AWS Elastic Inference

For PyTorch

Kris Skrinak

2020-04-21

The AWS ML Stack

Broadest and most complete set of Machine Learning capabilities

AI SERVICES

VISION	SPEECH		TEXT			SEARCH	CHATBOTS	PERSONALIZATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS
Amazon Rekognition	Amazon Polly	Amazon Transcribe *+Medical*	Amazon Comprehend *+Medical*	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Contact Lens *For Amazon Connect*

ML SERVICES

Amazon SageMaker

Ground Truth

AWS Marketplace for ML

SageMaker Studio IDE

Built-in algorithms | Notebooks | Experiments | Processing | Model training & tuning | Debugger | Autopilot | Model hosting | Model Monitor

Neo

Augmented AI

ML FRAMEWORKS & INFRASTRUCTURE

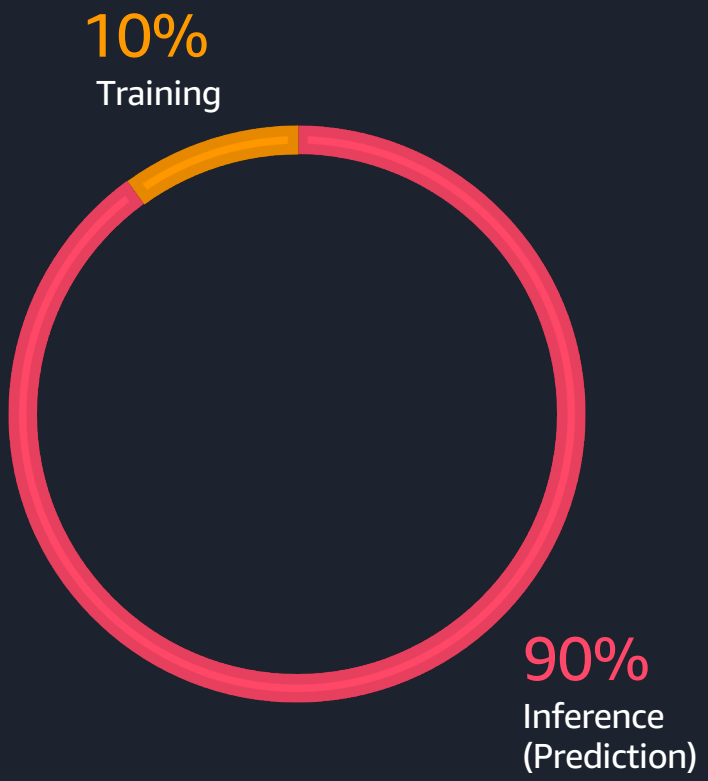
TensorFlow | mxnet | PYTORCH

GLUON | Keras | DeepGraphLibrary

Deep Learning AMIs & Containers | GPUs & CPUs | Elastic Inference | Inferentia | FPGA

aws

Predictions drive complexity and cost in production



aws machine learning

Amazon Elastic Inference

Reduce deep learning inference costs up to 75%

Lower inference costs

Match capacity to demand

Available between 1 to 32 TFLOPS per accelerator

Integrated with Amazon EC2 and Amazon SageMaker

AUTODESK.

Support for TensorFlow, Apache MXNet

PyTorch (coming soon)

EAGLEVIEW TECHNOLOGIES

Single and mixed-precision operations

gumgum

If you are using EI for the first time

Dependencies must be set up

- AWS PrivateLink VPC Endpoints
- IAM policies, and
- Security group rules

For convenience you may want to use the AWS Deep Learning AMI

No worries!

There’s a setup script on GitHub that does most of the heavy lifting for you.

aws/amazon-elastic-inference-tools

Before your run ensure…

- Python 3 is installed on your the machine where you expect to run the tool.
- Install the AWS SDK for Python (Boto3).
- You've set the correct Amazon VPC to the Region where you are launching the instance (could be your default VPC).
- You've setup the Subnet where you'd like to launch the instance.
- You have the EC2 Key Pair.
- You have AWS credentials. (duh)

Important

- You must convert your PyTorch models into TorchScript and use the Elastic Inference API for inference.

TorchScript

PyTorch's use of dynamic computational graphs presents unique challenges for production model deployment

TorchScript bridges this gap by providing the ability to compile and export models to a Python-free graph-based representation


```
def forward(
    self,
    tokens: torch.Tensor,
    seq_lens: torch.Tensor,
    dict_feat: Tuple[torch.Tensor, torch.Tensor, torch.Tensor],
    actions: List[List[int]],
    contextual_token_embeddings: torch.Tensor,
    beam_size: int = 1,
    top_k: int = 1,
) -> List[Tuple[torch.Tensor, torch.Tensor]]:
    actions_idx = actions[0]
    assert len(actions_idx) > 0, "actions must be provided for training"

    token_embeddings = self.embedding(
        tokens, dict_feat, contextual_token_embeddings
    )
    beam = [self.gen_init_state(tokens, token_embeddings)]
    all_finished = False
    while not all_finished:
        # Stores plans for expansion as (score, state, action)
        plans : List[Plan] = []
        all_finished = True
        # Expand current beam states
        for state in beam:
            # Keep terminal states
            if state.finished():
                plans.append(Plan(state.neg_prob, const.TERMINAL_ELEMENT, state))
            else:
                all_finished = False
                plans.extend(self.gen_plans(state))
```

```
beam = [self.gen_init_state(tokens, token_embeddings)]
all_finished = False
while not all_finished:
    # Stores plans for expansion as (score, state, action)
    plans : List[Plan] = []
    all_finished = True
    # Expand current beam states
    for state in beam:
        # Keep terminal states
        if state.finished():
            plans.append(Plan(state.neg_prob, const.TERMINAL_ELEMENT, state))
        else:
            all_finished = False
            plans.extend(self.gen_plans(state))

    beam.clear()
    # Take actions to regenerate the beam
    plans.sort()
    for plan in plans[:beam_size]:
        beam.append(self.execute_plan(plan, actions_idx, beam_size))

beam.sort()
res = jit.annotate(List[Tuple[torch.Tensor, torch.Tensor]], [])
for state in beam[:top_k]:
    res.append(
        (
            torch.tensor([state.predicted_actions_idx]),
            # Unsqueeze to add batch dimension
            torch.cat(state.action_scores).unsqueeze(0),
        )
    )
return res
```

```
@torch.jit.script
class Plan:
    def __init__(self, score: float, action: int, state: ParserState):
        self.score = score
        self.action = action
        self.state = state

    def __lt__(self, other):
        # type: (Plan) -> bool
        return self.score < other.score
```

```
beam = [self.gen_init_state(tokens, token_embeddings)]
all_finished = False
while not all_finished:
    # Stores plans for expansion as (score, state, action)
    plans : List[Plan] = []
    all_finished = True
    # Expand current beam states
    for state in beam:
        # Keep terminal states
        if state.finished():
            plans.append(Plan(state.neg_prob, const.TERMINAL_ELEMENT, state))
        else:
            all_finished = False
            plans.extend(self.gen_plans(state))

    beam.clear()
    # Take actions to regenerate the beam
    plans.sort()
    for plan in plans[:beam_size]:
        beam.append(self.execute_plan(plan, actions_idx, beam_size))

beam.sort()
res = jit.annotate(List[Tuple[torch.Tensor, torch.Tensor]], [])
for state in beam[:top_k]:
    res.append(
        (
            torch.tensor([state.predicted_actions_idx]),
            # Unsqueeze to add batch dimension
            torch.cat(state.action_scores).unsqueeze(0),
        )
    )
```

Run the script

```
$ aws configure
$ python amazonei_setup.py --region us-west-2 --instance-type m5.xlarge
```

Read all about it

bit.ly/LaunchEI *(that's a capital i, as in inference)*

bit.ly/PTwEI

bit.ly/CheapPT

bit.ly/TSandTE

Hands On

TorchServe and TorchElastic Announced

techcrunch.com

TC

Join Extra Crunch

Login

Search

Startups
Videos
Audio
Newsletters
Extra Crunch
Advertise
Events

More

AWS and Facebook launch an open-source model server for PyTorch

Frederic Lardinois @fredericl / 9:30 am PDT • April 21, 2020

Comment

Read our latest COVID-19 coverage, and find out more about our events strategy.

Thank You!

Kris Skrinak / skrinak@amazon.com
@skrinak

 |

aws machine learning