that you can train complex models with large datasets, just as you would in traditional machine learning workflows, and then bring those models directly into the browser. In a future material of AI Adventures, we will explore an example of this process in more detail.

For those looking to further customize and specialize their models, TensorFlow.js allows you to take an imported model and continue training it. This means you can start with an existing state-of-the-art research model and fine-tune it using your own training data. For example, you could use images from your webcam to personalize the model's performance.

You may be wondering about the relationship between TensorFlow.js and deeplearn.js. TensorFlow.js is an ecosystem of JavaScript tools for machine learning, and it evolved from deeplearn.js. The core functionality of TensorFlow.js is derived from deeplearn.js, while additional features and tools have been added to make machine learning in JavaScript more accessible.

In terms of performance, TensorFlow.js with WebGL is approximately one and a half to two times slower than TensorFlow using Python with AVX enabled for predictions. On the training side, large models can be 10 to 15 times slower in the browser compared to TensorFlow on Python. While TensorFlow.js may not match the raw performance of server-side training, it offers exciting opportunities for client-side machine learning experiences.

This overview has provided a glimpse into the capabilities of TensorFlow.js. We encourage you to visit js.tensorflow.org to start exploring and experimenting with TensorFlow.js today. In future episodes, we will delve deeper into TensorFlow.js, covering additional tips and tricks to enhance your machine learning projects.

Thank you for watching this material of Cloud AI Adventures. If you found it helpful, please consider liking the material and subscribing to stay updated with the latest episodes. Get started with TensorFlow.js and unleash your creativity in machine learning!

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: IMPORTING KERAS MODEL INTO TENSORFLOW.JS

TensorFlow.js and Keras are both powerful tools in the field of artificial intelligence. And the good news is, they can work together seamlessly. In this educational material, we will explore the process of importing a trained Keras model into TensorFlow.js.

TensorFlow.js offers various functionalities, but one of the most exciting ones is its ability to collaborate with other tools. This allows machine learning tasks to be performed across different platforms, languages, and devices, all optimized for peak performance in their respective environments.

However, loading a Keras model into TensorFlow.js can be a bit tricky. Keras models are typically saved as HDF5 files using the `model.save('my_model.h5')` command. To overcome this challenge, TensorFlow.js provides a tool called the TensorFlow.js converter. This converter is used to convert your Keras model into a format that TensorFlow.js can consume, which is a folder containing a file called `model.json`.

To put this into context, let's consider the Keras model we have been using in previous materials. Suppose we have exported this model as a `.h5` file. To begin the process, we need to download the Keras model file, which we can do by accessing the Output section of the appropriate Kernel.

Once we have the Keras model file downloaded, we need to install the TensorFlow.js converter by running `pip install tensorflowjs` in our Python environment. It is important to note that the library name for the pip installation is `tensorflowjs`, despite the converter being called `tensorflowjs converter`.

Next, we create a new folder, let's call it `tfjs_files`, to store the output files of the TensorFlow.js model. This folder will serve as the destination for the converted files.

Now, we can run the converter, using a flag to indicate that it is a Keras model. Additionally, we need to provide the path to the Keras HDF5 file within the `tfjs_files` folder. This will ensure that the converter knows where to place the new files.

After running the converter, we can take a look inside the `tfjs_files` folder. Here, we will find the `model.json` file, as well as three additional files named `group1-shard1of1`, `group2-shard1of1`, and `group3-shard1of1`. These shards enable faster loading by the browser and are likely to be cached for subsequent calls when serving the files.

With the Keras model successfully converted to `model.json`, we are left with one final step - loading the model into JavaScript. Thankfully, this is a simple one-line task. We can use `tf.modelload` to point to the URL where the files are hosted, and TensorFlow.js will handle the rest.

It is worth noting that since the model operates on the client side using JSON files, it may not be suitable for scenarios requiring model security. In such cases, client-side models are recommended.

Now that you have learned how to import your own Keras models into TensorFlow.js, you can unleash your creativity and create amazing machine learning-inspired games on the web.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: DEEP LEARNING VM IMAGES

Deep Learning VM Images on Google Compute Engine provide an easy and efficient way to set up a new environment for machine learning. These VM images come preinstalled with popular deep learning and machine learning frameworks, such as TensorFlow and PyTorch, and offer support for Cloud TPU and GPU.

To get started with Deep Learning VM Images, you can either use the Google Cloud Platform Cloud Launcher UI or the command line. The images are available in the marketplace, where you can choose from a variety of configurations, including CPU, memory, storage, GPU, and installed libraries. This allows you to fine-tune the VM to meet your specific workload requirements.

One of the advantages of using VMs is the flexibility to edit the hardware configuration on the fly. Unlike a traditional computer, you can easily add more memory, increase disk space, or even add additional GPUs to your VM.

To create a VM using the gcloud command line tool, you need to specify the zone and the machine learning library you want to use. For example, if you want to use TensorFlow, you can choose the tf-latest-cpu image. Once you run the command, the machine will be provisioned and the necessary software packages will be installed. You can then SSH into the machine using a modified SSH command that enables you to connect to JupyterLab via your local browser.

JupyterLab is a powerful tool for data science workflows. With JupyterLab, you can run multiple notebooks in different tabs, use different versions of Python, and manage your entire data science workflow. To access JupyterLab, simply go to localhost-8080 in your browser.

If you want to use GPUs, there are a few additional considerations. You need to choose the GPU model, check its availability in different regions, and ensure that you have enough quota to provision a machine with that GPU in the desired zone. If you don't have enough quota, you can request an increase through the Quotas page in the Cloud Console.

Once you have determined the GPU model, the zone, and confirmed your quota, you can start a GPU-powered Deep Learning VM using a slightly different workflow than the CPU-powered VM.

Deep Learning VM Images on Google Compute Engine provide a convenient way to set up a machine learning environment without the hassle of manual installation and configuration. These VMs come preinstalled with popular frameworks and offer support for Cloud TPU and GPU. With the flexibility to edit hardware configurations and the power of tools like JupyterLab, you can efficiently work on your machine learning projects.

To create a GPU-powered virtual machine (VM) for deep learning on Google Cloud Platform, there are a few additional steps you need to take. Firstly, set the maintenance policy to terminate, as live migration is not supported for GPU-powered VMs. This means that during maintenance, the VM will be shut down, but you can configure it to auto restart. It is important to save your work before this happens.

Next, select the GPU type and count based on the quota deep dive we previously discussed. This ensures that you are using the appropriate GPU resources for your needs. Finally, add a piece of metadata to your VM. This metadata ensures that the necessary GPU drivers are installed automatically.

Although there are a few additional steps involved in setting up GPU-powered VMs, the effort is well worth it. Once you run the necessary command, the rest of the workflow remains the same. You can SSH into your machine and find JupyterLab waiting for you. By running "PIP freeze," you can confirm that the installed version of TensorFlow is tensorflow-gpu, indicating that the GPU support is enabled.

Instead of manually installing various libraries, consider using the Deep Learning VM Images on Google Cloud Platform. These preconfigured images provide a ready-to-use deep learning environment, saving you time and effort.

Thank you for exploring this topic with us. If you found this material helpful, please consider liking and subscribing to stay updated with the latest episodes. Head over to your Google Cloud Platform console and create your own deep learning environment in the cloud.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: TENSORFLOW HUB FOR MORE PRODUCTIVE MACHINE LEARNING

Code reuse is a fundamental aspect of software development, and it is equally important in machine learning. In this didactic material, we will explore how TensorFlow Hub can be used to easily load and customize state-of-the-art models in TensorFlow code.

TensorFlow Hub is a library that facilitates the publication, discovery, and consumption of reusable components of machine learning models. Its primary use case is transfer learning, which involves building machine learning models based on pre-trained models that have been trained on large datasets. By leveraging transfer learning, developers can train customized models on smaller datasets, improving generalization and reducing training time.

TF Hub is seamlessly integrated with TensorFlow, allowing users to import sections of a TensorFlow graph with ease. These reusable components, known as modules, can be imported into a TensorFlow program by creating a module object using either a URL or a file system path.

TF Hub offers a wide range of pre-trained models, with a focus on two main categories: images and text. The image models are extensively trained to classify objects and images. By reusing their feature detection capabilities, developers can create models that recognize custom classes with significantly less training data and time compared to building models from scratch. Some of the available image models include Inception V1, V2, and V3, Mobile Net, NASNet, PNASNet, and ResNet.

Text-based models in TF Hub are also highly capable. Currently, there are models like the universal sentence encoder, Elmo, NNLM, and others. These models have been trained on large datasets such as the 1 billion word benchmark and the Google News dataset, enabling developers to leverage their language processing capabilities.

To assist users in utilizing the modules effectively, TF Hub provides comprehensive guides on how to load and use the models for various use cases. These guides demonstrate how to perform tasks such as text classification of movie reviews using the NNLM model. Users can easily swap out datasets and models to suit their specific needs.

TF Hub is not just a repository for machine learning models; it is a complete framework for publishing and consuming models, all with a user-friendly and consistent interface. As developers create new and innovative models with their own data, they are encouraged to contribute to TF Hub by creating modules that others can reuse. This collaborative approach allows for continuous improvement and the discovery of new use cases.

By leveraging TensorFlow Hub, developers can build on top of some of the most advanced models available, expanding the possibilities of machine learning. TF Hub empowers users to think at a higher level of abstraction, encouraging innovative ways of remixing and reusing existing models.

TensorFlow Hub is a powerful tool for advancing in machine learning. By leveraging its capabilities, developers can easily load and customize state-of-the-art models, significantly reducing development time and improving model performance.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: TENSORFLOW EAGER MODE

The TensorFlow graph is a fundamental aspect of TensorFlow, but it can also be a source of frustration for users. In this episode, we will explore how to address some of the pain points associated with working with the TensorFlow graph.

One of the main challenges with the TensorFlow graph is that everything is part of it. This includes the numbers used to represent models and the operations themselves. This can make debugging difficult, as it feels like you need to write everything perfectly before running the code.

To address this issue, TensorFlow introduced Eager mode. In Eager mode, operations are executed immediately, allowing you to see the results as you work. This eliminates the need for a separate session and simplifies the debugging process.

To illustrate the difference, let's consider a basic example of multiplying numbers together. Without Eager mode, we need to wrap the code in a session and run it to obtain the results. However, with Eager mode enabled, the results are printed immediately, making it much more efficient and convenient.

Eager mode allows you to use TensorFlow more efficiently without sacrificing any functionality or features. It enables a more iterative and trial-and-error approach to software development, where you can quickly test code and identify errors.

Eager mode in TensorFlow provides a solution to the frustration associated with working with the TensorFlow graph. By enabling immediate execution of operations, it allows for more efficient and effective development. Give it a try and see if it's the right fit for you.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: JUPYTER ON THE WEB WITH COLAB

Creating and maintaining a data science environment can be a time-consuming and productivity-draining task. However, there is a tool that can simplify this process and allow you to focus on data science and machine learning. This tool is called Colaboratory, or Colab for short. Colab is a Jupyter Notebook environment specifically designed for machine learning education and research.

One of the key advantages of Colab is that it requires no setup to use. Unlike traditional environments, there is no need for local installation, library updates, or Python environment management. Colab comes preinstalled with many Python libraries, allowing you to start working immediately. Additionally, it seamlessly saves your work directly to Google Drive, ensuring that your progress is always up to date and easily accessible. This also enables collaboration, as you can share your work with others and receive comments, similar to Google Docs.

Colab also offers revision history, allowing you to access previous versions of your work, download them, or revert to them entirely if needed. This feature ensures that you can easily track changes and manage different iterations of your projects.

One of the most loved features of Colab is the ability to use a free GPU. Without the need for credit card information or signups, you can simply switch the runtime to include a GPU accelerator. Additionally, Colab also supports TPUs (Tensor Processing Units), which are Google's custom-built accelerators specifically designed for machine learning. TPUs enable fast training of advanced machine learning models.

In the Colab interface, there is a menu on the left-hand side that provides access to three important areas: the Table of Contents, Code snippets, and Files. The Table of Contents displays clickable headers that help you navigate through your notebook and understand its structure. This feature is particularly useful if you annotate your notebook cells properly. The Code snippets section is a valuable resource that provides useful samples for various tasks, such as file uploading and downloading, visualizations, and even image capturing with the camera. Each snippet links to a full Colab notebook where you can try the code in its own environment before incorporating it into your own work. Lastly, there is a file browser that helps you keep track of your files without the need to constantly run commands like "!ls".

By using Colab, you can skip the hassle of installing, setting up, and maintaining your own data science environment. It provides a user-friendly and collaborative platform for learning and experimenting with data science and machine learning models.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: UPGRADING COLAB WITH MORE COMPUTE

Colab is a popular platform for data science and machine learning, but sometimes we need more compute power to run certain models or handle larger datasets. In this episode, we will explore how to upgrade Colab with more compute by using Google Cloud Platform's deep learning VMs.

To begin, we need to create a deep learning VM with our desired specifications. We can do this by going to the Cloud Marketplace and selecting the deep learning VM. In this example, let's name our VM "colab-box" and give it 16 CPUs and 60 gigs of memory. Additionally, we can choose to add GPUs, such as the V100s. In this case, we will use two V100 GPUs.

While the VM is spinning up, let's discuss how we can connect Colab to our VM. There are two tricks we will use to make this connection possible. First, Colab can connect to any local runtime, including a Jupyter Notebook server running on your laptop. This means we can use Colab as the front end to our local server. To access this feature, simply go to the upper right-hand menu in Colab.

The second trick involves port forwarding on the deep learning VM. By running a specific command in our local terminal, we can forward the VM's port to our local machine. This allows us to access the Jupyter Server from a local front end. It's important to note that the command should be run in the actual local terminal, not in Cloud Shell.

Once the port forwarding is set up, we can connect Colab to our local runtime. This connection will be established with the port forwarded from the deep learning VM. Now, when we hook the two together, we can see that our Colab front end has access to the two V100 GPUs.

By understanding how Colab and deep learning VMs can work together, we can optimize our setup based on our needs. For example, we can start developing against a local back end and then switch to the deep learning VM when we require more power.

Upgrading Colab with more compute power is possible by utilizing Google Cloud Platform's deep learning VMs. By following the steps outlined in this episode, we can connect Colab to our VM and take advantage of its resources. This allows us to handle larger models and datasets, enhancing our data science and machine learning workflows.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: KUBEFLOW - MACHINE LEARNING ON KUBERNETES

Kubernetes is a platform used for managing containers. Machine learning workflows can become complex, especially in production environments. To address this, Kubeflow was developed as an open-source project to simplify running machine learning training and prediction on Kubernetes clusters. The goal of Kubeflow is to make deploying machine learning workflows on Kubernetes easy, portable, and scalable.

Kubeflow allows users to deploy their code to various environments, ranging from personal laptops to powerful GPU-equipped machines or even large-scale training and production Kubernetes clusters. It leverages the scalability of Kubernetes to handle increased demand and workload.

Originally, Kubeflow was created to open source TensorFlow Extended (TFX), which is Google's internal framework for running TensorFlow. However, it has evolved into a multi-architecture, multi-cloud framework for running complete machine learning pipelines. Kubeflow has a vibrant ecosystem with contributions from many individuals and organizations.

Even if you are not familiar with Kubernetes, Kubeflow provides an accessible way to deploy TensorFlow and other machine learning workloads onto Kubernetes successfully. It aims to be compatible with any Kubernetes environment, from personal laptops to bare-metal servers and public cloud platforms.

Installing Kubeflow on Google Kubernetes Engine (GKE) is straightforward thanks to the click-to-deploy user interface. With a few simple steps, you can enter your project ID, choose a deployment name, and select a zone. Kubeflow will automatically provision the necessary resources, making the deployment process simple and efficient.

Once Kubeflow is installed, you gain access to a user-friendly interface that enables you to train models, visualize them with TensorBoard, serve models, build pipelines, and manage these tasks through JupyterHub. Most of the time, you can use the same training code you would run locally, making it easy to transition to Kubeflow. You only need to add some configuration files specific to Kubeflow, and you can run your code without any modifications.

Kubeflow allows you to perform training on your Kubernetes cluster and choose whether to keep the exported model on disk within the cluster or send it to a Google Cloud Storage bucket. This flexibility enables easy sharing and deployment of trained models to systems outside of the cluster.

If you are interested in running machine learning pipelines on Kubernetes, it is time to explore Kubeflow. In the description below, you will find links to codelabs and resources that can help you get started. Whether you are a beginner or an experienced practitioner, Kubeflow provides a powerful toolset for managing machine learning workflows on Kubernetes.

Kubeflow project: https://goo.gle/2kmHqqh
Getting started: https://goo.gle/2lWhTnY
Intro to Kubeflow Codelab: https://goo.gle/2kz6OJ8
Intro to Kubeflow Pipelines Codelab: https://goo.gle/2k8Ntyu

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: ADVANCING IN MACHINE LEARNING
TOPIC: BIGQUERY ML - MACHINE LEARNING WITH STANDARD SQL

BigQuery ML is a tool that allows users to utilize machine learning within their SQL environment. It enables the creation and execution of machine learning models in BigQuery using standard SQL queries. BigQuery ML supports linear regression, binary logistic regression, and multiclass logistic regression.

Linear regressions are used to predict numerical values, such as insurance premiums or home values. Binary logistic regressions predict one of two classes, like determining whether a machine needs maintenance or identifying whether an image is a cat or a dog. Multiclass logistic regressions predict more than two classes, for example, determining whether to buy, hold, or sell a stock or classifying traffic on a highway as light, medium, heavy, or a parking lot.

To use BigQuery ML, you can access it through the BigQuery UI, command line, or API. In this example, we will explore the BigQuery UI. The queries used in the UI are identical to those used in the command line or API.

To demonstrate BigQuery ML, we will use a sample data set from Google Analytics. The data set includes information about transactions, such as the user's operating system, whether it was a mobile device, the country of origin, and the number of page views. We will try to predict whether a transaction took place based on these criteria.

To build a model, we add a create model statement at the top of the SQL query, specifying the table name and giving the model a name. In this case, we will call it "sample model." We also specify that we are performing a logistic regression. By creating a column named "label," BigQuery ML knows which column we want to predict on and which columns are inputs. If the column is named differently or if we want to use a different column as the label, we can set the input label calls option in the create model statement.

Once the model starts training, we can check the training statistics to see the progress and performance of the model. This can be done by clicking on the model in the left-hand panel and going to the Model Stats tab.

After the create model statement finishes running, we can evaluate the model using ML.EVALUATE. We wrap the original SQL query in ML.EVALUATE, passing in the newly created model. This will provide us with various metrics of the model's performance.

Finally, we can make predictions using the model by using the ML.PREDICT function. We wrap the same SQL query used in evaluation with ML.PREDICT and pass in the model. This will generate predictions based on the model.

BigQuery ML is a powerful tool that allows users to perform machine learning within their SQL environment. It eliminates the need to move data to other platforms and provides an easy way to analyze patterns in data.

BigQuery ML is a powerful tool that allows you to perform machine learning tasks directly within your SQL environment. With BigQuery ML, you can build and deploy machine learning models using standard SQL statements, making it easy for data analysts and SQL developers to leverage the power of machine learning.

One of the key features of BigQuery ML is its ability to perform predictions on large datasets. By using the PREDICT function, you can apply your trained models to new data and generate predictions. In the transcript, it is mentioned that you can see the predictions for each country ranked, giving you valuable insights into the data.

The transcript also highlights the importance of analyzing data across different dimensions. In this case, it suggests diving deeper into the data to understand how transactions match up across mobile devices and different operating systems. This demonstrates the flexibility of BigQuery ML in exploring and analyzing data from various perspectives.

BigQuery ML provides a simple and efficient way to perform machine learning tasks within your SQL environment. It allows you to build and deploy models using standard SQL statements, and provides features for

performing predictions on large datasets. By leveraging the power of BigQuery ML, you can gain valuable insights from your data and make informed decisions.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: PYTORCH ON GCP

Google Cloud Platform (GCP) offers more than just TensorFlow for running machine learning libraries. In this episode, we will explore how to run PyTorch on GCP.

If you want to use PyTorch without any installation or setup, you can use platforms like Colab or Kaggle kernels. Simply sign in with your Google account and you're ready to use PyTorch. Colab even has a GitHub integration feature that allows you to import public IPython notebook files directly into Colab. This makes it easy to modify and run code from examples repositories.

Kaggle also provides a vast library of kernels created by the community, covering various datasets. These kernels often come with excellent discussions and annotations, so you don't have to start from scratch. Just sign in and you can edit your code right away.

For PyTorch developers, there are deep learning virtual machines available on GCP. These virtual machines come with pre-baked PyTorch images and Nvidia drivers, making it easy to have full control over your environment and utilize GPUs for increased compute power. You can also save your work directly to the virtual machine.

Additionally, Google is collaborating with the PyTorch team to enable TensorBoard support for visualizing training progress and TPU (Tensor Processing Unit) support. Running Colab with TPUs for free provides an excellent combination of resources. If you have ideas for using Cloud TPUs with PyTorch, you can email the PyTorch-TPU@googlegroups.com team to discuss your PyTorch workloads and potential acceleration options.

Whether you're just getting started with PyTorch or need to run a large training job, GCP offers various PyTorch-friendly options to suit your needs. Start exploring PyTorch on GCP today and take advantage of the powerful capabilities it provides.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: AUTOML TABLES

Structured data is a valuable resource for machine learning, and Google has introduced a new tool called AutoML Tables to automatically build and deploy machine learning models on structured data. AutoML Tables is designed to simplify the modeling process and save time by automating the selection of models and hyperparameters.

One of the key features of AutoML Tables is its ability to handle a wide range of data types, including numbers, classes, strings, timestamps, lists, and nested fields. What sets it apart is that it requires no coding work. Users can simply import a CSV file, make a few selections, and let the system do the rest.

To get started with AutoML Tables, users need to import their training data. This can be done by selecting a table from BigQuery or a file from Google Cloud Storage. The system then analyzes the columns of the dataset and allows users to edit the auto-generated schema and select the column for prediction.

The next step is the Analyze tab, which provides an overview of the data. Users can click on different column names to see statistics about their data. After analyzing the data, users can proceed to the training phase by clicking the Train button. There are options to set a maximum training budget, allowing users to experiment with their data and limit training time before committing to a full training run.

During training, AutoML Tables not only tunes the model but also selects the most suitable model. This process may take some time, but users don't have to do anything and can take a break while waiting. Once training is complete, users can evaluate the model's performance in the Evaluation tab, which provides metrics and insights.

The final step is to deploy the model and get predictions. AutoML Tables offers an editor in the browser that allows users to make requests to the endpoint without setting up a local environment. Users can also toggle between online predictions via the REST API and batch predictions, which process entire files or tables.

AutoML Tables offers a significant advantage in terms of model performance compared to manual modeling. By leveraging state-of-the-art models and carefully tuning them during training, users can achieve better results with less effort. If you have structured data and are considering machine learning, AutoML Tables is a tool worth trying.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: TENSORFLOW PRIVACY

Machine learning aims to generalize about a set of data, but sometimes it is possible to recover information about specific training examples from a trained model. To address this, TensorFlow Privacy was developed as a tool to train models with privacy. Google's responsible AI practices recommend safeguarding the privacy of ML models and using techniques that establish mathematical guarantees for privacy.

When training a machine learning model, the goal should be to learn general patterns rather than specific facts about training examples. This is important to prevent the model from memorizing specific data, especially when it corresponds to user actions or attributes. TensorFlow Privacy ensures that models do not learn or remember any details about specific users by introducing modifications to the gradient calculation during the training process.

Gradients express the direction and magnitude of updates to a model's internal state. Typically, a batch of data is used to determine the gradient and update the model's weights. TensorFlow Privacy modifies this process by averaging multiple mini-gradient updates on a subset of the full batch. Each mini-gradient's value is clipped to limit its impact, and Gaussian random noise is added to the average. This ensures that no individual example is memorized by the model.

To use TensorFlow Privacy, there is no need to modify the model architecture or training procedures. Instead, a different optimizer provided by the TensorFlow Privacy Library is used. The optimizer's clipping, noise level, and number of mini-batches to average together can be customized. Although this introduces additional hyperparameters, it is a simpler solution compared to rewiring the entire model or risking exposure of users' data.

An example highlighted by the TensorFlow Privacy team involved a language learning model that identifies English sentences resembling financial news. When comparing the model trained with TensorFlow Privacy to one without, there were sentences where both models agreed, but also sentences that they disagreed on. These anomalous sentences accounted for most of the differences in accuracy between the two models. This demonstrates the importance of considering more than just metrics and suggests that TensorFlow Privacy can help identify aspects of a data set.

TensorFlow Privacy is a tool that allows for training machine learning models with privacy guarantees. It modifies the gradient calculation process to prevent memorization of specific examples and offers an alternative optimizer that can be easily integrated into existing models. By using TensorFlow Privacy, models can capture general patterns while protecting user privacy.

TensorFlow Privacy is a tool that allows users to utilize the concepts of science and math behind differential privacy. It provides a means to incorporate privacy protection into machine learning models. While tuning hyperparameters can still require some artistry, the TensorFlow Privacy team has provided reasonable initial values to facilitate the exploration process.

For more in-depth information and practical examples, you can refer to the blog post linked in the description. It contains additional details that can enhance your understanding of TensorFlow Privacy and its applications.

Thank you for watching this material of Cloud AI Adventures. If you found it valuable, please show your support by liking the material and subscribing to receive notifications for future episodes. Stay updated with the latest developments in the field of artificial intelligence.

Make use of TensorFlow Privacy to ensure the safety and privacy of your users in machine learning applications.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: VISUALIZING CONVOLUTIONAL NEURAL NETWORKS WITH LUCID

In recent years, new research has revolutionized our understanding of how neural networks work and how they make predictions. This research has provided tools and building blocks for explaining the inner workings of neural networks. In this material, we will explore convolutional neural networks (CNNs) and how they perceive the world.

CNNs have significantly advanced the field of image recognition and achieved remarkable accuracy across various tasks. However, understanding the intermediate layers of a CNN and what happens between the inputs and outputs has been a challenge. In this material, we will explore different approaches to gain intuition about how CNNs work, ranging from focusing on individual neurons to analyzing the responses of entire layers.

Understanding how a neural network arrives at a prediction is crucial in real-world applications. It provides additional value by helping users better understand the reasons behind a prediction. For example, in medical imaging, knowing which parts of an image contribute to a prediction can be crucial for accurate diagnosis. By understanding the reasoning behind a prediction, doctors can assess whether the model is making predictions for the right reasons and potentially discover new patterns that were previously unknown.

To construct this understanding of a neural network's predictions, we need tools and approaches that allow us to analyze and visualize the network. Before diving into these tools, let's first understand the structure of a convolutional neural network.

Convolutional neural networks are named after their convolutional layers, which work together to understand different details of an image. In the early layers, close to the inputs, the convolutional layers detect basic lines, shapes, and patterns. As we move further into the network, the layers respond to more complex inputs, resembling parts of real-world objects or animals. This information propagates through the network, eventually reaching the final layers that generate the network's conclusions, which directly correspond to the expected categories.

Each convolutional layer consists of multiple parts. At the lowest level are individual neurons, which are the fundamental building blocks of a neural network. The strength of a neuron's response to input determines its behavior. Neurons are connected in channels, which process the outputs of the previous layer. Each channel looks for different features within the input. All the outputs from the channels are combined and passed to the next layer.

Now that we have a basic understanding of how convolutional neural network layers are structured, let's explore how we can determine what a specific neuron or group of neurons is "looking for." During a forward pass of an image through the network, each neuron responds or activates to a different degree. We can measure this activation strength, which is simply the numerical value associated with the activation. A larger magnitude indicates a stronger activation. Activations can be positive or negative, representing different types of responses.

To visualize and understand what a neuron is looking for, we can analyze its activation patterns and identify the types of inputs that trigger strong responses. This information helps us interpret the behavior of individual neurons and gain insights into the network's decision-making process.

Understanding how convolutional neural networks perceive and interpret images is crucial for interpreting their predictions and gaining insights into their decision-making process. By analyzing individual neurons and their activation patterns, we can better understand what features and patterns the network is looking for. This knowledge can be valuable in various applications, such as medical imaging, where accurate and explainable predictions are essential.

Convolutional neural networks (CNNs) are a type of artificial intelligence model commonly used in image recognition tasks. In order to better understand how these networks work, it is important to visualize how individual neurons and channels within the network respond to different inputs. This can help us gain insights into the patterns and features that the network is able to detect.

One way to visualize the activation of a neuron is by optimizing an input image to maximize its activation value. By starting with a random noise image and iteratively adjusting it based on the neuron's activation, we can eventually obtain an image that maximally activates that specific neuron. These optimized images often have a focused region of interest surrounded by a blended background.

Lucid, a library for visualizing neural networks, simplifies the process of optimizing input images. Instead of manually writing an optimization loop, Lucid allows users to specify the layer, channel, and neuron they are interested in, and it takes care of the rest.

In addition to visualizing individual neurons, we can also optimize for entire channels of neurons. This results in images that blend the patterns from multiple neurons within the channel. By interpolating between different neurons, we can create images that lie along the spectrum between the two channels, providing further insights into the patterns detected by the network.

In the next episode, we will explore feature visualization at the image level, creating activation grids that show how different parts of an image activate the network. This will allow us to gain a holistic understanding of how the network operates as a whole.

If you are interested in learning more about feature activations and experimenting with Lucid yourself, you can check out the Distilled.pub article mentioned in the material and explore the Lucid Library on GitHub.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: UNDERSTANDING IMAGE MODELS AND PREDICTIONS USING AN ACTIVATION ATLAS

Feature activations in convolutional neural networks can be extended to entire images to create activation grids and produce an activation atlas. Spatial activations give us insight into the features that are most strongly activated in specific regions of an image. By building activation grids for different layers of the network, we can understand how activations propagate through each layer.

To determine the importance of a region, we can scale the size of each grid based on the activation strength. This allows us to see which regions are most strongly activated per layer and how these activations propagate through the network. Activation grids provide information about the saliency of different parts of the image.

To understand other types of activations, we can feed in a large number of example images and record their activation routes. These activations correspond to higher-dimensional locations in the neural network's activation space. To visualize this space, we can project it down to two dimensions and divide it into grids. Taking the average of activations within each grid, we can resize the grids based on the number of activations, giving us a sense of how strongly the network responds to different inputs.

This entire process can be repeated for different layers, resulting in an activation atlas. The activation atlas provides a web interface to navigate through the space of activations. Starting from a specific layer, we can observe the smooth transition of images as we move through different regions. It is also possible to filter the atlas to focus on specific class activations, such as the great white shark.

The activation atlas is a powerful tool for understanding the behavior of convolutional neural networks and gaining insights into the features they learn to recognize in images.

Neural networks are powerful tools that can learn to recognize patterns and make predictions based on data. However, they can sometimes produce unexpected results. In this material, we will explore how activation atlases can help us understand and uncover these peculiarities in image models and predictions.

Activation atlases are visualizations that show how different parts of an image contribute to the predictions made by a neural network. By examining these atlases, we can gain insights into the inner workings of the network and identify any unusual associations it might have learned.

In the provided material, the speaker discusses their experiments with a neural network trained to classify images. They noticed that the network had learned to associate certain images, such as baseballs, with unrelated categories like great white sharks. Curious about this behavior, they decided to investigate further.

The speaker attempted to fool the network by adding a baseball image on top of an image of a gray whale. Initially, the network did not classify the altered image as a great white shark. However, by adjusting the size and position of the baseball image, they eventually managed to deceive the network into predicting a great white shark.

This experiment highlights the ability of neural networks to be influenced by certain visual cues and the importance of understanding their vulnerabilities. The speaker also discovered that the network associated airships with great white sharks in a different layer. They replicated the previous experiment by manipulating the airship image, achieving similar results.

The speaker encourages the audience to try these experiments themselves and share their findings. They emphasize that while neural networks can be opaque, tools like activation atlases provide a window into their inner workings. By using activation atlases, we can gain a better understanding of how the network processes information and uncover any unexpected associations it may have learned.

Activation atlases are valuable tools for understanding image models and predictions made by neural networks. They allow us to visualize and analyze the contributions of different parts of an image to the network's predictions. By exploring these visualizations, we can gain insights into the network's behavior and uncover any unusual associations it may have learned. Experimenting with activation atlases can help us better understand

the strengths and weaknesses of neural networks and pave the way for future advancements in this field.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: NATURAL LANGUAGE PROCESSING - BAG OF WORDS

Natural language processing (NLP) presents unique challenges compared to other data types like images and structured data. To effectively model natural language, a foundational technique called "bag of words" is often used. In this approach, words are converted into numerical representations, allowing them to be processed by machine learning algorithms.

Natural language is characterized by its inherent structure and free-form nature. Multiple ways of expressing the same idea and the existence of similar words with different meanings make it necessary to transform text into matrices or tensors, which are the preferred input formats for machine learning algorithms. While images and structured data already have inherent representations as matrices, natural language requires a different approach.

One way to convert words into numerical representations is through the bag of words approach. Imagine learning English for the first time and having a limited vocabulary consisting of 10 words, such as "dataframe," "graph," "plot," "color," and "activation." To classify arbitrary text, we can identify the presence of these words in a sentence. For example, given the sentence "how to plot dataframe bar graph," we recognize the words "plot," "dataframe," and "graph." By representing our vocabulary as an array and setting the corresponding indices to 1, we encode the sentence into an array of numbers. The order of the words in the sentence doesn't matter; what matters is the order of the words in the vocabulary list.

The same encoding process is applied to the labels, which represent the topics or categories we want to classify. Each label is represented as an array, with relevant indices set to 1 and the rest set to 0. This allows us to handle sentences that may have multiple labels attached to them.

By converting both the inputs (words) and outputs (labels) into numerical representations, we can use machine learning algorithms to map one set of numbers to another. The preprocessing step, where text is transformed into numerical representations, is crucial in this process. Bag of words is a simple yet effective approach for this task, and it often yields surprising results in certain situations.

To implement a bag of words model with code, we can utilize the Tokenizer class in the Keras preprocessing library. The Tokenizer allows us to specify the size of the vocabulary we want to use. After fitting the Tokenizer on the training data, selecting the most common words, we can build the model using a standard fully connected deep neural network. For multiple label classification, where more than one label can be true for a single input, we use a sigmoid activation function and binary cross-entropy loss.

The bag of words approach is a foundational technique in natural language processing. By converting words into numerical representations, we can effectively process text using machine learning algorithms. The bag of words model, when combined with appropriate preprocessing and model building techniques, can yield accurate results in classifying natural language.

Natural language processing (NLP) is a field of artificial intelligence (AI) that focuses on the interaction between computers and human language. It involves the ability of computers to understand, interpret, and generate human language in a way that is meaningful and useful. One approach to NLP is the use of machine learning techniques, such as the bag of words model.

The bag of words model is a simple and effective way to represent text data for machine learning. It treats each document as a collection of words and ignores the order in which the words appear. Instead, it focuses on the frequency of each word in the document. This approach is called the "bag of words" because it essentially creates a "bag" of words and their frequencies.

To implement the bag of words model, we first need to preprocess the text data. This involves removing any irrelevant information, such as punctuation and stop words (common words like "the" and "and" that do not carry much meaning). We also need to convert the text into a numerical representation that can be used by machine learning algorithms.

Once the text data has been preprocessed, we can create a vocabulary of unique words that appear in the documents. Each word in the vocabulary is assigned a unique index. We then represent each document as a vector, where each element of the vector corresponds to the frequency of a word in the document. This vector is known as the "bag of words" representation of the document.

The bag of words model has several advantages. It is simple to implement and computationally efficient. It also allows us to easily compare documents and measure their similarity based on the words they contain. However, it has some limitations. It does not take into account the order of the words, which can be important in some applications. It also does not capture the semantic meaning of the words, only their frequency.

The bag of words model is a useful approach to natural language processing that allows us to represent text data in a numerical form that can be used by machine learning algorithms. It is a simple and effective way to encode text and can be a good starting point for further exploration of NLP techniques.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: AUTOML NATURAL LANGUAGE FOR CUSTOM TEXT CLASSIFICATION

Natural Language processing can be challenging, especially when dealing with domain-specific language. Standard pre-trained models may not always be effective in industry-specific applications. In this educational material, we will explore how to train a text classification system on custom data using AutoML natural language for custom text classification.

To begin, we will look at how to use AutoML natural language to build a custom text classifier. We will use a dataset from Stack Overflow, a valuable tool for developers. However, manually tagging questions on Stack Overflow can be tedious. Therefore, we will build a machine learning model to recommend tags for Stack Overflow questions about machine learning.

To train our model, we will utilize the BigQuery datasets program, which provides over 25 gigabytes of Stack Overflow questions. This dataset will serve as our training data. AutoML Natural Language is convenient for this use case because it allows us to experiment with different options easily.

We will start by building a simple classification model that differentiates between questions about TensorFlow and other topics. To preprocess the data, we will combine the title and text fields, remove HTML formatting, and replace vertical bars in the tags with commas. We will limit our questions to those tagged with TensorFlow, Keras, Matplotlib, Pandas, and Scikit Learn, focusing specifically on machine learning topics.

While it may be tempting to create a tag recommendation system using an IF statement, there are cases where questions are about TensorFlow without explicitly mentioning it. To address this, we will replace obvious words with a different word, such as "avocado." This prevents the model from using the word TensorFlow as a cheat to identify TensorFlow-related questions.

Training AutoML Natural Language is straightforward. Once trained, we can test the model with questions outside our training set. If the model correctly recognizes the topic, we can proceed to label questions with up to five different tags related to machine learning libraries.

AutoML Natural Language provides training scores, false positives, and false negatives for each class, allowing us to identify potential errors in our data or model. We can test sample questions in the Predict tab to verify the accuracy of the tags and scores.

AutoML Natural Language simplifies the process of testing ideas, labeling, and training models. We can focus on the problem at hand, tagging Stack Overflow questions, without getting caught up in tokenization, word representation models, or embeddings.

Once we have a successful model, we can weigh the trade-off between building a custom model ourselves and using the model created by AutoML Natural Language.

AutoML Natural Language offers a convenient solution for building custom text classifiers. By utilizing training data from Stack Overflow and experimenting with different options, we can create accurate models for tagging questions. This allows us to focus on the problem we are trying to solve rather than the complexities of modeling text.

Artificial Intelligence (AI) has become an integral part of many industries, and one of the key players in this field is Google Cloud Machine Learning. In this didactic material, we will focus on a specific area of expertise within machine learning, namely AutoML Natural Language for custom text classification.

AutoML Natural Language is a powerful tool provided by Google Cloud that allows users to build custom models for text classification. With this technology, businesses can automate the process of analyzing and categorizing large volumes of text, saving time and resources.

The first step in using AutoML Natural Language is to create a dataset. This dataset consists of labeled examples that will be used to train the model. The more diverse and representative the dataset is, the better

the model will perform. Once the dataset is prepared, it can be uploaded to Google Cloud.

Next, the AutoML Natural Language platform takes over. It automatically analyzes the dataset and trains a machine learning model specifically tailored to the user's needs. This model is capable of understanding and classifying text based on the patterns it has learned during the training process.

One of the key advantages of AutoML Natural Language is its ability to handle custom categories. This means that users can define their own classification labels, allowing the model to accurately categorize text according to their specific requirements. For example, a company in the retail industry might want to classify customer reviews as positive, negative, or neutral. With AutoML Natural Language, this can be easily achieved.

To ensure the accuracy of the model, it is important to evaluate its performance. AutoML Natural Language provides various evaluation metrics, such as precision, recall, and F1 score, to assess how well the model is performing. These metrics help users understand the strengths and weaknesses of their model and make necessary adjustments if needed.

Once the model is trained and evaluated, it can be deployed for production use. This means that it can be integrated into existing systems or applications to automate text classification tasks. The model can process large volumes of text in a matter of seconds, making it a valuable tool for businesses dealing with vast amounts of textual data.

AutoML Natural Language for custom text classification is a powerful technology offered by Google Cloud Machine Learning. It allows users to build accurate and efficient models for analyzing and categorizing text. By leveraging this technology, businesses can save time and resources while gaining valuable insights from their textual data.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: TENSOR PROCESSING UNITS - HISTORY AND HARDWARE

Machine learning has gained significant popularity in recent years, leading to a growing demand for specialized computing resources for training and predictions. To meet this demand, Tensor Processing Units (TPUs) were developed. In this didactic material, we will explore the history and hardware of TPUs, focusing on the original TPU design.

The first TPU was created as a PCI Express expansion card that could be plugged into existing server racks in Google data centers. It had a clock speed of 700 megahertz and consumed 40 watts of power. Since its production in 2015, the TPU has been used in various applications, including Google Photos, Google Translate, and the AlphaGo match in South Korea.

What made the TPU V1 stand out was its superior performance compared to existing CPUs and GPUs at the time, as well as its high performance per watt of energy. The chip was specifically designed for deep learning, utilizing reduced precision, a matrix processor, and a minimalistic design to minimize overhead.

Reducing the precision of the chip to 8-bit integers instead of the conventional 32-bit floating-point numbers allowed for more integer multiplier units to fit into a single chip. This reduction was achieved through a technique called quantization, which mapped 32-bit floating-point numbers to 8-bit integers.

The TPU's matrix processor played a crucial role in its efficiency. Unlike conventional processing systems that frequently read and write to memory, the TPU's matrix processor performed calculations without memory access. This was made possible by using a systolic array, which allowed for large hard-wired matrix calculations. In simple terms, while a CPU prints out letters one by one and a GPU prints out a whole line at a time, the TPU stamps out entire pages at a time.

The core of the TPU is a massive systolic array capable of performing up to 250,000 operations per clock cycle. Most of the chip is dedicated to matrix multiplication followed by addition operations. Despite its seemingly modest clock speed of 700 megahertz, the TPU performs a significant number of operations per clock cycle.

It is important to note that the TPU V1 was designed solely for predictions. While the original TPU is not directly accessible to users, it powers various Google services such as Google Photos and Google Translate.

In the next material of "AI Adventures," we will delve into the TPU V2 and V3, exploring how they work and how to run code on them. Stay tuned for more insights into the world of high-performance computing and machine learning.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: EXPERTISE IN MACHINE LEARNING
TOPIC: DIVING INTO THE TPU V2 AND V3

Previously on "AI Adventures," we explored the Tensor Processing Unit (TPU) v1 and its design. In this episode, we will dive into the architecture of the TPU v2 and v3, as well as the hardware design choices that make these chips powerful and specialized for machine learning.

The TPU v2 was developed based on the lessons learned from the TPU v1. It is considerably larger and features four chips instead of one. With massive heat sinks, the TPU v2 has 180 teraflops of compute, allowing it to perform 180 trillion floating-point operations per second. Unlike its predecessor, the TPU v2 is capable of both training and prediction.

The layout of the TPU v2 is interesting. Each board contains four chips, and each chip has two cores. Each core consists of a matrix unit, a vector unit, and a scalar unit, all connected to 8 gigabytes of high-bandwidth memory. This means that each board has 8 cores and 64 gigabytes of memory. The matrix unit is a 128 x 128 systolic array.

What sets the TPU v2 apart is its use of a new data type called bfloat16. The bfloat16 combines the range of a 32-bit floating-point number with the storage space of a 16-bit floating-point number. The bfloat16 was introduced by the Google Brain team, hence the 'b' in bfloat16. Although it represents fewer decimal places than a standard 16-bit floating-point number, the reduced precision is acceptable for neural networks while maintaining high accuracy. The use of bfloat16 allows the TPU v2 to fit in more computational power.

TPU v2 pods are another exciting development. A TPU v2 pod consists of 64 TPUs connected together, and the entire pod can be used as if it were one machine. Each chip has two cores, and with four chips per board and 64 TPUs per pod, a TPU pod has a total of 512 cores, providing over 11 petaflops of processing power. Smaller subdivisions of the pods, such as a quarter pod or a half pod, can also be used.

Moving on to the TPU v3, it is essentially an upgraded version of the TPU v2 with a blue color and water cooling. The water cooling system allows the TPU v3 pods to support more TPUs in a smaller vertical space. A full TPU v3 pod is eight times faster than a v2 pod and provides over 100 petaflops of compute power.

Both the TPU v2 and TPU v3 are available for use. The pricing varies depending on the region and the chosen TPU, with TPU pods being more expensive due to the increased number of connected TPUs. However, for large-scale training jobs that require quick results, the cost is justified.

Programming the TPU is becoming easier with each iteration. With the release of TensorFlow 2.0, the Keras API can be used to code up training for TPUs. There are also great samples and implementations of state-of-the-art models available. By swapping in your custom dataset, you can easily train your models using TPUs.

To learn more and get started with TPUs, visit g.co/cloudtpu.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: AI PLATFORM TRAINING WITH BUILT-IN ALGORITHMS

Google Cloud's AI Platform Training offers built-in algorithms that allow users to train machine learning models without writing code. This feature acts as a convenient wrapper around AI Platform Training, bridging the gap between AutoML and full custom training jobs. While still in beta at the time of recording, it may already be generally available in the future.

To demonstrate the capabilities of AI Platform Training with built-in algorithms, we will use the US Census dataset, a classic binary classification problem. Our goal is to predict whether a household income will be above or below $50,000 per year.

To begin using AI Platform Training, navigate to the Jobs menu in the Cloud console and click on New Training Job. Select the option for Built-in Algorithm Training. Currently, there are three structured data algorithms available: XGBoost, Linear Learner, and Wide and Deep Learner. XGBoost utilizes gradient-boosted trees, Linear Learner models linear regression, and Wide and Deep Learner combines the benefits of a wide linear model with deep neural networks.

Once an algorithm is selected, the next step is to specify the training data. The system requires the input data to be in a Google Cloud Storage URL path and expects a headerless CSV file for structured data. The prediction target column must be the first column in the CSV file. If necessary, light preprocessing may be required to rearrange the data. For easier generation of the appropriate output format, consider using a spreadsheet software to move the label column to the first position and export the file as a CSV.

Validation and test data can be specified as separate files or as a percentage of the training data. Using percentages is recommended if the data is mixed together to ensure representative metrics. If separate files are used, ensure that all three (training, validation, and test) are representative of the same reality.

The next screen allows for the specification of algorithm arguments. Each algorithm has a list of parameters that can be set, with default values provided. Extensive documentation is available for each parameter. Additionally, these parameters can be used for HyperTune, Google Cloud Platform's hyperparameter tuning service. To use HyperTune, enable it and provide lower and upper bounds for the hyperparameter to be tuned. Enabling early stopping of unpromising trials can save compute and time.

After setting the algorithm arguments, provide a job ID, region, and select the scale tier for the training job. Once these details are provided, the training will begin. During the training process, users can engage in other activities or start additional training jobs in parallel since starting a job is lightweight.

Once the training is complete, users can evaluate the model's performance, make predictions, and deploy the model for production use.

When using Google Cloud AI Platform for training models with built-in algorithms, you have access to various features that allow you to view job details and resource utilization. By clicking on the job, you can see information such as resource usage and links to Stackdriver Logging. Additionally, you can deploy your trained model directly from the platform, which includes a Deploy Model button that retrieves the exported model from Google Cloud Storage and sets up a REST API for making predictions.

The AI Platform training and deployment process is designed to be user-friendly and does not require writing any code. This allows users to save time while still having control over the model's specific details. You can adjust the parameters and settings to customize the model for your specific use case and dataset, ensuring high-quality results.

To learn more about the AI Platform and explore further examples, you can refer to the links provided in the description below. These resources will provide you with additional information and guidance on using the platform effectively.

Thank you for watching this material on Cloud AI Adventures. If you found it helpful, please consider liking the

material and subscribing to the channel for future updates. Now, take a look at AI Platform Training using built-in algorithms and discover the kind of results you can achieve.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: TRAINING MODELS WITH CUSTOM CONTAINERS ON CLOUD AI PLATFORM

Are you interested in running machine learning on Docker containers in the cloud? Do you want to future-proof your workflow and have the flexibility to use any library of your choice? If so, you've come to the right place. In this educational material, we will explore how to run custom containers on Google Cloud AI Platform.

Using custom containers on Google Cloud AI Platform offers several key benefits. Firstly, it provides faster start-up time. By using a custom container with all your dependencies preinstalled, you can save the time it would take to install those dependencies when starting up your training application. This leads to faster training and reduces the usage of network and compute resources.

Secondly, custom containers allow you to use any machine learning framework you prefer. Simply install your chosen framework in your custom container and use it to run your jobs on AI Platform. This not only gives you the freedom to use any tool you want but also future-proofs your workflow, allowing you to easily switch libraries or add new ones for experimentation.

Thirdly, custom containers enable you to use the latest build of a library or even an older version if required. You have the flexibility to choose the specific version of a library that suits your needs, without being forced to update if you're not ready.

It's important to note that when building your own container image, you need to install any additional functionality you require. For example, if you want hyperparameter tuning or GPU support, you must include the necessary libraries to enable those features.

Now, let's take a look at how you can use custom containers to train a model. In this example, we will use PyTorch to train on the MNIST dataset.

To begin, you need to create the container. This can be done by writing a Dockerfile that installs PyTorch, Cloud ML Hypertune, and downloads gsutil for exporting cloud storage. The Dockerfile also configures some paths and copies in the training code. It's important to test your Dockerfile locally before using it by building the image and running it. Once you are satisfied with the Dockerfile, you can upload the image to Google Container Registry (GCR). AI Platform will look for your image in GCR during the training job.

Once your image is uploaded to GCR, creating a job is similar to normal usage of AI Platform Training. The only difference is that you need to include an extra command line argument when submitting the training job. Use the --master-image-uri flag and pass in the GCR URI of your image.

You might wonder why you should go through the trouble of building a Docker image if you can simply run the training locally. There are several scenarios where using custom containers is particularly useful. For example, if your data is larger than your machine's storage or if the dataset is secured behind permissions that prevent running the full training job on a local machine. Additionally, if your local machine lacks adequate compute resources, such as multiple GPUs, using custom containers can solve this issue. Finally, you might simply prefer working with containers or need to do distributed training across multiple machines.

AI Platform Training with custom containers provides a great set of features and functionality for those who require containers. It offers flexibility, faster start-up time, and the ability to use any machine learning framework of your choice. We encourage you to explore and experiment with AI Platform Custom Containers for all your custom training needs.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: USING THE WHAT-IF TOOL FOR EXPLAINABILITY

The What-If Tool is an open-source project developed by the People and AI Research team at Google. It is designed to provide a way to inspect machine learning models with minimal code required, allowing users to better understand the behavior of their models.

The tool can be used within a Jupyter or Colab notebook or embedded into the TensorBoard web application. It specializes in structured data and text analysis, making it a valuable tool for exploring and probing machine learning models.

In this example, the What-If Tool is demonstrated using the US Census data set. Two models, a linear classifier and a deep classifier, were trained on this data set and will be compared using the tool. The tool refers to these models as Model 1 and Model 2.

The first view of the What-If Tool includes Facets Dive, which allows users to slice and dice their data along the x and y-axes. This view also shows the distribution of data points and provides the ability to edit values and see the corresponding prediction. By changing the values near the decision boundary, users can gain insights into which features have a greater effect on the outcome of predictions. The tool also includes the ability to show the nearest counterfactuals, which are the most similar data points that received a different classification.

The next tab in the tool is called Performance and Fairness. This tab allows users to analyze overall model performance and investigate model performance across different data slices. By adjusting the prediction threshold and slicing the data by up to two fields, users can explore the impact on the confusion matrix and examine fairness optimization strategies.

The third tab in the tool is Facets Overview, which provides an overview of the distribution of data points across different features. This tab is useful for identifying imbalanced data and gaining key insights about each feature.

The What-If Tool offers many features and use cases that cannot be covered in detail in this episode. However, if you are interested in learning more, you can join the What-If Tool community on the Google Group, where you can find announcements of new features and engage in discussions.

The What-If Tool is a powerful tool for inspecting and understanding machine learning models. It provides visualizations and analysis capabilities for structured data and text, allowing users to gain insights into model behavior, performance, and fairness.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: INTRODUCTION TO EXPLANATIONS FOR AI PLATFORM

Getting explanations for predictions is an increasingly important aspect of artificial intelligence. In this didactic material, we will explore how to use Google Cloud AI Platform's Prediction service to generate explanations for your predictions.

AI Explanations is a feature that is now built into Cloud AI Platform's Prediction service. It integrates feature attributions into AI Platform Prediction, allowing you to understand your model's outputs for classification and regression tasks. With AI Explanations, you can determine how much each feature in the data contributed to the predicted results. This is valuable because it enables you to verify that your model is behaving as expected, identify and address bias, and gain insights on how to improve your training data and model.

Feature attributions are available for both tabular and image data, and are specific to individual predictions. It is important to note that AI Explanations currently only supports models trained on TensorFlow 1.x. If you are using Keras to specify your model, you will need to convert it into an estimator using the model to estimator utility.

AI Explanations offers two methods for feature attribution: sampled Shapley and integrated gradients. Both methods are based on the concept of Shapley values, which is a cooperative game theory algorithm that assigns credit to each player in a game for a particular outcome. In the context of AI Explanations, each feature is treated as a player in the game, and credit is assigned to each feature for the outcome of a prediction.

Integrated gradients are well-suited for differential models like neural networks, especially those with large feature spaces. This method computes the gradients of the output with respect to the input, multiplied element-wise with the input itself. It provides a Taylor approximation of the prediction function at the input.

Sampled Shapley, on the other hand, assigns credit for the outcome of each feature and considers different permutations of those features. It is most suitable for non-differential models like ensembles of trees.

For those interested in diving deeper into these explainability methods, there are several articles and research papers that provide more detailed explanations. These resources are linked below and offer valuable insights into the concepts discussed.

To try out AI Explanations for your deployed model, you can refer to the guides in the documentation. There is a guide for tabular data and another one for image data. These guides are presented as Colab notebooks, making it easy to experiment with the feature.

Lastly, it is worth mentioning that AI Explanations can be used in conjunction with the What-If Tool, an open-source tool for understanding model predictions. The process for doing so is detailed in the Colab notebooks mentioned earlier.

Thank you for exploring AI Explanations with us in this didactic material. If you found this content helpful, please like and subscribe to our channel to stay updated with the latest episodes. For more information, visit Google Cloud AI Platform and start exploring AI Explanations for your predictions.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: CLOUD AI DATA LABELING SERVICE

Getting high quality data is crucial for achieving good machine learning outcomes. While public datasets are available, they may not always be suitable for custom use cases. In such situations, it becomes necessary to efficiently label your data without spending excessive time in front of a computer. This is where the cloud AI platform data labeling service can be of great help.

The data labeling service allows you to collaborate with human labelers to create labeled datasets that can be used to train machine learning models. It is particularly useful when the labels are not known at the time of data collection, such as with crowdsourced data or unstructured data like images, videos, or text.

To create a labeling task using the data labeling service, you will need three core resources: data sets, label sets, and instructions. The data set resource represents your dataset and is stored as a CSV file on Google Cloud Storage. Each row in the CSV file corresponds to an individual data element, such as an image or text file.

In addition to the CSV file, you need to provide a name for the data set and select the data type (images, video, or text). Once the data set is created, you can proceed to create a label set. A label set consists of the labels that you want the human labelers to use when labeling your data. Each label set can contain up to 100 labels, and you can mix and match label sets with different data sets depending on the model you are training.

When creating instructions for the labelers, it is important to ensure that the instructions are easy to understand, even for labelers who may not have domain knowledge. Include a list of all possible labels with descriptions for each label. Provide examples for every label, including at least three positive examples and one negative example that cover different cases. Clarify any potential edge cases, such as how to handle partially covered items in bounding boxes.

The data labeling service supports various types of labeling tasks based on the data type. For image data, you can choose from classification, bounding boxes, oriented bounding boxes, bounding polygons, polylines, or segmentation tasks. For material data, the service supports classification, object detection, object tracking, and events. For text data, you can perform classification, sentiment analysis, or entity extraction.

To ensure high labeling quality, it is recommended to request multiple human labelers to annotate each piece of data. In cases where there is disagreement among labelers, additional opinions are gathered until a consensus is reached or the maximum number of labelers has been reached. For example, in an image classification task with three labels, all three labelers will classify each image, and the majority vote will determine the final answer.

The cloud AI platform data labeling service is a valuable tool for efficiently labeling data for machine learning tasks. By leveraging human labelers and providing clear instructions, you can create high-quality labeled datasets to train your machine learning models.

The data labeling service provided by Google Cloud Platform (GCP) is a valuable tool for creating accurate data sets for machine learning training. In this service, image bounding box tasks are performed by multiple labelers. The first labeler draws the box, and the second labeler verifies it. If there is disagreement, a third labeler is involved to obtain a majority opinion.

One common concern is who will label the data. It is important to note that all data sets are labeled by officially onboarded subprocessors. For more information and details about data processing and security terms, you can refer to the GCP documentation.

When transmitting data, ensuring its security and protection is of utmost importance. Fortunately, all data stored in Google Cloud is encrypted by default, both during transit and at rest. Human labelers can only view the data during the labeling process. Furthermore, the data labeling service is HIPAA compliant, providing reassurance for those working with sensitive healthcare data.

To ensure the best results and efficient use of resources, it is recommended to ramp up data labeling jobs

incrementally. Start with a small amount of data for the first labeling job and evaluate the results. Based on this evaluation, you can revise your instructions and create subsequent jobs. This iterative process allows you to gradually increase the quantity of data to be labeled, ensuring high-quality results while optimizing time and budget.

The data labeling service offered by Google Cloud Platform is a reliable solution for accurately labeling data sets for machine learning training. By following the recommended practices and guidelines, you can leverage this service effectively and obtain high-quality results. Whether you need to use the data labeling service or not, understanding the principles of label creation and instructions is crucial for creating accurate and reliable data sets.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: INTRODUCTION TO JAX

NumPy is a fast library for numerical computations, but there are ways to make it even faster. In this material, we will explore a new library called JAX, developed by Google Research, which can significantly speed up machine learning tasks.

JAX is capable of automatically differentiating native Python and NumPy functions. It can differentiate through loops, branches, recursion, and closures, and it can even take derivatives of derivatives of derivatives. JAX supports both reverse mode differentiation (also known as back-propagation) using the grad function, as well as forward mode differentiation. These two modes can be composed arbitrarily in any order.

Aside from auto-differentiation, JAX also offers the ability to speed up code execution using a special compiler called Accelerated Linear Algebra (XLA). XLA is a domain-specific compiler for linear algebra that performs optimizations such as fusing operations together to avoid unnecessary memory writes. This allows for faster and more efficient processing. JAX leverages XLA to compile and run NumPy programs on GPUs and TPUs, achieving accelerated performance.

In addition to XLA, JAX provides other powerful features. One of them is the jit function, which allows you to just-in-time compile your own Python functions into XLA-optimized kernels. This enables maximum performance without leaving the Python environment. Another feature is pmap, which compiles a function using XLA and executes it in parallel across multiple devices, such as GPUs or TPU cores. With pmap, you can perform compilations on multiple devices simultaneously and differentiate through them all.

JAX also includes vmap, which is used for automatic vectorization. It allows you to transform a function that can handle only one data point into a function that can handle a batch of data points of any size with just a single wrapper function. This is particularly useful for tasks like training deep neural networks on large datasets.

To illustrate how these features come together, let's consider an example of training a deep neural network on the MNIST dataset. The example starts by creating utility functions to define the neural network architecture. We then demonstrate how JAX can be used to handle both single image inputs and batches of images efficiently using the vmap function. The example also shows how to use the grad and jit functions to build the remainder of the model and the training code. Finally, we demonstrate how to load the MNIST dataset using TensorFlow data sets and run the training loop.

JAX is a powerful library for machine learning that provides automatic differentiation, accelerated code execution, parallelization, and vectorization capabilities. It allows you to express sophisticated algorithms and achieve maximum performance, all within the Python ecosystem.

During the training process, we observed that it took approximately 22 seconds per epoch across 10 epochs. However, we mentioned earlier the use of the jit function. In order to incorporate this function, we added the @jit decorator at the top of our update function and renamed it as jit_update. We then re-initialized our parameters using the init_network_params function and executed the new training loop. Surprisingly, this modification significantly reduced the time required, with each epoch now taking only 8 seconds. By adding just four characters to the top of the update loop, we achieved a substantial improvement in efficiency.

It is important to note that JAX is currently a research project and not an official Google product. Consequently, it is possible to encounter bugs and other issues while using it. To assist users, the JAX team has compiled a list of common pitfalls and created a gotchas notebook. As this list is continually evolving, it is advisable to stay up-to-date with the current state of JAX if you plan on utilizing jit for your own projects.

Thank you for watching this material of "Cloud AI Adventures." If you found it informative, please show your support by clicking the Like button and subscribing to receive the latest updates. To further explore JAX, visit GitHub, where you can try it out, submit bug reports, and provide feedback to the development team.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: SETTING UP AI PLATFORM PIPELINES

AI Platform Pipelines is a powerful tool that allows you to improve the reproducibility and reliability of your data science and machine learning workflows. It addresses the growing need for MLOps, which combines machine learning and operations, by applying DevOps best practices to the machine learning ecosystem.

One of the key features of AI Platform Pipelines is the ability to orchestrate your machine learning workflows as reproducible and reusable pipelines. It does this by setting up Kubeflow Pipelines with TensorFlow Extended (TFX) on Google Kubernetes Engine. Kubeflow is an open-source toolkit for running machine learning workflows on Kubernetes, while TFX is an open-source project that allows you to define your TensorFlow-based machine learning workflows as a pipeline.

By using AI Platform Pipelines, you can take advantage of pre-built TFX components to ingest and transform data, train and evaluate models, and deploy trained models for inference, among other things. This means you don't have to build custom components for every single step of your machine learning process, making it easier to orchestrate and manage your workflows.

Setting up AI Platform Pipelines is straightforward, even for those who are not familiar with Kubernetes or DevOps. There is a dedicated dashboard in the AI Platform menu in Cloud Console where you can deploy a new pipeline. From there, you can create a Kubernetes cluster or select an existing one. Once the cluster is created, you simply need to give your pipelines instance a name and click Deploy. The necessary software will be installed automatically, abstracting away the complexity of Kubernetes.

Once deployed, you can access the Pipelines Dashboard UI, which provides a user-friendly interface for managing your pipelines. The dashboard includes a left-hand navigation bar with different categories, including Getting Started, which contains demos and materials for TFX and Kubeflow Pipelines. The Pipelines Menu is where you can see a list of all your pipelines, and you can run them by clicking on their names and selecting the Create Run button.

Each run of a pipeline produces a new set of outputs, and these runs can be organized into experiments. You can name your experiments as you wish and associate them with different types of pipeline runs. The dashboard provides a clear overview of the runs in the Experiments tab, and you can track the progress of each run as the components of the pipeline are executed.

AI Platform Pipelines is a valuable tool for improving the reproducibility and reliability of your machine learning workflows. It allows you to orchestrate your workflows as reproducible and reusable pipelines, leveraging pre-built components for common tasks. Setting up AI Platform Pipelines is straightforward, and the Pipelines Dashboard provides a user-friendly interface for managing your pipelines and tracking the progress of your runs.

In AI Platform Pipelines, you have the flexibility to include different types of pipelines within the same experiment. When running pipelines, you can generate artifacts, which can be accessed by drilling down into individual runs or by clicking the Artifacts tab in the left-hand menu. Kubeflow Pipelines currently supports various output viewers, such as confusion matrix, markdown, ROC curve, table, TensorBoard, and even a web app for more advanced visualizations.

To create your own pipeline, there are two options available: TFX SDK and Kubeflow Pipelines SDK. The choice between the two depends on your specific requirements. If you are working with TensorFlow, TFX is recommended as it is designed specifically for TensorFlow and seamlessly integrates with TensorFlow code. On the other hand, if you are using any other library, Kubeflow Pipelines is the suitable choice. It is designed to support arbitrary libraries, allowing you to incorporate scikit-learn models, PyTorch, and more into your pipeline.

TFX provides the advantage of having more pre-built components and follows a more opinionated approach. It offers a structured framework for building pipelines. On the other hand, Kubeflow Pipelines provides a more open-ended approach, allowing you to design your own pipeline components from scratch. This flexibility enables you to customize your pipeline according to your specific needs.

In future materials of AI Adventures, we will delve into more details on how to utilize these two SDKs effectively. For now, you can find two longer materials in the description below that explore AI Platform Pipelines in greater depth. These materials will provide you with a comprehensive understanding of the platform.

Thank you for watching this material of Cloud AI Adventures. If you found it informative, please consider clicking the Like button and subscribing to our channel to stay updated with the latest episodes. You can also follow me, Yufeng Guo, on Twitter @YufengG. If you are interested in longer-form machine learning and cloud content, make sure to check out the Adventures in the Cloud YouTube channel, which is linked in the description.

Now, you can start setting up your own pipeline on Google Kubernetes Engine (GKE) using AI Platform Pipelines on Google Cloud Platform. Get ready to explore the power of AI Platform Pipelines and unleash the potential of your machine learning workflows.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: AI PLATFORM OPTIMIZER

AI Platform Optimizer is a product developed by the Google AI Team that allows for hyperparameter tuning and optimization of machine-learning models. It is designed to optimize parameters such as learning rate, batch size, and other typical machine-learning hyperparameters. However, it can also optimize any evaluable system, not just machine-learning models.

For example, AI Platform Optimizer can be used to determine the most effective combination of background color, font size, and link color on a news website's subscription button, taking A/B testing to a new level. It can also be used to find the ideal buffer size and thread count to minimize computing resources for a job. Additionally, it can optimize non-food related things, although the specific use cases are not mentioned.

To use AI Platform Optimizer, there are three terms that need to be understood: study configurations, studies, and trials. A study configuration defines the optimization problem, including the result to be optimized and the parameters that affect that result. These parameters are the inputs being optimized. A study is the implementation of a study configuration, using its goal and input parameters to conduct experiments or trials. A trial refers to a specific set of input values that produce a measured outcome.

AI Platform Optimizer suggests input values for each trial but does not run the trials itself. Users are responsible for running the trials. A study continues until it reaches a preset limit of trials or until it is manually ended. As a service, AI Platform Optimizer suggests trials, records their outcomes, and uses machine learning to suggest future trials based on those outcomes. This cycle can be continued as desired.

It is worth noting that AI Platform Training also has a built-in feature called HyperTune, which handles hyperparameter tuning. HyperTune uses the same technology as AI Platform Optimizer but is specific to AI Platform Training. On the other hand, AI Platform Optimizer is a generic system that can be used to optimize any system in any location. It can be used with various platforms, including local machines, data centers, or other Google Cloud Platform services such as Compute Engine or Kubernetes Engine. Any system that can make a REST API call can utilize AI Platform Optimizer.

AI Platform Optimizer is a powerful tool for optimizing machine-learning models and other systems. It allows for the tuning of various parameters and provides suggestions for trials based on previous outcomes. Its flexibility enables its use in different environments and platforms.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: PERSISTENT DISK FOR PRODUCTIVE DATA SCIENCE

When it comes to running machine learning and data science workloads in the cloud, storage is often overlooked. However, it plays a crucial role in the performance and efficiency of these workloads. In this didactic material, we will explore different storage options on Google Cloud and discuss their trade-offs, so you can choose the right tool for your task.

One commonly used storage option on Google Cloud is Google Cloud Storage (GCS). GCS allows you to store files in buckets and access them over the network via REST API calls. It offers convenience, replication across multiple regions, and pay-as-you-go pricing based on the size of the files. GCS is especially useful for serverless environments and quick file transfers.

For higher performance, you may need to consider using persistent disks. Persistent disks provide high-performance reads and writes, and they appear as local file systems to your code. This makes it easier to migrate existing workflows into the cloud. In a Google data center, virtual machines are connected via a fast local network, allowing multiple virtual machines to connect to the same persistent disk. These disks are actually composed of slices from different hard drives scattered across the data center, enabling parallel reading and writing. Larger persistent disks offer faster I/O performance, but there is a limit to their performance improvement.

Persistent disks come in three tiers: standard, SSD, and local SSD. Standard persistent disks use spinning hard drives and are commonly used in data storage systems. SSDs offer improved read and write performance compared to standard disks. Local SSDs are co-located with the compute resource and provide the highest performance, but they come in fixed increments of 375 gigabytes. It is recommended to use local SSDs only when high-performance read and write operations are necessary.

Both standard and SSD disks can be used in zonal or regional configurations. Regional persistent disks offer high availability by allowing failover to a second zone in case of an outage in the primary zone. This is particularly useful for machine learning use cases where large datasets need to be loaded onto a persistent disk and attached in read-only mode to multiple instances.

The choice of storage option depends on your specific use case. If scalability and flexibility are important, GCS may be the right choice. If high-performance reads and writes are required, persistent disks are a better option. From small random reads to large sequential reads, the block size of the disk should be set to match the anticipated use case.

Google Cloud offers various storage options for machine learning and data science workloads. Understanding the trade-offs between these options will help you make an informed decision based on your specific requirements.

Artificial Intelligence (AI) has become an integral part of various industries, and Google Cloud offers powerful tools and platforms to harness its potential. In this educational material, we will focus on Google Cloud Machine Learning and Google Cloud AI Platform, specifically discussing the use of Persistent Disk for productive data science.

Persistent Disk is a storage option provided by Google Cloud that allows users to store and access their data reliably. It is designed to be durable, scalable, and high-performing, making it suitable for data-intensive workloads such as machine learning.

When it comes to productive data science, Persistent Disk plays a crucial role in ensuring efficient data storage and access. By using Persistent Disk, data scientists can store their datasets, models, and other important files securely. This allows for easy collaboration and reproducibility of experiments.

Persistent Disk offers various features that enhance productivity in data science workflows. One such feature is the ability to attach and detach disks to virtual machines, enabling users to easily transfer data between different instances. This flexibility allows data scientists to scale their workloads and optimize resource

allocation based on their specific needs.

In addition, Persistent Disk provides consistent and reliable performance for data-intensive tasks. It leverages Google's infrastructure to deliver high I/O throughput and low latency, ensuring fast and efficient data processing. This is particularly important for training machine learning models, where quick access to large datasets is crucial.

To further optimize performance, Persistent Disk supports different disk types, including standard and SSD-based options. Users can choose the appropriate disk type based on their workload requirements, balancing cost and performance.

Furthermore, Google Cloud provides detailed documentation and resources to help users maximize the benefits of Persistent Disk for productive data science. These resources offer insights into disk management, performance optimization techniques, and best practices for data storage and retrieval.

Persistent Disk is a valuable tool offered by Google Cloud for productive data science. Its durability, scalability, and high-performance capabilities make it an ideal choice for storing and accessing data in machine learning workflows. By leveraging Persistent Disk, data scientists can enhance collaboration, improve productivity, and achieve efficient data processing.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: TRANSLATION API

Translation API is a powerful tool provided by Google Cloud that allows developers to integrate translation capabilities into their websites and apps. With Translation API, you can make your apps instantly usable across the world in multiple languages.

To get started with Translation API, you need to enable it in your project and create a service account for authentication. Make sure to provide the Cloud Translation API editor role to your service account and generate a JSON key for authentication.

Once you have set up the authentication, you can use the Translation API to translate text. The API can automatically identify more than 100 languages and translate them. You can set both the source and target languages, but the source language is optional as the API can autodetect it.

To make a translation request, you need to pass the JSON key as a payload with an HTTP post request to the Translation API. The API will return the translated text in the desired target language.

In some cases, you may want more control over the translation of specific words or phrases, such as product names or company names. For this, Translation API provides an Advanced Glossary feature. You can create a glossary file with samples of your source and target language phrases and save it in Google Cloud Storage. Then, you can make a translation request using the glossary file, and the API will consistently translate your domain-specific terminology.

If you need to translate multiple files in multiple languages, Translation API also offers a Batch Translation feature. You can upload the files you want to translate into Google Cloud Storage, request translations in multiple languages, and store the translations in Google Cloud Storage.

Translation API is a powerful tool that allows you to easily integrate translation capabilities into your websites and apps. You can translate text, use advanced features like glossaries for domain-specific terminology, and perform batch translations for multiple files and languages.

EITC/AI/GCML GOOGLE CLOUD MACHINE LEARNING DIDACTIC MATERIALS
LESSON: GOOGLE CLOUD AI PLATFORM
TOPIC: AUTOML TRANSLATION

Translation is an important aspect of global communication, especially in fields like machine learning and artificial intelligence (AI). In this episode, we will explore the concept of custom translation models and how they can be used to power global apps.

Custom translation models are particularly useful when dealing with specialized terminology and concepts that may not be captured accurately by standard translation APIs. For example, if you were to work in a Spanish-speaking country like Peru, you would need a translation model that can effectively communicate technical terms related to machine learning and AI with Spanish-speaking developers.

This is where AutoML Translation comes into play. AutoML Translation allows you to create your own custom translation models that are specific to your domain. For instance, if you run a financial-reporting platform that needs to translate time-sensitive documents in real-time for markets like Peru, India, and France, AutoML Translation can automate the translation process for over a hundred language pairs, enabling you to enter new markets quickly.

While the Translation API is great for general-purpose text and allows you to provide glossaries of specific words or phrases, AutoML Translation excels in cases where you don't have a glossary or need more control over word translations. It bridges the gap between generic translation tasks and specific niche vocabularies.

To create a custom translation model with AutoML Translation, you need high-quality data that covers the vocabulary, usage, and grammatical quirks of your industry or area of focus. Once you have the data, you can use the AutoML Translation UI to create a data set by selecting the source and target languages, uploading the files from your computer, and storing them in Cloud Storage.

After preparing the data set, you can train the model, which may take several hours depending on the amount of data. Once the training is complete, you can evaluate the model using metrics such as the BLEU (Bilingual Evaluation Understudies) score, which compares the machine translation to a ground-truth translation. A higher BLEU score indicates a better overlap between human and machine output.

Finally, you can test your model using the Predict tab in the AutoML Translation UI. You can compare the results from your custom model to the Google NMT (Neural Machine Translation) model to see the difference.

AutoML Translation is a powerful tool that allows you to create high-quality, production-ready, custom translation models quickly. It enables accurate and efficient translation of specialized content, making it an essential tool for global applications in various industries.

Federated learning is an important concept in the field of Artificial Intelligence (AI) that allows for collaborative training of machine learning models without sharing raw data. In this didactic material, we will explore the concept of federated learning and its significance in the development of AI systems.

Federated learning is a distributed approach to training machine learning models. Instead of centralizing data in a single location, federated learning enables the training process to be performed on local devices or edge servers. This decentralized approach ensures data privacy and security, as sensitive information remains on the device or server where it is generated.

The key idea behind federated learning is that instead of sending raw data to a central server, only model updates or gradients are shared. These updates are calculated locally on each device or server using locally stored data. The server aggregates the updates from multiple devices or servers and computes a global update. This global update is then sent back to the devices or servers, and the process iterates until the model converges.

Federated learning offers several advantages over traditional centralized training approaches. Firstly, it addresses privacy concerns by keeping data locally and minimizing the risk of data breaches. Secondly, it allows for training on devices with limited computational resources, as the majority of the computation is performed

locally. This is particularly useful in scenarios where data may be sensitive or where there are bandwidth constraints.

Google Cloud Machine Learning and Google Cloud AI Platform provide tools and services to support federated learning. One such tool is AutoML Translation, which enables the development of machine translation models using federated learning techniques. AutoML Translation allows users to train models on their own data while keeping it secure and private.

Federated learning is a powerful approach in the field of AI that enables collaborative training of machine learning models while preserving data privacy. It allows for training on local devices or edge servers, minimizing the need for centralized data storage. Google Cloud Machine Learning and Google Cloud AI Platform offer tools like AutoML Translation that leverage federated learning techniques to develop secure and efficient machine translation models.