Quickstart

Setup

Get up and running with Actions! In this quickstart you will learn the basics to get up and running with Actions and Giza Platform. You will create a Giza Workspace, initialize your first Action, generate a deployment and create executions with Actions Runs

Before you begin, make sure you have all the necessary libraries installed:

```

Copy pipinstallgiza-actions pipxinstallgiza-cli

```

From your terminal, create a Giza user through our CLI in order to access the Giza Platform:

```

Copy gizauserscreate

```

After creating your user, log into Giza:

```

Copy gizauserslogin

```

Optional : you can create an API Key for your user in order to not regenerate your access token every few hours.

```

Copy gizauserscreate-api-key

```

To create Actions Runs you will need a Giza Workspace, create it executing the following command in your terminal:

```

Copy gizaworkspacescreate

```

Create your first Action

You can use our Giza CLI to initialize an Action project:

```

Copy gizaactionsnew

```

After submitting the required parameters you will get a directory with the following structure:

```

Copy gizaactionsnewmnist_example cdexample tree

```
. ├——README.md ├——mnist_example | ├——__init__.py | ├——predict_cairo_action.py | ├——predict_onnx_action.py | ├——train_action.py | └——utils.py ├——pyproject.toml ├——tests | └——__init__.py └——zero.jpg
```

This structure represents a ready-to-use Python project to work with Giza Actions based on MNIST Tutorial .
```

An example of how to define an action can be found inmnist_example/train_action.py . This script is responsible for creating yourAction (workflow) for training your neural network and deploying the action on your GizaWorkspace .

```

Copy

# mnist_example/train_action.py

importnumpyasnp importtorch importtorch.nnasnn importtorch.onnx importtorch.optimasoptim importtorchvision fromgiza_actions.actionimportAction,action fromgiza_actions.taskimporttask fromscipy.ndimageimportzoom fromtorch.utils.dataimportDataLoader,TensorDataset

device=torch.device("cuda"iftorch.cuda.is_available()else"cpu")

# Neural Network Definition

input_size=196# 14x14 hidden_size=10 num_classes=10 num_epochs=10 batch_size=256 learning_rate=0.001

classNeuralNet(nn.Module): def__init__(self,input_size,hidden_size,num_classes): super(NeuralNet, self).**init**() self.input_size=input_size self.l1=nn.Linear(input_size, hidden_size) self.relu=nn.ReLU() self.l2=nn.Linear(hidden_size, num_classes)

defforward(self,x): out=self.l1(x) out=self.relu(out) out=self.l2(out) returnout

# Task Definitions

defresize_images(images): returnnp.array([zoom(image[0], (0.5,0.5))forimageinimages])

@task(name="Prepare Datasets") defprepare_datasets(): print("Prepare dataset...") train_dataset=torchvision.datasets.MNIST(root="./data", train=True, download=True) test_dataset=torchvision.datasets.MNIST(root="./data", train=False)

x_train=resize_images(train_dataset) x_test=resize_images(test_dataset)

x_train=torch.tensor(x_train.reshape(-1,14*14).astype("float32")/255) y_train=torch.tensor([labelfor_, labelintrain_dataset], dtype=torch.long)

x_test=torch.tensor(x_test.reshape(-1,14*14).astype("float32")/255) y_test=torch.tensor([labelfor_, labelintest_dataset], dtype=torch.long)

print("✅ Datasets prepared successfully")

returnx_train,y_train,x_test,y_test

@task(name="Create Loaders") defcreate_data_loaders(x_train,y_train,x_test,y_test): print("Create loaders...")

train_loader=DataLoader( TensorDataset(x_train, y_train), batch_size=batch_size, shuffle=True ) test_loader=DataLoader( TensorDataset(x_test, y_test), batch_size=batch_size, shuffle=False )

print("✅ Loaders created!")

returntrain_loader,test_loader

@task(name="Train model") deftrain_model(train_loader): print("Train model...")

model=NeuralNet(input_size, hidden_size, num_classes).to(device) criterion=nn.CrossEntropyLoss() optimizer=optim.Adam(model.parameters(), lr=learning_rate)

forepochinrange(num_epochs): fori,(images,labels)inenumerate(train_loader): images=images.to(device).reshape(-1,14*14) labels=labels.to(device)

outputs=model(images) loss=criterion(outputs, labels)

optimizer.zero_grad() loss.backward() optimizer.step()

if(i+1)%100==0: print( f"Epoch [{epoch+1}/{num_epochs}], Step [{i+1}/{len(train_loader)}], Loss:{loss.item():.4f}" )
```

print("✅ Model trained successfully") returnmodel

@task(name="Test model") deftest_model(model,test_loader): print("Test model...") withtorch.no_grad(): n_correct=0 n_samples=0 forimages,labelsintest_loader: images=images.to(device).reshape(-1,14*14) labels=labels.to(device) outputs=model(images) _,predicted=torch.max(outputs.data,1) n_samples+=labels.size(0) n_correct+= (predicted==labels).sum().item()

acc=100.0*n_correct/n_samples print(f"Accuracy of the network on the 10000 test images:{acc}%")

@task(name="Convert To ONNX") defconvert_to_onnx(model,onnx_file_path): dummy_input=torch.randn(1, input_size).to(device) torch.onnx.export(model, dummy_input, onnx_file_path, export_params=True, opset_version=10, do_constant_folding=True,)

print(f"Model has been converted to ONNX and saved as{onnx_file_path}")

@action(name="Action: Convert To ONNX", log_prints=True) defexecution(): x_train,y_train,x_test,y_test=prepare_datasets() train_loader,test_loader=create_data_loaders(x_train, y_train, x_test, y_test) model=train_model(train_loader) test_model(model, test_loader)

Convert to ONNX

onnx_file_path="mnist_model.onnx" convert_to_onnx(model, onnx_file_path)

if__name__=="**main**": action_deploy=Action(entrypoint=execution, name="pytorch-mnist-action") action_deploy.serve(name="pytorch-mnist-deployment")

```

Now you are set to create your first Action deployment! Execute the Action file to deploy it.

```

Copy python mnist_example/train_action.py

```

Your deployment is ready for executions! You can create Actions Runs directly from the GizaWorkspace . To learn more aboutcreating an Action Run check our linked guided overview.

To deploy the other actions in themnist_example directory in your workspace, you must first create a model on the Giza platform. To better understand the process, please refer to ourstep-by-step tutorial .

What's next?

Congrats! Now that you've completed the Actions quickstart, check out our guides and learn how to do more specific things in ourhow-to-guides andtutorials . If you're interested in learning more about Actions core concepts, grab a cup of coffee and take a look at ourConceptual Guides !

Previous Installation Next Contribution Guidelines

Last updated1 month ago
```