Deep Learning AMI

Guia do desenvolvedor

Deep Learning AMI: Guia do desenvolvedor

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

O que é a AWS Deep Learning AMI?

Bem-vindo ao Guia do usuário da AWS Deep Learning AMI.

A AWS Deep Learning AMI (DLAMI) é o único local para deep learning na nuvem. Essa instância de máquina personalizada está disponível na maioria das regiões do Amazon EC2 para vários tipos de instância, de uma instância pequena de apenas CPU até instâncias de várias GPUs com alta potência. Ela é fornecida pré-configurada com o NVIDIA CUDA e o NVIDIA cuDNN, bem como com as versões mais recentes das estruturas de deep learning mais populares.

Sobre este guia

Este guia ajudará você a iniciar e usar a DLAMI. Ele abrange vários casos de uso que são comuns em deep learning, para treinamento e inferência. A escolha da AMI correta para o objetivo e o tipo de instâncias de sua preferência também é abordada. A DLAMI é fornecida com vários tutoriais para cada uma das estruturas. Você encontrará instruções sobre como configurar o Jupyter para executar os tutoriais em seu navegador.

Pré-requisitos

Você deve estar familiarizado com ferramentas de linha de comando e o Python básico para executar a DLAMI com êxito. Tutoriais sobre como usar cada estrutura são fornecidos pelas próprias estruturas, no entanto, este guia pode mostrar como ativar cada uma delas e encontrar os tutoriais apropriados para começar a usar.

Usos de exemplo da DLAMI

Aprender sobre deep learning: a DLAMI é uma ótima escolha para o aprendizado ou o ensino de estruturas de aprendizagem de máquina e deep learning. Ela acaba com a dor de cabeça de solucionar problemas das instalações de cada estrutura e de fazê-las funcionar em conjunto no mesmo computador. A DLAMI é fornecida com um Jupyter notebook e facilita a execução dos tutoriais fornecidos pelas estruturas para pessoas não familiarizadas com aprendizagem de máquina e deep learning.

Desenvolvimento de aplicativos: se você for um desenvolvedor de aplicativos e estiver interessado em usar deep learning para fazer com que seus aplicativos usem os últimos avanços de AI, a DLAMI é o campo de teste perfeito para você. Cada estrutura é fornecida com tutoriais sobre como começar a usar o deep learning, e muitas têm vários modelos que facilitam testar o deep learning sem precisar criar as redes neurais você mesmo ou fazer qualquer treinamento de modelo. Alguns exemplos mostram como criar um aplicativo de detecção de imagem em apenas alguns minutos, ou como criar um aplicativo de reconhecimento de voz para seu próprio chatbot.

Aprendizagem de máquina e análise de dados: se você for um cientista de dados ou estiver interessado em processar seus dados com deep learning, descobrirá que muitas das estruturas são compatíveis com R e Spark. Você encontrará tutoriais sobre como fazer regressões simples, até a criação de sistemas de processamento de dados escaláveis para sistemas de personalização e previsões.

Pesquisa: se você for um pesquisador e quiser experimentar uma nova estrutura, testar um novo modelo ou treinar novos modelos, a DLAMI e os recursos da AWS para escala podem aliviar o problema de

instalações tediosas e o gerenciamento de vários nós de treinamento. Você pode usar os modelos do EMR e do AWS CloudFormation para lançar facilmente um cluster repleto de instâncias prontas para uso para treinamento escalável.

> **Note**
>
> Embora a escolha inicial possa ser atualizar seu tipo de instância para um tipo de instância maior com mais GPUs (até 8), você também pode escalar horizontalmente criando um cluster de instâncias de DLAMI. Para configurar rapidamente um cluster, você pode usar o modelo predefinido do AWS CloudFormation. Confira Recursos e suporte (p. 36) para obter mais informações sobre criações de clusters.

Recursos da DLAMI

Estruturas pré-instaladas

No momento, há três tipos principais da DLAMI com outras variações relacionadas ao sistema operacional (SO) e versões de software:

- Deep Learning AMI com Conda (p. 5) – estruturas instaladas separadamente usando pacotes do `conda` e ambientes separados do Python
- Deep Learning AMI com código-fonte (p. 6) – estruturas instaladas da origem em conjunto no mesmo ambiente do Python
- Deep Learning Base AMI (p. 6) – nenhuma estrutura instalada. Somente o NVIDIA CUDA e outras dependências

A nova Deep Learning AMI com Conda usa ambientes Anaconda para isolar cada estrutura, para que você possa alternar entre eles conforme sua necessidade e não se preocupar com suas dependências conflitantes. A Deep Learning AMI com código-fonte tem todas as estruturas de deep learning instaladas no mesmo ambiente do Python, junto com a origem das estruturas.

Para obter mais informações sobre como selecionar a melhor DLAMI para você, visite Conceitos básicos (p. 4).

Esta é a lista completa de estruturas compatíveis entre Deep Learning AMI com Conda e Deep Learning AMI com código-fonte:

- Apache MXNet
- Caffe
- Caffe2
- CNTK
- Keras
- PyTorch
- TensorFlow
- Theano**
- Torch*

> **Note**
>
> * Disponível somente na Deep Learning AMI com código-fonte.
> ** O Theano está sendo desativado, uma vez que ele não é mais um projeto ativo.

Software de GPU pré-instalado

Mesmo que você use somente uma instância de CPU, a DLAMI terá o NVIDIA CUDA e o NVIDIA cuDNN. O software instalado é o mesmo, independentemente do tipo de instância. Lembre-se de que as ferramentas específicas à GPU só funcionam em uma instância que tenha pelo menos uma GPU. Mais informações sobre isso são fornecidas em Seleção do tipo de instância para DLAMI (p. 8).

- NVIDIA CUDA 9
- NVIDIA cuDNN 7
- CUDA 8 também está disponível. Consulte Instalações do CUDA e associações da estrutura (p. 7) para obter mais informações.

Fornecimento e visualização do modelo

A Deep Learning AMI com Conda vem pré-instalada com dois tipos de servidores modelo, um para MXNet e um para TensorFlow, bem como TensorBoard, para visualizações de modelos.

- Servidor modelo MXNet (p. 32)
- TensorFlow Serving (p. 33)
- TensorBoard (p. 34)

Conceitos básicos

Como começar a usar a DLAMI

É fácil começar. Neste guia, estão incluídas dicas de como selecionar a DLAMI correta para você com a seleção de um tipo de instância que se adequa a seu caso de uso e a seu orçamento, e Recursos e suporte (p. 36) que descreve configurações personalizadas que podem ser de seu interesse. Ou você pode ir diretamente para a última DLAMI no AWS Marketplace, a Deep Learning AMI com Conda (p. 5), e ativar uma instância a partir dela.

Se você não estiver familiarizado com o uso da AWS ou do Amazon EC2, comece com a Deep Learning AMI com Conda (p. 5). Se estiver familiarizado com o Amazon EC2 e outros serviços da AWS, como o Amazon EMR, o Amazon EFS ou o Amazon S3, e estiver interessado em integrar esses serviços para projetos que precisam de treinamento distribuído ou inferência, confira Recursos e suporte (p. 36) para ver o que se adequa a seu caso de uso.

Se estiver familiarizado com o uso de AMIs e desejar ativar uma agora, acesse o AMI Marketplace agora. Mas, primeiro, recomendamos conferir Escolher sua DLAMI (p. 4) para ter uma ideia do tipo de instância que pode ser melhor para seu aplicativo.

Cadastre-se na AWS

Se você ainda não tiver uma conta da AWS, primeiro use as etapas para cadastrar-se e, em seguida, retorne para Escolher sua DLAMI (p. 4) para continuar.

Cadastre-se para uma conta AWS

1. Abra o https://aws.amazon.com/ e escolha em Criar uma conta da AWS.

   Note

   Isso pode estar indisponível no seu navegador se, anteriormente, você iniciou a sessão no Console de gerenciamento da AWS. Nesse caso, escolha Fazer login com uma conta diferente e, em seguida, Criar uma nova conta da AWS.

2. Siga as instruções online.

   Parte do procedimento de cadastro envolve uma chamada telefônica e a digitação de um PIN usando o teclado do telefone.

Próxima etapa

Escolher sua DLAMI (p. 4)

Escolher sua DLAMI

Se você executar uma pesquisa por deep learning no AMI Marketplace, descobrirá que existem muitas opções e que não está claro qual é a mais adequada para seu caso de uso. Esta seção ajuda você a decidir. Quando nos referimos a uma DLAMI, muitas vezes estamos nos referindo a um grupo de AMIs

centralizadas em torno de um tipo ou de uma funcionalidade comum. Há três variáveis que definem esses tipos e/ou funcionalidade:

- Conda versus Fonte versus Base
- CUDA 8 versus CUDA 9
- Amazon Linux versus Ubuntu versus Windows

O restante dos tópicos deste guia ajuda você a obter mais informações e mais detalhes.

Tópicos

A seguir

Deep Learning AMI com Conda (p. 5)

Deep Learning AMI com Conda

A DLAMI mais recente usa ambientes virtuais de Anaconda. Esses ambientes são configurados para manter as instalações de estruturas diferentes separadas. Também facilita a alternância entre estruturas. Isso é ótimo para aprendizado e testes com todas as estruturas que a DLAMI tem para oferecer. A maioria dos usuários acreditam que a nova Deep Learning AMI com Conda é perfeita para eles.

Essas AMIs "Conda" serão as DLAMIs principais. Elas serão atualizadas frequentemente com as versões mais recentes das estruturas, e terão o software e os drivers de GPU mais recentes. Geralmente, ela é referenciada como a AWS Deep Learning AMI na maioria dos documentos.

- Essa DLAMI tem as seguintes estruturas: Apache MXNet, Caffe, Caffe2, CNTK, Keras, PyTorch, TensorFlow e Theano

Estabilidade versus novas tecnologias

As AMIs do Conda usam os lançamentos formais mais recentes de cada estrutura usando os pacotes PIP em PyPI. Versões "release candidate" e recursos experimentais não devem ser esperados. As versões CUDA e cuDNN corresponderão com tudo que tiver suporte do release oficial. Se você está interessado em compilações mais atualizadas das estruturas criadas diretamente da origem, você pode querer o Deep Learning AMI com CUDA 9 (p. 41) que oferece suporte a CUDA 9 e cuDNN 7. Ou, se você deseja realizar a instalação da origem usando opções de compilação personalizadas ou otimizadas, a de Deep Learning Base AMI (p. 6) pode ser a melhor opção para você.

CUDA Support

A versão CUDA da Deep Learning AMI com Conda e as estruturas compatíveis com cada um (o suporte a Keras depende de cada estrutura):

- CUDA 9: Apache MXNet, Caffe2, PyTorch, Theano
- CUDA 8: CNTK, TensorFlow e Caffe

Os números de versões de uma estrutura em específico podem ser encontrados no

Escolha este tipo de DLAMI ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

- Deep Learning AMI com Conda (p. 39)

A seguir

Deep Learning AMI com código-fonte (p. 6)

Deep Learning AMI com código-fonte

A Deep Learning AMI com código-fonte faz referência às DLAMIs que usam um ambiente Python para todas as estruturas. Tecnicamente, há dois ambientes, o Python 2.7 e o Python 3.5, e as estruturas são instaladas uma vez em cada uma dessas versões do Python.

Essas AMIs contêm compilações mais atualizadas de estruturas criadas diretamente da origem, com otimizações avançadas que não são encontradas em pacotes PIP existentes já oferecidos pelas estruturas. Ela também tem o código-fonte para as estruturas, o que faz dela uma imagem de disco maior. Ela é usada com frequência por alguém que planeja modificar o código-fonte diretamente ou por colaboradores de projetos de código aberto das estruturas.

- A Deep Learning AMI com código-fonte (CUDA 9) é composta por: Apache MXNet, Caffe2, Keras, PyTorch e TensorFlow
- Deep Learning AMI com código-fonte (CUDA 8) é composta por: Apache MXNet, Caffe, Caffe2, CNTK, Keras, TensorFlow, Theano e Torch

Os números de versões de uma estrutura em específico podem ser encontrados no

Escolha este tipo de DLAMI ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

- Deep Learning AMI com código-fonte (p. 40)

A seguir

Deep Learning Base AMI (p. 6)

Deep Learning Base AMI

A Deep Learning Base AMI é como uma tela vazia para deep learning. Ela vem com tudo o que você precisa até o ponto da instalação de uma determinada estrutura. Ela terá sua opção de CUDA 8 ou CUDA 9. Esse grupo de AMIs é útil para colaboradores de projeto que desejam usar um projeto de deep learning e criar o mais recente. É para alguém que deseja implementar seu próprio ambiente com a confiança de que o software NVIDIA mais recente está instalado e funcionando para que possa concentrar-se em escolher as estruturas e versões que deseja instalar.

> **Note**
>
> Lembre-se de que você pode facilmente clonar a origem e criar na Deep Learning AMI com Conda ou na Deep Learning Base AMI, de modo que apenas ter acesso à origem não deve fazer você se apoiar na AMI de origem.

Escolha este tipo de DLAMI ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

- Deep Learning Base AMI (p. 40)

A seguir

Instalações do CUDA e associações da estrutura (p. 7)

Instalações do CUDA e associações da estrutura

O deep learning é todo de última geração, no entanto, cada estrutura oferece versões "estáveis". Essas versões estáveis podem não funcionar com a implementação e os recursos mais recentes do CUDA ou do cuDNN. Como você decide? Em última análise, isso vai depender do seu caso de uso e os recursos de que você precisa. Se você não tem certeza, use a Deep Learning AMI com Conda (p. 39) mais recente. Ela tem binários de PIP oficiais de todas as estruturas com CUDA 8 e CUDA 9, usando a versão mais recente que for compatível com cada estrutura.

Consulte o nosso guia em Estabilidade versus novas tecnologias (p. 5) para obter mais orientações.

CUDA Support

A versão CUDA da Deep Learning AMI com código-fonte e as estruturas compatíveis com cada uma:

- Deep Learning AMI com CUDA 9 (p. 41)Apache MXNet, Caffe2, PyTorch, TensorFlow
- Deep Learning AMI com CUDA 8 (p. 41): Apache MXNet, Caffe, Caffe2, CNTK, PyTorch, Theano, TensorFlow e Torch

Os números de versões de uma estrutura em específico podem ser encontrados no

Escolha este tipo de DLAMI ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

Escolha uma das versões do CUDA e analise a lista completa das DLAMIs que têm essa versão no Apêndice, ou saiba mais sobre as diferentes DLAMIs com a opção a seguir.

- Deep Learning AMI com CUDA 9 (p. 41)
- Deep Learning AMI com CUDA 8 (p. 41)

A seguir

Opções do sistema operacional da DLAMI (p. 7)

Opções do sistema operacional da DLAMI

As DLAMI s são oferecidas nos sistemas operacionais listados abaixo. Se estiver mais familiarizado com o CentOS ou o RedHat, você ficará mais confortável com as Versões da AWS Deep Learning AMI, Amazon Linux (p. 42). Caso contrário, você pode considerar que a Versões do Ubuntu da AWS Deep Learning AMI, (p. 42) é mais de sua preferência.

Se você precisar do Windows como seu sistema operacional, terá duas opções encontradas aqui: Versões da AWS Deep Learning AMI, do Windows (p. 43).

Escolha um dos sistemas operacionais e analise a lista completa no Apêndice ou vá para as próximas etapas para escolher sua AMI e tipo de instância.

- Versões da AWS Deep Learning AMI, Amazon Linux (p. 42)
- Versões do Ubuntu da AWS Deep Learning AMI, (p. 42)
- Versões da AWS Deep Learning AMI, do Windows (p. 43)

Conforme mencionado na visão geral de Conceitos básicos, você tem algumas opções para acessar uma DLAMI, mas primeiro você deve avaliar o tipo de instância de que você precisa. Você também deve identificar a região que irá usar.

A seguir

Seleção do tipo de instância para DLAMI (p. 8)

Seleção do tipo de instância para DLAMI

A seleção do tipo de instância pode ser outro desafio, mas facilitaremos isso para você com algumas dicas sobre como escolher o melhor. Lembre-se de que a DLAMI é gratuita, mas que os recursos de computação subjacentes não são.

- Se você não estiver familiarizado com o deep learning, provavelmente desejará uma instância de "nível básico" com uma única GPU.
- Se você tiver um orçamento limitado, precisará começar de forma mais econômica e examinar as instâncias de CPU apenas.
- Se estiver interessado em executar um modelo treinado para inferência e previsões (e não treinamento), você poderá desejar uma instância de CPU com muita memória ou até um cluster desses para serviços de alto volume.
- Se você estiver interessado treinar um modelo com grande quantidade de dados, talvez você deseje uma instância maior ou até mesmo um cluster de instâncias de GPU.

Os DLAMIs não estão disponíveis em todas as regiões, mas é possível copiar DLAMIs para a região de sua escolha. Consulte Cópia de uma AMI para obter mais informações. Cada região oferece suporte a uma diferente variedade de tipos de instância e, muitas vezes, um tipo de instância tem um custo um pouco diferente em regiões diferentes. Em cada página principal da DLAMI, você verá uma lista de custos de instâncias. Observe a lista de seleções de regiões e escolha uma região que esteja próxima de você ou de seus clientes. Se você planeja usar mais de uma DLAMI e potencialmente criar um cluster, certifique-se de usar a mesma região para todos os nós do cluster.

- Para obter mais detalhes sobre instâncias, consulte Tipos de instância do EC2.
- Para obter mais informações sobre regiões, visite Regiões do EC2.

Portanto, com todos esses pontos em mente, anote o tipo de instância que melhor se aplique a seu caso de uso e orçamento. O restante dos tópicos deste guia ajudam você a obter mais informações e detalhes.

Tópicos

A seguir

Instâncias de GPU recomendadas (p. 8)

Instâncias de GPU recomendadas

Uma instância de GPU é recomendada para a maioria dos objetivos de deep learning. O treinamento de novos modelos será mais rápido em uma instância de GPU do que em uma instância de CPU. Você pode dimensionar de forma sublinear quando tem instâncias de várias GPUs ou usa treinamento distribuído em muitas instâncias com GPUs.

- As Instâncias P3 do Amazon EC2 têm até 8 GPUs NVIDIA Tesla V100.
- As Instâncias P2 do Amazon EC2 têm até 16 GPUs NVIDIA K80.

- As Instâncias G3 do Amazon EC2 têm até 4 GPUs NVIDIA Tesla M60.
- Confira os Tipos de instância do EC2 e escolha Computação acelerada para ver as diferentes opções de instâncias de GPU.

> **Note**
>
> O tipo de instância P3 ainda não é compatível para AMIs do Windows.

A seguir

Instâncias de CPU recomendadas (p. 9)

Instâncias de CPU recomendadas

Independentemente de você ter um orçamento, estar aprendendo sobre o deep learning ou apenas querer executar um serviço de previsão, você tem muitas opções acessíveis na categoria de CPU.

- As Instâncias C5 do Amazon EC2 têm até 72 vCPUs Intel.
- As instâncias C4 do Amazon EC2 têm até 36 vCPUs Intel.
- Confira os Tipos de instância do EC2 e procure Compute Optimized para ver as diferentes opções de instâncias de CPU.

> **Note**
>
> As instâncias C5 (não disponíveis em todas as regiões) são excelentes para modelagem científica, processamento em lotes, análise distribuída, computação de alto desempenho (HPC) e inferência de machine/deep learning.
>
> **Note**
>
> - Se você planeja usar o Caffe em um Deep Learning AMI com código-fonte (CUDA 8), escolha uma instância de GPU no lugar delas. O Caffe no Deep Learning AMI com código-fonte (CUDA 8) foi criado com suporte a GPU e não pode ser executado no modo CPU.

A seguir

Definição de preço para a DLAMI (p. 9)

Definição de preço para a DLAMI

A DLAMI é gratuita, no entanto, você ainda é responsável pelos custos do Amazon EC2 ou de outros serviços da AWS. As estruturas de deep learning incluídas são gratuitas, e cada uma tem suas próprias licenças de código aberto. O software de GPU da NVIDIA é gratuito e também tem licenças próprias.

Como isso é gratuito, mas não gratuito? Quais são os custos do "Amazon EC2" ou de outros serviços da AWS?

Essa é uma pergunta comum. Alguns tipos de instância no Amazon EC2 são identificados como gratuitos. Essas normalmente são instâncias menores, e é possível executar a DLAMI em uma dessas instâncias gratuitas. Isso significa que isso é totalmente gratuito quando você usa essa capacidade de instância. Se você decidir que deseja uma instância mais robusta, com mais núcleos de CPU, mais espaço em disco, mais RAM e uma ou mais GPUs, esses requisitos provavelmente não estarão na classe de instâncias da camada gratuita. Isso significa que você precisará pagar por esses custos. Uma maneira fácil de ver isso é que o software ainda é gratuito, mas você precisa pagar pelo hardware subjacente que está usando.

Consulte Seleção do tipo de instância para DLAMI (p. 8) para obter mais informações sobre qual tamanho de instância escolher e quais tipos de coisas você pode esperar de cada tipo.

Próxima etapa

Iniciar e configurar uma DLAMI (p. 11)

Iniciar e configurar uma DLAMI

Se você está aqui, você já deve ter uma boa ideia de qual AMI deseja iniciar. Caso contrário, escolha uma DLAMI usando as diretrizes de seleção de AMI encontradas em todo o Conceitos básicos (p. 4) ou use a lista completa de AMIs na seção Apêndice, Opções do AMI (p. 39).

Você também deve saber qual tipo de instância e região escolherá. Caso contrário, procure Seleção do tipo de instância para DLAMI (p. 8).

> **Note**
>
> Usaremos p2.xlarge como o tipo de instância padrão nos exemplos. Basta substituir por qualquer tipo de instância que você tenha em mente.

Tópicos

Etapa 1: Iniciar uma DLAMI

> **Note**
>
> Para esta demonstração, podemos fazer referências específicas à Deep Learning AMI (Ubuntu). Mesmo que selecione outra DLAMI, você deve ser capaz de seguir este guia.

Iniciar a instância

1. Você tem algumas rotas para iniciar uma DLAMI. Escolha uma:
   - Console do EC2 (p. 12)
   - Pesquisa no Marketplace (p. 12)

   > **Tip**
   >
   > Opção da CLI: se você optar por ativar uma DLAMI usando a AWS CLI, precisará do ID da AMI, da região e do tipo de instância e das informações do token de segurança. Verifique se você tem a AMI e os IDs de instância prontos. Se você ainda não tiver configurado a AWS CLI, faça isso primeiro usando o guia para Instalar a interface de linha de comando da AWS.
2. Depois de concluir as etapas de uma dessas opções, aguarde até que a instância esteja pronta. Isso normalmente leva apenas alguns minutos. Você pode verificar o status da instância no Console do EC2.

Console do EC2

1. Abra o Console do EC2.
2. Observe sua região atual na navegação superior. Se essa não for a região da AWS desejada, altere essa opção antes de continuar. Para obter mais informações, consulte Regiões do EC2.
3. Escolha Executar instância.
4. Procure a instância desejada por nome:
   1. Escolha AWS Marketplace ou...
   2. Escolha QuickStart. Apenas um subconjunto das DLAMI disponíveis serão listadas aqui.
   3. Pesquisar `AWS Deep Learning AMI`. Além disso, procure o subtipo, como o sistema operacional desejado, e se você deseja de base, Conda, de origem etc.
   4. Navegue pelas opções e clique em Select na sua escolha.
5. Reveja os detalhes e escolha Continue.
6. Escolha um tipo de instância.
7. Escolha Review and Launch.
8. Reveja os detalhes e a definição de preço. Escolha Executar.

> Tip
>
> Confira Conceitos básicos do deep learning com a Deep Learning AMI da AWS para uma demonstração com capturas de tela!

Próxima etapa

Etapa 2: Conectar-se à DLAMI (p. 12)

Pesquisa no Marketplace

1. Navegue pelo AWS Marketplace e pesquise AWS Deep Learning AMI.
2. Navegue pelas opções e clique em Select na sua escolha.
3. Reveja os detalhes e escolha Continue.
4. Reveja os detalhes e anote a Region. Se essa não for a região da AWS desejada, altere essa opção antes de continuar. Para obter mais informações, consulte Regiões do EC2.
5. Escolha um tipo de instância.
6. Escolha um par de chaves, use o seu padrão ou crie um novo.
7. Reveja os detalhes e a definição de preço.
8. Escolha Launch with 1-Click.

Próxima etapa

Etapa 2: Conectar-se à DLAMI (p. 12)

Etapa 2: Conectar-se à DLAMI

Conecte-se à DLAMI que você iniciou em um cliente (Windows, MacOS ou Linux). Para obter mais informações, consulte Conectar-se à instância do Linux no Guia do usuário do Amazon EC2 para instâncias do Linux.

Mantenha uma cópia do comando login do SSH à mão se desejar configurar o Jupyter depois de fazer login. Você usará uma variação dele para conectar-se à página da web do Jupyter.

> Tip
>
> Se você executar uma instância do Amazon Linux, a caixa de diálogo de conexão terá "root@" no comando de login do SSH. Ele deve ser substituído por "ec2-user@".

Próxima etapa

Etapa 4: Testar a DLAMI (p. 13)

Etapa 3: proteger sua instância de DLAMI

Sempre mantenha o sistema operacional e outros softwares instalados atualizados executando patches e atualizações assim que forem disponibilizados.

Se você está usando o Amazon Linux ou Ubuntu, ao efetuar login na DLAMI, você será notificado se houver atualizações disponíveis e verá as instruções para atualização. Para obter mais informações sobre a manutenção do Amazon Linux, consulte . Para instâncias do Ubuntu, consulte a Documentação oficial do Ubuntu.

No Windows, verifique o Windows Update regularmente para instalar atualizações de software e de segurança. Se você preferir, as atualizações podem ser instaladas automaticamente.

> Important
>
> Para obter mais informações sobre as vulnerabilidades Spectre e Meltdown e como corrigir seu sistema operacional para lidar com elas, consulte o Boletim de segurança da AWS-2018-013.

Etapa 4: Testar a DLAMI

Dependendo da sua versão de DLAMI, você terá diferentes opções de teste:

- Deep Learning AMI com Conda (p. 5) – vá para Como usar a Deep Learning AMI com Conda (p. 22).
- Deep Learning AMI com código-fonte (p. 6) – consulte a documentação de conceitos básicos da estrutura desejada.
- Deep Learning Base AMI (p. 6) – consulte a documentação de instalação da estrutura desejada.

Você também pode criar um notebook Jupyter, experimentar tutoriais ou começar a codificação em Python. Para obter mais informações, consulte Configurar um servidor de notebook Jupyter (p. 14).

Limpar

Quando não precisar mais da DLAMI, você poderá interrompê-la ou fechá-la para evitar cobranças contínuas. A interrupção de uma instância a manterá disponível para que você possa retomá-la posteriormente. Suas configurações, arquivos e outras informações não voláteis estão sendo armazenados em um volume no Amazon S3. Será cobrada uma pequena taxa do S3 para reter o volume enquanto a instância estiver interrompida, mas os recursos de computação não serão cobrados enquanto

ela estiver no estado interrompido. Quando você iniciar a instância novamente, ela montará esse volume e seus dados estarão disponíveis. Se você encerrar uma instância, ela será perdida e você não poderá iniciá-la novamente. Seus dados ainda residirão no S3, portanto, para evitar cobranças adicionais você precisará excluir o volume também. Para obter mais instruções, consulte Encerre sua instância no Guia do usuário do Amazon EC2 para instâncias do Linux.

Configurar um servidor de notebook Jupyter

O Jupyter Notebook é um aplicativo Web que permite gerenciar documentos de bloco de notas usando um navegador da web.

Para configurar um notebook Jupyter, você:

- Configura o servidor de notebook Jupyter em sua instância do Amazon EC2.
- Configura o cliente para que possa conectar-se ao servidor de notebook Jupyter. Fornecemos instruções de configuração para Windows, macOS e clientes Linux.
- Testa a configuração fazendo login no servidor de notebook Jupyter.

Para obter mais informações sobre notebooks Jupyter, consulte Jupyter.

Tópicos

Configurar o notebook Jupyter em sua DLAMI

Para usar o Jupyter, você precisa abrir a porta 8888 (ou uma porta de sua escolha) em sua instância do firewall. Você também pode configurar um certificado SSL e uma senha padrão, mas isso é opcional. Quando a porta for aberta, você iniciará o servidor e, em seguida, executará o SSH para o servidor e criará um túnel para acessar a interface da web do Jupyter. As etapas a seguir demonstram esse processo.

Abra o painel do EC2 e escolha Security Groups na barra de navegação do EC2 na seção Network & Security. Haverá um ou mais security groups na lista dessa página. Localize o mais recente (a descrição tem uma data e hora), selecione-o, escolha a guia Inbound e, em seguida, clique em Edit. Em seguida, clique em Add Rule. Isso adiciona uma nova linha. Preencha os campos com as seguintes informações:

Type: **`Custom TCP Rule`**

Protocol: **`TCP`**

Port Range: **`8888`**

Source: **`Anywhere`** (**`0.0.0.0/0,::/0`**)

> Tip
>
> Para que o Jupyter execute mais perfeitamente, você pode modificar o arquivo de configuração do Jupyter. No arquivo de configuração, você define alguns dos valores a serem usados para a

autenticação na web, incluindo o caminho do arquivo do certificado SSL e uma senha. Você pode passar pela Configuração personalizada de SSL e do servidor (p. 15) agora ou mais tarde.

Próxima etapa

Inicie o servidor de notebook Jupyter (p. 16)

Configuração personalizada de SSL e do servidor

Aqui configuramos um notebook Jupyter com SSL e um arquivo de configuração.

Conecte-se à instância do Amazon EC2 e, em seguida, conclua o procedimento a seguir.

Configurar o servidor do Jupyter

1. Crie um certificado SSL.

```
$ cd
$ mkdir ssl
$ cd ssl
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout "cert.key" -out
 "cert.pem" -batch
```

2. Crie uma senha. Você usa essa senha para fazer login no servidor de notebook Jupyter em seu cliente para que possa acessar notebooks no servidor.

   a. Abra o terminal do iPython.

   ```
   $ ipython
   ```

   No prompt do iPython, execute o comando `passwd()` para definir a senha.

   ```
   iPythonPrompt> from IPython.lib import passwd
   iPythonPrompt> passwd()
   ```

   Você obtém o hash da senha (por exemplo, `sha1:examplefc216:3a35a98ed...`).

   b. Registre o hash da senha.

   c. Saia do terminal do iPython.

   ```
   $ exit
   ```

3. Edite o arquivo de configuração do Jupyter.

   Localize `jupyter_notebook_config.py` no diretório `~/.jupyter`.

4. Atualize o arquivo de configuração para armazenar sua senha e as informações do certificado SSL.

   a. Abra o arquivo .config.

   ```
   $ vi ~/.jupyter/jupyter_notebook_config.py
   ```

   b. Cole o texto a seguir no final do arquivo. Você precisará fornecer o hash da senha.

   ```
   c = get_config()  # Get the config object.
   c.NotebookApp.certfile = u'/home/ubuntu/ssl/cert.pem' # path to the certificate we
    generated
   c.NotebookApp.keyfile = u'/home/ubuntu/ssl/cert.key' # path to the certificate key
    we generated
   ```

```
c.IPKernelApp.pylab = 'inline'  # in-line figure when using Matplotlib
c.NotebookApp.ip = '*'  # Serve notebooks locally.
c.NotebookApp.open_browser = False  # Do not open a browser window by default when
 using notebooks.
c.NotebookApp.password = 'sha1:fc216:3a35a98ed980b9...'
```

Note

Se você estiver usando uma DLAMI que não tenha um arquivo de configuração padrão do Jupyter, será necessário criar um.

```
$ jupyter notebook --generate-config
```

Depois de criado, siga as mesmas etapas para atualizar a configuração com as informações de SSL.

Isso conclui a configuração do servidor do Jupyter.

Próxima etapa

Inicie o servidor de notebook Jupyter (p. 16)

Inicie o servidor de notebook Jupyter

Agora você pode acionar o servidor fazendo login na instância e executando os comandos a seguir.

```
$ source activate python3
$ jupyter notebook
```

Se você tiver aberto uma porta diferente de 8888, atualize o parâmetro de porta para a porta aberta. Com o servidor iniciado, agora você pode se conectar a ele por meio de um túnel SSH em seu computador cliente. Quando o servidor for executado, você verá alguma saída do Jupyter confirmando que o servidor está em execução. Nesse ponto, ignore o texto informando que você pode acessar o servidor por meio de uma URL do host local, pois isso não funcionará em uma instância em nuvem.

Note

O motivo pelo qual você precisa para ativar um ambiente do Python 3 é que interface do Jupyter espera o Python 3 e não é totalmente compatível com o Python 2. Não se preocupe se você quiser usar uma estrutura com o Python 2. O Jupyter resolverá isso para você quando você alternar as estruturas usando a interface da web do Jupyter. Você pode encontrar mais informações sobre este assunto em Alternar ambientes com o Jupyter (p. 27).

Próxima etapa

Configurar o cliente para se conectar ao servidor do Jupyter (p. 16)

Configurar o cliente para se conectar ao servidor do Jupyter

Depois de configurar o cliente para se conectar ao servidor de notebook Jupyter, você pode criar e acessar notebooks no servidor em sua área de trabalho e executar o código de deep learning no servidor.

Para obter informações de configuração, escolha um dos links a seguir.

Tópicos

Configurar um cliente Windows

Preparação

Certifique-se de que você tem as seguintes informações, que são necessárias para configurar o túnel SSH:

- O nome DNS público de sua instância do Amazon EC2. Você pode localizar o nome DNS público no console do EC2.
- O par de chaves do arquivo de chave privada. Para obter mais informações sobre como acessar seu par de chaves, consulte Pares de chaves do Amazon EC2 no Guia do usuário do Amazon EC2 para instâncias do Linux.

Configurar o PuTTY

As instruções passo a passo a seguir explicam como conectar-se à sua instância do EC2 e configurar um túnel SSH usando o PuTTY, um cliente SSH gratuito para Windows. Se você receber um erro ao tentar conectar-se à instância, consulte Resolução de problemas para se conectar à sua instância. Como pré-requisito, faça download e instale o PuTTY de Página de download do PuTTY. Se você já tiver uma versão mais antiga do PuTTY instalada, recomendamos fazer download da versão mais recente. Instale o pacote inteiro.

1. Para conectar-se à instância do EC2 usando o PuTTY, você primeiro precisa converter seu arquivo de chave privada (.pem) gerado pelo Amazon EC2 em um formato reconhecido pelo PuTTY (.ppk). Você pode localizar as instruções para criar o arquivo .ppk em Conectar à instância Linux no Windows utilizando PuTTY. Pesquise por "Converter sua chave privada usando o PuTTYgen" nesse tópico.
2. Agora, abra o PuTTY e navegue para Session em Category no painel à esquerda. Insira as seguintes informações:

   Tipo de conexão: SSH

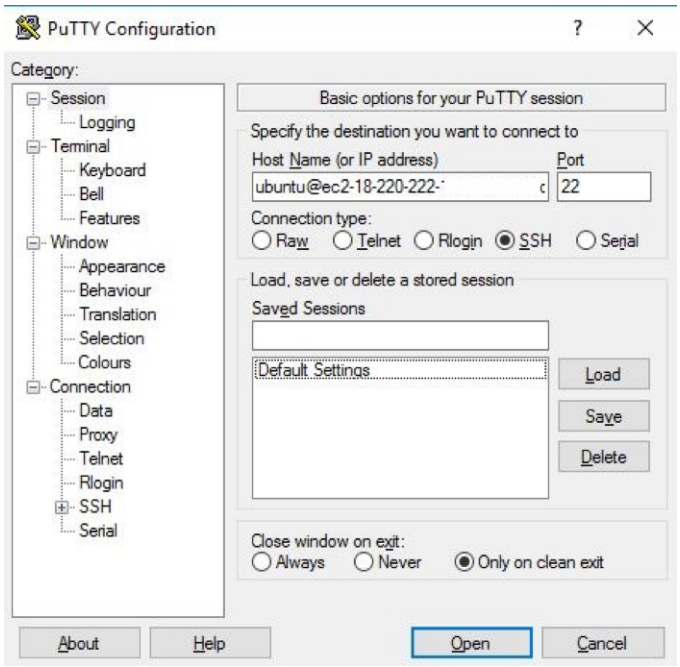
   Nome doNome do host: **ubuntu@*YourInstancePublicDNS***

   Porta: 22

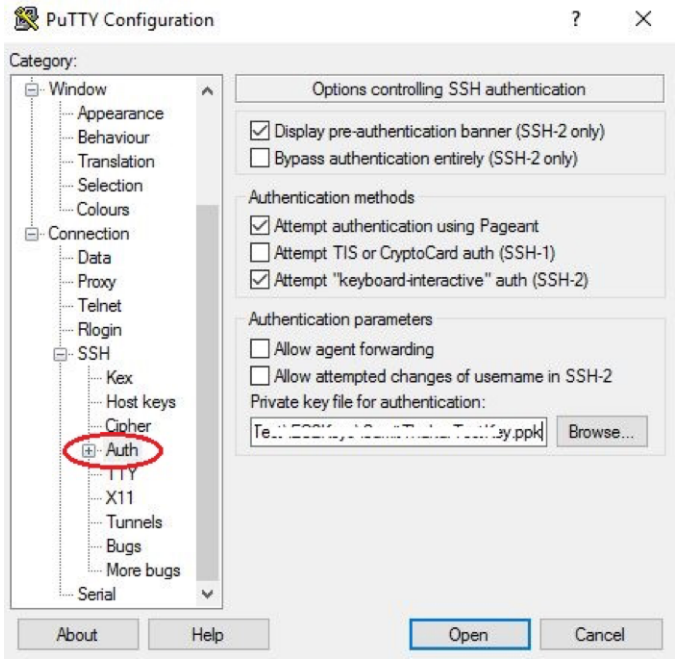
   Especifique o nome de usuário apropriado para sua AMI. Por exemplo:

   - Para uma instância da AMI do Amazon Linux, o nome do usuário padrão é ec2-user.
   - Para uma AMI do RHEL, o nome do usuário é ec2-user ou root.
   - Para uma AMI do Ubuntu, o nome do usuário é ubuntu ou root.
   - Para uma AMI do Centos, o nome do usuário é centos.
   - Para uma AMI do Fedora, o nome do usuário é ec2-user.
   - Para SUSE, o nome do usuário é ec2-user ou root.

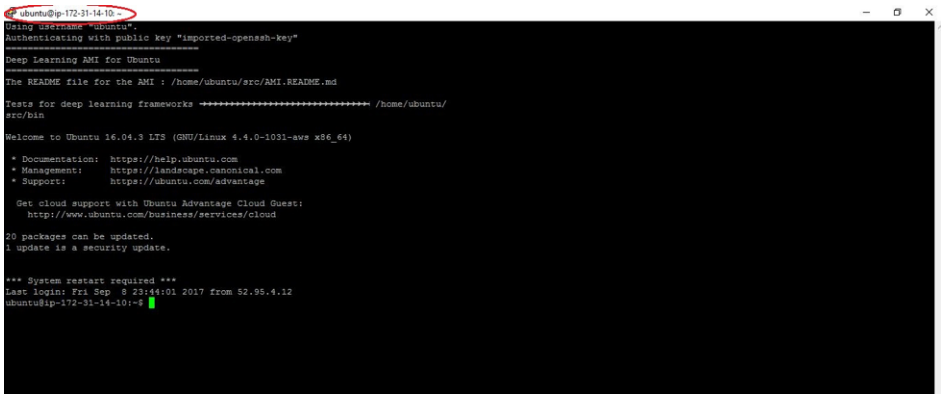
   Caso contrário, se o ec2-user e o root não funcionarem, verifique com o provedor da AMI.



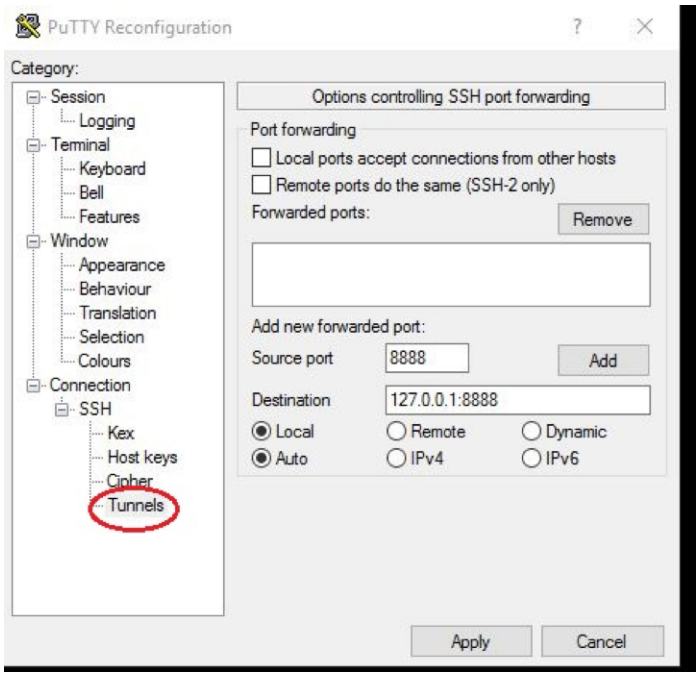
3. Agora, expanda Connection no painel à esquerda. Navegue até Auth sob SSH. Procure e adicione o arquivo .ppk criado na Etapa 1 anterior. Em seguida, clique em Open na parte inferior da tela.



4. Se essa for a primeira vez que você se conecta a essa instância, o PuTTY exibirá uma caixa de diálogo de alerta de segurança perguntando se você confia no host ao qual está se conectando. Escolha Sim. Uma janela se abrirá e você estará conectado à sua instância.
5. Agora, para configurar o túnel SSH para sua instância do EC2, clique com o botão direito do mouse no canto superior esquerdo da janela da instância, conforme destacado a seguir. Ele abre um menu suspenso. Selecione Change Settings no menu para exibir a tela PuTTY Reconfiguration.



6. Navegue até Tunnels sob SSH no painel à esquerda. Preencha a Source port e Destination, conforme mostrado a seguir. Selecione Local e Auto como opções de encaminhamento de porta. Finalmente, clique em Add seguido por Apply para concluir a configuração do túnel SSH.



As instruções simplificadas acima oferecem uma forma rápida e fácil para conectar-se à sua instância do EC2 usando o PuTTY. Se quiser saber mais sobre o tópico, você pode examinar nosso Conecte-se à sua instância do Linux no Guia do Windows.

Próxima etapa

Teste fazendo login no servidor de notebook Jupyter (p. 20)

Configurar um cliente Linux

1. Abra um terminal.
2. Execute o comando a seguir para encaminhar todas as solicitações da porta local 8157 para a porta 8888 em sua instância do Amazon EC2 remota. Atualize o comando substituindo *`ec2-###-##-##-###.compute-1.amazonaws.com`* pelo nome DNS público de sua instância do EC2.

```
$ ssh -i ~/mykeypair.pem -L 8157:127.0.0.1:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

Esse comando abre um túnel entre o cliente e a instância do EC2 remota que está executando o servidor de notebook Jupyter. Depois de executar o comando, você pode acessar o servidor de notebook Jupyter em **`https://127.0.0.1:8157`**.

Próxima etapa

Teste fazendo login no servidor de notebook Jupyter (p. 20)

Configurar um cliente MacOS

1. Abra um terminal.
2. Execute o comando a seguir para encaminhar todas as solicitações da porta local 8157 para a porta 8888 em sua instância do Amazon EC2 remota. Atualize o comando substituindo *`ec2-###-##-##-###.compute-1.amazonaws.com`* pelo nome DNS público de sua instância do EC2 ou apenas use o endereço IP público.

```
$ ssh -i ~/mykeypair.pem -L 8157:127.0.0.1:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

Esse comando abre um túnel entre o cliente e a instância do EC2 remota que está executando o servidor de notebook Jupyter. Depois de executar o comando, você pode acessar o servidor de notebook Jupyter em **`http://127.0.0.1:8157`**.

Próxima etapa

Teste fazendo login no servidor de notebook Jupyter (p. 20)

Teste fazendo login no servidor de notebook Jupyter

Agora, você está pronto para fazer login no servidor de notebook Jupyter.

A próxima etapa é testar a conexão ao servidor por meio de seu navegador.

1. Na barra de endereço do navegador, digite a seguinte URL.
   - Para clientes macOS e Linux, digite a URL a seguir.

     ```
     http://127.0.0.1:8157
     ```

   - Para clientes Windows, use localhost ou o nome DNS público da instância do Amazon EC2 e o número da porta do Jupyter. Geralmente, a porta do Jupyter é 8888.

     ```
     http://127.0.0.1:8888
     ```

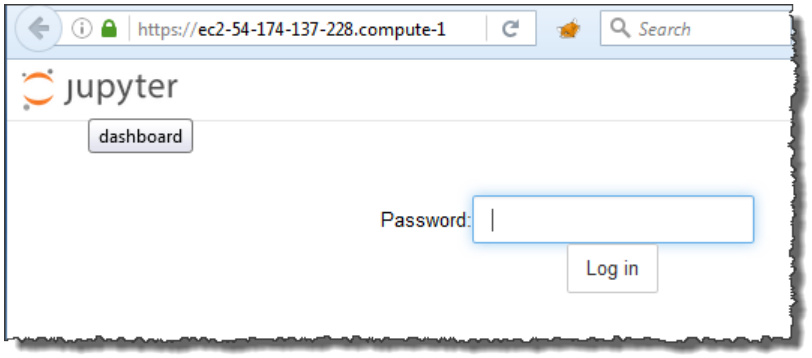
> **Note**
>
> `https` funciona somente se você passou pela etapa extra de Configuração personalizada de SSL e do servidor (p. 15). Esses exemplos usam `http`, mas assim que seu certificado SSL é configurado, você pode mudar para `https`.

2. Se a conexão for bem-sucedida, você verá a página da web do servidor de notebook Jupyter. Neste ponto, pode ser solicitado uma senha ou um token. Se você fez uma configuração simples sem configurar o Jupyter, o token será exibido na janela de terminal onde você iniciou o servidor. Procure por algo como:

   ```
   Copy/paste this URL into your browser when you connect for
   the first time, to login with a token: http://localhost:8888/?
   token=0d3f35c9e404882eaaca6e15efdccbcd9a977fee4a8bc083
   ```

   Copie o token (a série longa de dígitos), nesse caso seria `0d3f35c9e404882eaaca6e15efdccbcd9a977fee4a8bc083`, e use-o para acessar o servidor de notebook Jupyter.

   Por outro lado, se você tiver editado a configuração do Jupyter com um Configuração personalizada de SSL e do servidor (p. 15), digite a senha criada ao configurar o servidor de notebook Jupyter.

   > **Note**
   >
   > Por padrão, a tela de login do Jupyter solicita um token. No entanto, se você configurar uma senha, o prompt solicitará uma senha.

   Agora você tem acesso ao servidor de notebook Jupyter que está em execução na DLAMI. Você pode criar novos notebooks ou executar os Tutoriais e exemplos (p. 22) fornecidos.

Tutoriais e exemplos

Tópicos

As seções a seguir descrevem como a Deep Learning AMI com Conda pode ser usada para alternar ambientes, executar o código de exemplo de cada uma das estruturas e como executar o Jupyter para que você possa testar diferentes tutoriais de notebook.

A seguir: Início rápido ao alternar entre estruturas com o conda

Como usar a Deep Learning AMI com Conda (p. 22)

Como usar a Deep Learning AMI com Conda

Tópicos

Introdução à Deep Learning AMI com Conda

O Conda é um sistema de gerenciamento de pacotes e de ambientes de código aberto que é executado no Windows, macOS e Linux. O Conda instala, executa e atualiza pacotes e suas dependências rapidamente. O Conda cria, salva, carrega e alterna facilmente entre ambientes em seu computador local.

A Deep Learning AMI com Conda foi configurada para que você alterne facilmente entre ambientes de deep learning. As instruções a seguir orientam você na execução de alguns comandos básicos com `conda`. Elas também ajudam você a verificar se a importação básica da estrutura está funcionando e se você pode executar algumas operações simples com a estrutura. Em seguida, você pode continuar com

tutoriais mais completos fornecidos com a DLAMI ou os exemplos de estruturas encontrados em cada site de projeto das estruturas.

Etapa 1: Fazer login na DLAMI

Depois que você faz login em seu servidor, você verá uma "mensagem do dia" (MOTD) do servidor que descreve vários comandos do conda que você pode usar para alternar entre as diferentes estruturas de deep learning.

```
=============================================================================
       __|  __|_  )
       _|  (     /   Deep Learning AMI  (Ubuntu)
      ___|\___|___|
=============================================================================

Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1039-aws x86_64v)

Please use one of the following commands to start the required environment with the
 framework of your choice:
for MXNet(+Keras1) with Python3 (CUDA 9) _______________________ source activate mxnet_p36
for MXNet(+Keras1) with Python2 (CUDA 9) _______________________ source activate mxnet_p27
for TensorFlow(+Keras2) with Python3 (CUDA 8) ________________ source activate
 tensorflow_p36
for TensorFlow(+Keras2) with Python2 (CUDA 8) ________________ source activate
 tensorflow_p27
for Theano(+Keras2) with Python3 (CUDA 9) _______________________ source activate theano_p36
for Theano(+Keras2) with Python2 (CUDA 9) _______________________ source activate theano_p27
for PyTorch with Python3 (CUDA 8) ______________________________ source activate pytorch_p36
for PyTorch with Python2 (CUDA 8) ______________________________ source activate pytorch_p27
for CNTK(+Keras2) with Python3 (CUDA 8) _________________________ source activate cntk_p36
for CNTK(+Keras2) with Python2 (CUDA 8) _________________________ source activate cntk_p27
for Caffe2 with Python2 (CUDA 9) ________________________________ source activate caffe2_p27
for base Python2 (CUDA 9) _______________________________________ source activate python2
for base Python3 (CUDA 9) _______________________________________ source activate python3
```

Cada comando do conda tem o seguinte padrão:

```
source activate framework_python-version
```

Por exemplo, você pode ver `for MXNet(+Keras1) with Python3 (CUDA 9) _____ source activate mxnet_p36`, o que significa que o ambiente tem o MXNet, o Keras 1, o Python 3 e o CUDA 9. E para ativar esse ambiente, o comando que você deve usar é:

```
$ source activate mxnet_p36
```

A outra variação desse comando seria:

```
$ source activate mxnet_p27
```

Isso significa que o ambiente terá o MXNet e o Python 2 (com o Keras 1 e o CUDA 9).

Etapa 2: Iniciar o ambiente do MXNet Python 3

Vamos testar o MXNet primeiro para que você tenha uma ideia geral de como ele é fácil.

- Ative o ambiente virtual do MXNet para Python 3.

```
$ source activate mxnet_p36
```

Isso ativa o ambiente para MXNet com Python 3. Como alternativa, você pode ativar o mxnet_p27 para obter um ambiente com Python 2.

Etapa 3: Testar um código do MXNet

Para testar a instalação, use o Python para gravar código do MXNet que cria e imprime uma matriz usando a API `NDArray`. Para obter mais informações, consulte a API NDArray.

1. Inicie o terminal iPython.

   ```
   (mxnet_p36)$ ipython
   ```

2. Importe o MXNet.

   ```
   import mxnet as mx
   ```

   Você pode ver uma mensagem de aviso sobre um pacote de terceiros. Você pode ignorá-la.

3. Crie uma matriz 5x5, uma instância do `NDArray` com elementos inicializados como 0. Imprima a matriz.

   ```
   mx.ndarray.zeros((5,5)).asnumpy()
   ```

   Verifique o resultado.

   ```
   array([[ 0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.]], dtype=float32)
   ```

   Você pode localizar mais exemplos do MXNet na seção de tutoriais do MXNet.

Etapa 4: Alternar para o ambiente do TensorFlow

Agora vamos alternar para o TensorFlow! Se você ainda estiver no console do iPython, use `quit()` e, em seguida, prepare-se para alternar os ambientes.

1. Ative o ambiente virtual do TensorFlow para Python 3,

   ```
   $ source activate tensorflow_p36
   ```

2. Inicie o terminal iPython.

   ```
   (tensorflow_36)$ ipython
   ```

3. Execute um programa rápido do TensorFlow.

   ```
   import tensorflow as tf
   ```

```
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Você deve ver "Hello, Tensorflow!" Você testou duas estruturas de deep learning diferentes e viu como alternar entre as estruturas.

Tip

Consulte as notas de release para obter informações sobre problemas conhecidos:

- Problemas conhecidos da Deep Learning AMI (Ubuntu) (p. 64)
- Problemas conhecidos da Deep Learning AMI (Amazon Linux) (p. 60)

A seguir

Executar os tutoriais do notebook Jupyter (p. 25)

Usar a Deep Learning Base AMI

Usar a Deep Learning Base AMI

A AMI Base é fornecida com uma plataforma base de drivers de GPU e bibliotecas de aceleração para implantar seu próprio ambiente personalizado de deep learning. Por padrão, a AMI é configurada com o ambiente NVidia CUDA 9. No entanto, você também pode alternar para um ambiente CUDA 8 reconfigurando a variável de ambiente `LD_LIBRARY_PATH`. Você simplesmente precisa substituir a parte do CUDA 9 da sequência da variável de ambiente pela CUDA 8 equivalente.

Configurar as versões do CUDA

Parte do CUDA 9 da sequência de LD_LIBRARY_PATH (instalado por padrão)

```
... :/usr/local/cuda-9.0/lib64:/usr/local/cuda-9.0/extras/CUPTI/lib64:/lib/
nccl/cuda-9 :...restante do valor de LD_LIBRARY_PATH
```

Substitua por CUDA 8

```
... :/usr/local/cuda-8.0/lib64:/usr/local/cuda-8.0/extras/CUPTI/lib64:/lib/
nccl/cuda-9 :...restante do valor de LD_LIBRARY_PATH
```

Tip

Consulte as notas de release para obter informações sobre problemas conhecidos:

- Problemas conhecidos da Deep Learning Base AMI (Amazon Linux) (p. 48)

Executar os tutoriais do notebook Jupyter

Tutoriais e exemplos são fornecidos com cada origem dos projetos de deep learning e, na maioria dos casos, eles são executados em qualquer DLAMI. Se você escolher a Deep Learning AMI com Conda (p. 5), obterá o benefício adicional de alguns tutoriais escolhidos a dedo já configurados e prontos para serem testados.

Para executar os tutoriais instalados no notebook Jupyter na DLAMI, você precisará Configurar um servidor de notebook Jupyter (p. 14).

Assim que o servidor do Jupyter estiver em execução, você poderá executar os tutoriais por meio de seu navegador da web. Se estiver executando a Deep Learning AMI com Conda ou se tiver configurado ambientes do Python, você poderá alternar os kernels do Python na interface do notebook Jupyter. Selecione o kernel apropriado antes de tentar executar um tutorial específico à estrutura. Exemplos adicionais são fornecidos para os usuários da Deep Learning AMI com Conda.

Como uma rápida recapitulação de como inicializar o Jupyter e se conectar, execute o seguinte em sua instância.

```
$ source activate python3
$ jupyter notebook
```

Em seguida, execute o seguinte localmente em seu cliente macOS ou Linux. No Windows, consulte as instruções detalhadas em Configurar o PuTTY (p. 17).

```
$ ssh -i ~/mykeypair.pem -L 8157:127.0.0.1:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

Se desejar testar outros tutoriais, basta baixá-los nessa pasta e executá-los no Jupyter.

> **Note**
>
> Muitos tutoriais exigem módulos adicionais do Python que podem não estar configurados na DLAMI. Se você receber um erro, como `"xyz module not found"`, faça login na DLAMI, ative o ambiente conforme descrito acima e, em seguida, instale os módulos necessários.

> **Tip**
>
> Com frequência, os tutoriais e exemplos de deep learning dependem de uma ou mais GPUs. Se seu tipo de instância não tiver uma GPU, talvez seja necessário alterar um pouco o código de exemplo para que ele seja executado.

Navegar pelos tutoriais instalados

Depois de fazer login no servidor do Jupyter e poder ver o diretório dos tutoriais, serão apresentadas pastas de tutoriais para cada nome de estrutura. Se você não vir uma estrutura listada, os tutoriais não estarão disponíveis para essa estrutura em sua DLAMI atual. Clique no nome da estrutura para ver os tutoriais listados e, em seguida, clique em um tutorial para iniciá-lo.

Na primeira vez que executa um notebook na Deep Learning AMI com Conda, você precisará informar qual ambiente deseja usar. Você poderá selecionar de uma lista. Cada ambiente é denominado de acordo com este padrão:

`Environment (conda_framework_python-version)`

Por exemplo, você pode ver `Environment (conda_mxnet_p36)`, o que significa que o ambiente tem o MXNet e o Python 3. A outra variação disso seria `Environment (conda_mxnet_p27)`, o que significa que o ambiente tem o MXNet e o Python 2.

> **Tip**
>
> Se você está preocupado sobre qual versão do CUDA está ativa, uma maneira de ver isso é no MOTD, quando você faz login pela primeira vez na DLAMI.

Alternar ambientes com o Jupyter

Se você decidir experimentar um tutorial para uma estrutura diferente, certifique-se de verificar o kernel em execução. Essas informações podem ser vistas no canto superior direito da interface do Jupyter, logo abaixo do botão de logout. Você pode alterar o kernel em qualquer notebook clicando no item de menu Kernel do Jupyter, em seguida, em Change Kernel e, em seguida, clicando no ambiente que atende ao notebook que você está executando.

Neste ponto, você precisará executar novamente todas as células porque uma alteração no kernel apagará o estado de qualquer coisa que tenha executado anteriormente.

> Tip
>
> Alternar entre as estruturas pode ser divertido e instrutivo, no entanto, você pode esgotar a memória. Se você começar a receber erros, examine a janela de terminal que tem o servidor do Jupyter em execução. Há mensagens úteis e um log de erros aqui, e você poderá ver um erro de falta de memória. Para corrigir isso, você pode ir para a página inicial do servidor do Jupyter, clicar na guia Running e, em seguida, em Shutdown para cada um dos tutoriais que provavelmente ainda estão em execução em segundo plano e consumindo toda a sua memória.

A seguir

Para obter mais exemplos e código de exemplo de cada estrutura, clique em Next ou continue para Apache MXNet (p. 27).

Apache MXNet

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Para Python 3 com CUDA 9:

```
$ source activate mxnet_p36
```

Para Python 2 com CUDA 9:

```
$ source activate mxnet_p27
```

Inicie o terminal iPython.

```
(mxnet_p36)$ ipython
```

Execute um programa rápido do MXNet. Crie uma matriz 5x5, uma instância do `NDArray` com elementos inicializados como 0. Imprima a matriz.

```
import mxnet as mx
mx.ndarray.zeros((5,5)).asnumpy()
```

Verifique o resultado.

```
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
```

```
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]], dtype=float32)
```

Mais tutoriais

Você pode localizar mais tutoriais na pasta de tutoriais Deep Learning AMI com Conda no diretório inicial da DLAMI. Para obter mais tutoriais e exemplos, consulte os documentos oficiais do Python referentes à estrutura, API Python para MXNet e o site Apache MXNet.

Caffe2

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Há apenas o Python 2 com a opção CUDA 9:

```
$ source activate caffe2_p27
```

Inicie o terminal iPython.

```
(caffe2_p27)$ ipython
```

Execute um programa rápido do Caffe2.

```
from caffe2.python import workspace, model_helper
import numpy as np
# Create random tensor of three dimensions
x = np.random.rand(4, 3, 2)
print(x)
print(x.shape)
workspace.FeedBlob("my_x", x)
x2 = workspace.FetchBlob("my_x")
print(x2)
```

Você deve ver as matrizes aleatórias iniciais do numpy impressas e as carregadas em um blob do Caffe2. Observe que, após o carregamento, elas são iguais.

Mais tutoriais

Para obter mais tutoriais e exemplos, consulte os documentos oficiais do Python referentes à estrutura, API Python para Caffe2 e o site Caffe2.

CNTK

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Para Python 3 com CUDA 9:

```
$ source activate cntk_p36
```

Para Python 2 com CUDA 9:

```
$ source activate cntk_p27
```

Inicie o terminal iPython.

```
(caffe2_p27)$ ipython
```

Execute um programa rápido do Caffe2.

```
import cntk as C
C.__version__
c = C.constant(3, shape=(2,3))
c.asarray()
```

Você deve ver a versão do CNTK e, em seguida, a saída de uma matriz 2x3 de 3s.

Se você tiver uma instância de GPU, poderá testá-la com o seguinte código de exemplo. Um resultado de `True` é o que você deve esperar se o CNTK puder acessar a GPU.

```
from cntk.device import try_set_default_device, gpu
try_set_default_device(gpu(0))
```

Mais tutoriais

Para obter mais tutoriais e exemplos, consulte os documentos oficiais do Python referentes à estrutura, API Python para CNTK e o site CNTK.

Keras

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Para o Keras 1 com um back-end do MXNet no Python 3 com o CUDA 9:

```
$ source activate mxnet_p36
```

Para o Keras 1 com um back-end do MXNet no Python 2 com o CUDA 9:

```
$ source activate mxnet_p27
```

Para o Keras 2 com um back-end do TensorFlow no Python 3 com o CUDA 8:

```
$ source activate tensorflow_p36
```

Para o Keras 2 com um back-end do TensorFlow no Python 2 com o CUDA 8:

```
$ source activate tensorflow_p27
```

Mais tutoriais

Você pode localizar mais tutoriais na pasta Deep Learning AMI com Conda tutorials/tensorflow no diretório inicial da DLAMI. Para obter mais tutoriais e exemplos, consulte o site do Keras.

PyTorch

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Para Python 3 com CUDA 8:

```
$ source activate pytorch_p36
```

Para Python 2 com CUDA 8:

```
$ source activate pytorch_p27
```

Inicie o terminal iPython.

```
(pytorch_p36)$ ipython
```

Execute um programa rápido do PyTorch.

```
import torch
x = torch.rand(5, 3)
print(x)
print(x.size())
y = torch.rand(5, 3)
print(torch.add(x, y))
```

Você deve ver a matriz aleatória inicial impressa, seu tamanho e a adição de outra matriz aleatória.

Mais tutoriais

Você pode localizar mais tutoriais na pasta de tutoriais Deep Learning AMI com Conda no diretório inicial da DLAMI. Para obter mais tutoriais e exemplos, consulte os documentos oficiais da estrutura, Documentação do PyTorch e o site do PyTorch.

TensorFlow

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Para TensorFlow + Keras 2 no Python 3 com o CUDA 8:

```
$ source activate tensorflow_p36
```

Para TensorFlow + Keras 2 no Python 2 com o CUDA 8:

```
$ source activate tensorflow_p27
```

Inicie o terminal iPython.

```
(tensorflow_p36)$ ipython
```

Execute um programa rápido do TensorFlow.

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Você deve ver "Hello, Tensorflow!"

Mais tutoriais

Você pode localizar mais tutoriais na pasta de tutoriais Deep Learning AMI com Conda no diretório inicial da DLAMI. Para obter mais tutoriais e exemplos, consulte os documentos oficiais da estrutura API do Python TensorFlow e o site do TensorFlow.

Theano

Para ativar a estrutura, siga estas instruções sobre a sua Deep Learning AMI com Conda.

Para Theano + Keras no Python 3 com o CUDA 9:

```
$ source activate theano_p36
```

Para Theano + Keras no Python 2 com o CUDA 9:

```
$ source activate theano_p27
```

Inicie o terminal iPython.

```
(theano_p36)$ ipython
```

Execute um programa rápido do Theano.

```
import numpy
import theano
import theano.tensor as T
from theano import pp
x = T.dscalar('x')
y = x ** 2
gy = T.grad(y, x)
pp(gy)
```

Você deve ver um Theano calculando um gradiente simbólico.

Mais tutoriais

Para obter mais tutoriais e exemplos, consulte os documentos oficiais da estrutura, API do Python Theano e o site do Theano.

Servidor modelo MXNet

O MXNet Model Server (MMS) é uma ferramenta flexível e fácil de usar para fornecer modelos de deep learning exportados do MXNet.

Fornecer um modelo de deep learning em dois minutos

O MMS vem pré-instalado com a Deep Learning AMI com Conda. Você pode colocar o modelo de fornecimento do MMS para funcionar muito rapidamente com apenas alguns comandos.

Neste tutorial, vamos ignorar a maioria dos recursos, mas certifique-se de ler a documentação MMS quando quiser saber mais. Veja a seguir um exemplo fácil para fornecer um modelo de classificação de imagem. Execute o servidor, que ouvirá solicitações de previsão. Em seguida, faça upload de uma imagem, e o servidor retornará uma previsão das 5 melhores das 1.000 classes que foram treinadas.

Conecte-se à Deep Learning AMI com Conda e ative o ambiente de MXNet.

```
$ source activate mxnet_p36
```

Em seguida, execute o `mxnet-model-server` com esta chamada que faz download de um modelo e o fornece.

```
$ mxnet-model-server \
--models squeezenet=https://s3.amazonaws.com/model-server/models/squeezenet_v1.1/
squeezenet_v1.1.model
```

Com o comando executado acima, você tem o MMS em execução no host, ouvindo as solicitações de inferência.

Para testá-lo, você precisa abrir uma nova janela no terminal.

Conecte-se à DLAMI, use os comandos abaixo para fazer download de uma imagem de gatinho e a envie ao endpoint de previsão MMS.

```
$ curl -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
$ curl -X POST http://127.0.0.1:8080/squeezenet/predict -F "data=@kitten.jpg"
```

O endpoint de previsão retornará uma resposta de previsão em JSON. O resultado será semelhante ao seguinte:

```
        {
        "prediction": [
        [
        {
        "class": "n02124075 Egyptian cat",
```

```
        "probability": 0.940
        },
        {
        "class": "n02127052 lynx, catamount",
        "probability": 0.055
        },
        {
        "class": "n02123045 tabby, tabby cat",
        "probability": 0.002
        },
        {
        "class": "n02123159 tiger cat",
        "probability": 0.0003
        },
        {
        "class": "n02123394 Persian cat",
        "probability": 0.0002        }
        ]
        ]
        }
```

Exemplo de detecção multiclasse

Na DLAMI, você encontrará um exemplo de aplicativo usando o MMS com um modelo de Single Shot Detection (SSD). Do terminal da DLAMI, localize a pasta `~/tutorials/MXNet-Model-Server/ssd`. As instruções sobre como executar o exemplo estão no arquivo `README.md`, ou as instruções podem ser visualizadas na versão mais recente do exemplo do repositório do MMS.

Outros recursos e exemplos

Se você quiser obter mais exemplos com informações sobre exportação de modelos, configuração do MMS com o Docker ou aproveitamento dos recursos mais recentes, acesse a página de projeto do MMS.

TensorFlow Serving

O TensorFlow Serving é um sistema de fornecimento flexível de alto desempenho para modelos de aprendizagem de máquina.

Treinar e fornecer um modelo MNIST com o TensorFlow Serving

O `tensorflow-serving-api` é pré-instalado com Deep Learning AMI com Conda. Você encontrará scripts de exemplo para treinar, exportar e fornecer um modelo MNIST em `~/tutorials/TensorFlow/serving`. Neste tutorial, vamos exportar um modelo e, em seguida, fornecê-lo com o aplicativo `tensorflow_model_server`. Finalmente, você pode testar o servidor modelo com um exemplo de script do cliente.

Conecte-se à Deep Learning AMI com Conda e ative o ambiente de TensorFlow Python 2.7. Os scripts de exemplo não são compatíveis com Python 3.x.

```
$ source activate tensorflow_p27
```

Agora, altere os diretórios para a pasta de scripts de exemplo de fornecimento.

```
$ cd ~/tutorials/TensorFlow/serving
```

Execute o script que treina e exporta um modelo MNIST. Como único argumento do script, você precisa fornecer uma pasta local para salvar o modelo. Por enquanto, podemos colocá-lo em `mnist_model`. O script criará a pasta para você.

```
$ python mnist_saved_model.py /tmp/mnist_model
```

Aguarde. Este script pode demorar um pouco antes de fornecer resultados. Ao concluir o treinamento e, por fim, exportar o modelo, você verá o seguinte:

```
        Done training!
        Exporting trained model to mnist_model/1
        Done exporting!
```

A próxima etapa é executar o `tensorflow_model_server` para fornecer o modelo exportado.

```
$ tensorflow_model_server --port=9000 --model_name=mnist --model_base_path=/tmp/mnist_model
```

Um script de cliente é disponibilizado para testar o servidor.

Para testá-lo, você precisa abrir uma nova janela no terminal.

```
$ python mnist_client.py --num_tests=1000 --server=localhost:9000
```

Outros recursos e exemplos

Se você quiser saber mais sobre o TensorFlow Serving, acesse o site do TensorFlow Serving.

TensorBoard

O TensorBoard é usado para visualizar seu gráfico do TensorFlow, plotar métricas quantitativas sobre a execução do seu gráfico e mostrar dados adicionais, como imagens que passam por ele.

Treinar um modelo MNIST e visualizá-lo com o TensorBoard

A ferramenta do `tensorboard` é pré-instalada com a Deep Learning AMI com Conda. Você encontrará um script de exemplo para treinar o modelo MNIST com recursos adicionais de registro em log em `~/tutorials/TensorFlow/board`. Esses logs são usados pelo TensorBoard para criar visualizações. O TensorBoard executa um servidor web para fornecer uma página da web para exibir e interagir com as visualizações.

Conecte-se à Deep Learning AMI com Conda e ative o ambiente de TensorFlow Python 2.7. Agora, altere os diretórios para o exemplo de pasta de scripts do TensorBoard.

```
$ source activate tensorflow_p27
$ cd ~/tutorials/TensorFlow/board
```

Execute o script que treinará o modelo MNIST ao mesmo tempo que executa o registro em log estendido.

```
$ python mnist_with_summaries.py
```

O script é codificado para gravar os logs em `/tmp/tensorflow/mnist`. Esse local é o parâmetro que você precisa repassar ao `tensorboard`:

```
$ tensorboard --logdir=/tmp/tensorflow/mnist
```

Por padrão, ele abrirá o servidor de visualização na porta 6006. Pode ser necessário alterar para a porta 80 ou outra para facilitar o acesso do seu navegador local. Você precisará abrir essa porta no security group do EC2 para a DLAMI. Você também pode usar o encaminhamento de porta. As instruções para alterar suas configurações do security group e encaminhamento de porta são as mesmas seguidas para realizar o Configurar um servidor de notebook Jupyter (p. 14). Observe que, se você deseja executar o servidor Jupyter e um servidor TensorBoard, você precisa escolher portas diferentes para cada um deles.

Para abrir a porta 6006 (ou outra de sua escolha) no firewall da instância:

Abra o painel do EC2 e escolha Security Groups na barra de navegação do EC2 na seção Network & Security. Haverá um ou mais security groups na lista dessa página. Localize o mais recente (a descrição tem uma data e hora), selecione-o, escolha a guia Inbound e, em seguida, clique em Edit. Em seguida, clique em Add Rule. Isso adiciona uma nova linha. Preencha os campos com as seguintes informações:

Type**: Custom TCP Rule**

Protocol: **TCP**

Port Range: **6006**

Source: **Anywhere** (**0.0.0.0/0,::/0**)

Abra a página da web para as visualizações do TensorBoard usando o IP público ou DNS da DLAMI e a porta aberta para o TensorBoard:

http:// ***YourInstancePublicDNS*****:6006**

Outros recursos e exemplos

Se você está interessado em saber mais sobre o TensorBoard, acesse o site do TensorBoard.

Recursos e suporte

Tópicos

Fóruns

- Fórum: Deep Learning AMIs da AWS

Postagens de blogs relacionadas

- Lista atualizada de artigos relacionados às Deep Learning AMIs
- Novas Deep Learning AMIs da AWS para profissionais de aprendizagem de máquina
- Conceitos básicos do deep learning com a Deep Learning AMI da AWS
- Apresentação da nova Deep Learning AMI da AWS para instâncias P3 do Amazon EC2
- Novos treinamentos técnicos disponíveis: introdução à aprendizagem de máquina e ao deep learning na AWS
- Jornada para o Deep Learning com a AWS

Perguntas frequentes

- P: Como controlo os anúncios de produtos relacionados à DLAMI?

  Aqui estão duas sugestões para isso:
  - Marque esta categoria do blog, "AWS Deep Learning AMIs" encontrada aqui: Lista atualizada de artigos relacionados às Deep Learning AMIs.
  - "Assista" ao Fórum: Deep Learning AMIs da AWS
- P: Os drivers NVIDIA e o CUDA estão instalados?

  Sim. Algumas DLAMIs têm versões diferentes. O Deep Learning AMI com Conda (p. 5) tem as versões mais recentes de todas as DLAMIs. Isso é abordado em mais detalhes em Instalações do CUDA e associações da estrutura (p. 7). Você também pode consultar a página de detalhes da AMI específica na loja para confirmar o que está instalado.
- P: O cuDNN está instalado?

  Sim.
- P: Como faço para ver se as GPUs estão detectadas e seu status atual?

  Executar `nvidia-smi`. Isso mostrará uma ou mais GPUs, dependendo do tipo de instância, juntamente com o seu consumo de memória atual.
- P. Há ambientes virtuais configurados para mim?

  Sim, mas somente na Deep Learning AMI com Conda (p. 5).

- P. Qual versão do Python está instalada?

  Cada DLAMI tem Python 2 e 3. A Deep Learning AMI com Conda (p. 5) tem ambientes para as duas versões de cada estrutura. A Deep Learning AMI com código-fonte (p. 6) tem as estruturas de deep learning instaladas em versões específicas do Python, indicadas com "2" ou "3" nos nomes das pastas.
- P: O Keras está instalado?

  Isso depende da AMI. A Deep Learning AMI com Conda (p. 5) tem o Keras disponível como um front-end para cada estrutura. A versão do Keras depende do suporte que a estrutura oferece para ele.
- P: É gratuito?

  Todas as DLAMIs são gratuitas. No entanto, dependendo do tipo de instância que você escolher, a instância poderá não ser gratuita. Consulte Definição de preço para a DLAMI (p. 9) para obter mais informações.
- P. Estou recebendo erros do CUDA ou mensagens relacionadas à GPU em minha estrutura. O que está errado?

  Verifique o tipo de instância que você usou. Para funcionar, ela precisa ter uma GPU para muitos exemplos e tutoriais. Se a execução do `nvidia-smi` não mostrar nenhuma GPU, você precisará criar outra DLAMI usando uma instância com uma ou mais GPUs. Consulte Seleção do tipo de instância para DLAMI (p. 8) para obter mais informações.
- P: Posso usar o Docker?

  O Docker não é instalado, mas você pode instalá-lo e usá-lo. Observe que você desejará usar o nvidia-docker em instâncias de GPU para fazer uso da GPU. Nessa situação, uma Versões do Ubuntu da AWS Deep Learning AMI, (p. 42) é a melhor opção, pois há atualmente algumas incompatibilidades com o nvidia-docker e a Deep Learning AMI (Amazon Linux).
- P: Em quais regiões as DLAMI s do Linux estão disponíveis?

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

- P: Em quais regiões as DLAMI s do Windows estão disponíveis?

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2

Região	Code
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
GovCloud	ec2-us-gov-west-1
Oeste dos EUA (Norte da Califórnia)	ec2-us-west-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
Canadá (Central)	ec2-ca-central-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1
UE (Londres)	ec2-eu-west-2
AS (São Paulo)	ec2-sa-east-1

Apêndice

Tópicos

Opções do AMI

Os tópicos a seguir descrevem as categorias das AWS Deep Learning AMIs.

Tópicos

Deep Learning AMI com Conda

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI (Ubuntu)
- Deep Learning AMI (Amazon Linux)

Essas DLAMIs estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1

Região	Code
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Deep Learning AMI com código-fonte

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI com código-fonte (CUDA 9, Ubuntu)
- Deep Learning AMI com código-fonte (CUDA 9, Amazon Linux)
- Deep Learning AMI com código-fonte (CUDA 8, Ubuntu)
- Deep Learning AMI com código-fonte (CUDA 8, Amazon Linux)

Essas DLAMIs estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Deep Learning Base AMI

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning Base AMI (Ubuntu)
- Deep Learning Base AMI (Amazon Linux)

Essas DLAMIs estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2

Região	Code
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Deep Learning AMI com CUDA 9

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI (Ubuntu)
- Deep Learning AMI (Amazon Linux)
- Deep Learning Base AMI (Ubuntu)
- Deep Learning Base AMI (Amazon Linux)
- Deep Learning AMI com código-fonte (CUDA 9, Ubuntu)
- Deep Learning AMI com código-fonte (CUDA 9, Amazon Linux)
- Deep Learning AMI (Windows 2012 R2)
- Deep Learning AMI (Windows 2016)

> **Note**
>
> A Deep Learning AMI com Conda tem o CUDA 8 e o CUDA 9. A estruturas usarão o CUDA mais recente para o qual oferecem suporte.
> A Deep Learning Base AMI também tem o CUDA 8 e o CUDA 9. Para alternar entre os dois, siga as instruções em Usar a Deep Learning Base AMI (p. 25).

Deep Learning AMI com CUDA 8

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI (Ubuntu)
- Deep Learning AMI (Amazon Linux)
- Deep Learning Base AMI (Ubuntu)
- Deep Learning Base AMI (Amazon Linux)
- Deep Learning AMI com código-fonte (CUDA 8, Ubuntu)
- Deep Learning AMI com código-fonte (CUDA 8, Amazon Linux)
- Deep Learning AMI (Windows 2012 R2)

- Deep Learning AMI (Windows 2016)

> **Note**
>
> A Deep Learning AMI com Conda tem o CUDA 8 e o CUDA 9. A estruturas usarão o CUDA mais recente para o qual oferecem suporte.
> A Deep Learning Base AMI também tem o CUDA 8 e o CUDA 9. Para alternar entre os dois, siga as instruções em Usar a Deep Learning Base AMI (p. 25).

Versões do Ubuntu da AWS Deep Learning AMI,

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI (Ubuntu)
- Deep Learning Base AMI (Ubuntu)
- Deep Learning AMI com código-fonte (CUDA 9, Ubuntu)
- Deep Learning AMI com código-fonte (CUDA 8, Ubuntu)

Essas DLAMIs estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Versões da AWS Deep Learning AMI, Amazon Linux

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI (Amazon Linux)
- Deep Learning Base AMI (Amazon Linux)
- Deep Learning AMI com código-fonte (CUDA 9, Amazon Linux)
- Deep Learning AMI com código-fonte (CUDA 8, Amazon Linux)

Essas DLAMIs estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Versões da AWS Deep Learning AMI, do Windows

Use o guia Iniciar e configurar uma DLAMI (p. 11) para continuar com uma dessas DLAMI.

- Deep Learning AMI (Windows 2016)
- Deep Learning AMI (Windows 2012 R2)

As DLAMIs do Windows estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
GovCloud	ec2-us-gov-west-1
Oeste dos EUA (Norte da Califórnia)	ec2-us-west-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
Canadá (Central)	ec2-ca-central-1

Região	Code
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1
UE (Londres)	ec2-eu-west-2
AS (São Paulo)	ec2-sa-east-1

Notas de release da DLAMI:

- Notas de release da AWS Deep Learning AMI atuais
- Arquivo de release da AWS Deep Learning AMI (p. 44)

Arquivo de release da AWS Deep Learning AMI

Tópicos

Arquivo de release base da AWS Deep Learning AMI

Tópicos

Detalhes das notas de release da Deep Learning Base AMI (Amazon Linux) versão 2.0

AWS Deep Learning AMI

A Deep Learning Base AMI é pré-compilada com o CUDA 8 e 9 e pronta para a sua configuração de deep learning personalizada. A Deep Learning Base AMI usa a plataforma Anaconda com Python2 e Python3.

Destaques da versão

1. Usou o Amazon Linux 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.1
5. CuBLAS 8 e 9
6. glibc 2.18
7. OpenCV 3.2.0

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI.

Suporte de tipos de instância de CPU

A AMI oferece suporte aos Tipos de instância de CPU.

Drivers de GPU instalados

- Nvidia 384.81
- CUDA 9.0
- CuDNN 7

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Deep Learning AMI (Amazon Linux)

Disponível nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Nenhum problema conhecido.

Detalhes das notas de release da Deep Learning Base AMI (Ubuntu) versão 2.0

AWS Deep Learning AMI

A Deep Learning Base AMI é pré-compilada com o CUDA 8 e 9 e pronta para a sua configuração de deep learning personalizada. A Deep Learning Base AMI usa a plataforma Anaconda com Python2 e Python3.

Destaques da versão

1. Usou Ubuntu 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.1
5. CuBLAS 8 e 9
6. glibc 2.18
7. OpenCV 3.2.0

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI.

Suporte de tipos de instância de CPU

A AMI oferece suporte aos Tipos de instância de CPU.

Drivers de GPU instalados

- Nvidia 384.81
- CUDA 9
- CuDNN 7

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Deep Learning AMI (Ubuntu)

Disponível nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2

Região	Code
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Nenhum problema conhecido.

Detalhes das notas de release da Deep Learning Base AMI (Amazon Linux) versão 1.0

AWS Deep Learning AMI

A Deep Learning Base AMI é pré-compilada com o CUDA 8 e 9 e pronta para a sua configuração de deep learning personalizada. A Deep Learning Base AMI usa a plataforma Anaconda com Python2 e Python3.

Destaques da versão

1. Usou o Amazon Linux 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0.5
5. CuBLAS 8 e 9
6. glibc 2.18
7. OpenCV 3.2.0

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI.

Suporte de tipos de instância de CPU

A AMI oferece suporte aos Tipos de instância de CPU.

Drivers de GPU instalados

- Nvidia 384.81
- CUDA 9.0
- CuDNN 7

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Deep Learning AMI (Amazon Linux)

Essa AMI está disponível nas seguintes regiões:

- Leste dos EUA (Ohio): ec2-us-east-2
- Leste dos EUA (Norte da Virgínia): ec2-us-east-1
- Oeste dos EUA (Norte da Califórnia): ec2-us-west-1
- Oeste dos EUA (Oregon): ec2-us-west-2
- Ásia-Pacífico (Seul): ec2-ap-northeast-2
- Ásia-Pacífico (Cingapura): ec2-ap-southeast-1
- Ásia-Pacífico (Tóquio): ec2-ap-northeast-1
- UE (Irlanda): ec2-eu-west-1

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos da Deep Learning Base AMI (Amazon Linux)

- Problema: as versões de pip e Python não são compatíveis com a Deep Learning AMI (DLAMI) do Amazon Linux, especificamente o pip, que deveria instalar as associações do Python 2, mas instala as associações do Python 3.

  Esse é um problema conhecido documentado no site do pip. Este problema será corrigido em uma versão futura.

  Solução: use o comando relevante a seguir para instalar o pacote da versão adequada do Python:

  ```
  python2.7 -m pip install some-python-package
  ```

  ```
  python3.4 -m pip install some-python-package
  ```
- Problema: o MOTD tem um link incorreto para essas notas de release.

  Solução: se você chegou até aqui, já saberá.

  Este problema será corrigido em uma versão futura.

Detalhes das notas de release da Deep Learning Base AMI (Ubuntu) versão 1.0

AWS Deep Learning AMI

A Deep Learning Base AMI é pré-compilada com o CUDA 8 e 9 e pronta para a sua configuração de deep learning personalizada. A Deep Learning Base AMI usa a plataforma Anaconda com Python2 e Python3.

Destaques da versão

1. Usou Ubuntu 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0.5
5. CuBLAS 8 e 9

6. glibc 2.18
7. OpenCV 3.2.0

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI.

Suporte de tipos de instância de CPU

A AMI oferece suporte aos Tipos de instância de CPU.

Drivers de GPU instalados

- Nvidia 384.81
- CUDA 9
- CuDNN 7

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Deep Learning AMI (Ubuntu)

Essa AMI está disponível nas seguintes regiões:

- Leste dos EUA (Ohio): ec2-us-east-2
- Leste dos EUA (Norte da Virgínia): ec2-us-east-1
- Oeste dos EUA (Norte da Califórnia): ec2-us-west-1
- Oeste dos EUA (Oregon): ec2-us-west-2
- Ásia-Pacífico (Seul): ec2-ap-northeast-2
- Ásia-Pacífico (Cingapura): ec2-ap-southeast-1
- Ásia-Pacífico (Tóquio): ec2-ap-northeast-1
- UE (Irlanda): ec2-eu-west-1

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos da Deep Learning Base AMI (Ubuntu)

- Problema: o MOTD tem um link incorreto para essas notas de release.

  Solução: se você chegou até aqui, já saberá.

  Este problema será corrigido em uma versão futura.

Arquivo de release do Conda da AWS Deep Learning AMI

Tópicos

Detalhes das notas de release da Deep Learning AMI (Amazon Linux) versão 2.0

AWS Deep Learning AMI

A AWS Deep Learning AMI é pré-compilada com o CUDA 8 e 9 e várias estruturas de deep learning. A DLAMI usa a plataforma Anaconda com Python2 e Python3 para alternar facilmente entre as estruturas.

Destaques da versão

1. Usou Deep Learning Base AMI (Amazon Linux) como a AMI de base
2. CUDA 9
3. CuDNN 7.0.3
4. NCCL 2.1
5. CuBLAS 8 e 9
6. glibc 2.18
7. OpenCV 3.2.0
8. Melhor desempenho de ambientes Conda de carregamento

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v1.0
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate caffe2_p27`
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: v2.2
  - Justificativa: última versão
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate cntk_p27`

  - Para Anaconda Python 3+ - `source activate cntk_p36`
- Keras: Keras - Biblioteca de deep learning para Python)

  Integração do Tensorflow com v2.0.9
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`

  Integração do MXNet integração com v1.2.2
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- PyTorch: PyTorch é um pacote python que fornece dois recursos de alto nível: computação Tensor (como numpy) com aceleração forte de GPU e redes neurais profundas criadas em um sistema autograd baseado em fita
  - Ramificação/tag usada: v0.3.0
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate pytorch_p27`
    - Para Anaconda Python 3+ - `source activate pytorch_p36`
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: v1.4
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: v0.9
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate theano_p27`
    - Para Anaconda Python 3+ - `source activate theano_p36`

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI para todas essas estruturas de deep learning instaladas, exceto o Caffe2:

1. Apache MXNet
2. Caffe2 (apenas Python 2.7)
3. CNTK
4. Keras
5. PyTorch
6. Tensorflow

7. Theano

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas.

Drivers de GPU instalados

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Regiões da Deep Learning AMI (Amazon Linux)

Disponível nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Referências

- Apache MXNet
- Caffe2

- CNTK
- Keras
- PyTorch
- TensorFlow
- Theano

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Problema: os testes do PyTorch não funcionam. ~ /src/bin/testPyTorch – instala o ambiente de teste que não é compatível com o pytorch 0.3.0

  Solução: N/D
- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8057`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

Detalhes das notas de release da Deep Learning AMI (Ubuntu) versão 2.0

AWS Deep Learning AMI

A AWS Deep Learning AMI é pré-compilada com o CUDA 8 e 9 e várias estruturas de deep learning. A DLAMI usa a plataforma Anaconda com Python2 e Python3 para alternar facilmente entre as estruturas.

Destaques da versão

1. Usou Deep Learning Base AMI (Ubuntu) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.1
5. CuBLAS 8 e 9
6. glibc 2.18
7. OpenCV 3.2.0
8. Melhor desempenho de ambientes Conda de carregamento

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v1.0
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate caffe2_p27`
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: v2.2
  - Justificativa: última versão
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate cntk_p27`
    - Para Anaconda Python 3+ - `source activate cntk_p36`
- Keras: Keras - Biblioteca de deep learning para Python)

  Integração do Tensorflow com v2.0.9
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`

  Integração do MXNet integração com v1.2.2
  - Justificativa: versão estável

  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- PyTorch: PyTorch é um pacote python que fornece dois recursos de alto nível: computação Tensor (como numpy) com aceleração forte de GPU e redes neurais profundas criadas em um sistema autograd baseado em fita
  - Ramificação/tag usada: v0.3.0
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate pytorch_p27`
    - Para Anaconda Python 3+ - `source activate pytorch_p36`
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: v1.4
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: v0.9
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate theano_p27`
    - Para Anaconda Python 3+ - `source activate theano_p36`

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI para todas essas estruturas de deep learning instaladas, exceto o Caffe2:

1. Apache MXNet
2. Caffe2 (apenas Python 2.7)
3. CNTK
4. Keras
5. PyTorch
6. Tensorflow
7. Theano

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN.

Drivers de GPU instalados

- CuDNN 7
- NVIDIA 384.81

- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ubuntu/src/bin.

Regiões da Deep Learning AMI (Ubuntu)

Disponível nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Referências

- Apache MXNet
- Caffe2
- CNTK
- Keras
- PyTorch
- TensorFlow
- Theano

Ambientes de teste

- Baseado em p2.16xlarge.

- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Problema: os testes do PyTorch não funcionam. ~ /src/bin/testPyTorch – instala o ambiente de teste que não é compatível com o pytorch 0.3.0

  Solução: N/D
- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8057`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

Detalhes das notas de release da Deep Learning AMI (Amazon Linux) versão 1.0

AWS Deep Learning AMI

A AWS Deep Learning AMI é pré-compilada com o CUDA 8 e 9 e várias estruturas de deep learning. A DLAMI usa a plataforma Anaconda com Python2 e Python3 para alternar facilmente entre as estruturas.

Destaques da versão

1. Usou o Amazon Linux 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7.0.3
4. NCCL 2.0.5
5. CuBLAS 8 e 9
6. glibc 2.18

7. OpenCV 3.2.0

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v0.12.0
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate caffe2_p27`
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: v2.2
  - Justificativa: última versão
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate cntk_p27`
    - Para Anaconda Python 3+ - `source activate cntk_p36`
- Keras: Keras - Biblioteca de deep learning para Python)

  Integração do Tensorflow com v2.0.9
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`

  Integração do MXNet integração com v1.2.2
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- PyTorch: PyTorch é um pacote python que fornece dois recursos de alto nível: computação Tensor (como numpy) com aceleração forte de GPU e redes neurais profundas criadas em um sistema autograd baseado em fita
  - Ramificação/tag usada: v0.2
  - Justificativa: estável e bem testado
  - Para ativar:

  - Para Anaconda Python 2.7+ - `source activate pytorch_p27`
  - Para Anaconda Python 3+ - `source activate pytorch_p36`
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: v1.4
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: v0.9
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate theano_p27`
    - Para Anaconda Python 3+ - `source activate theano_p36`

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI para todas essas estruturas de deep learning instaladas, exceto o Caffe2:

1. Apache MXNet
2. Caffe2 (apenas Python 2.7)
3. CNTK
4. PyTorch
5. Tensorflow
6. Theano

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas.

Drivers de GPU instalados

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Regiões da Deep Learning AMI (Amazon Linux)

As Deep Learning AMI (Amazon Linux)s estão disponíveis nas seguintes regiões:

- Leste dos EUA (Ohio): ec2-us-east-2
- Leste dos EUA (Norte da Virgínia): ec2-us-east-1
- Oeste dos EUA (Norte da Califórnia): ec2-us-west-1
- Oeste dos EUA (Oregon): ec2-us-west-2
- Ásia-Pacífico (Seul): ec2-ap-northeast-2
- Ásia-Pacífico (Cingapura): ec2-ap-southeast-1
- Ásia-Pacífico (Tóquio): ec2-ap-northeast-1
- UE (Irlanda): ec2-eu-west-1

Referências

- Apache MXNet
- Caffe2
- CNTK
- Keras
- PyTorch
- TensorFlow
- Theano

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos da Deep Learning AMI (Amazon Linux)

- Problema: as versões de pip e Python não são compatíveis com a Deep Learning AMI (DLAMI) do Amazon Linux, especificamente o pip, que deveria instalar as associações do Python 2, mas instala as associações do Python 3.

  Esse é um problema conhecido documentado no site do pip. Este problema será corrigido em uma versão futura.

  Solução: use o comando relevante a seguir para instalar o pacote da versão adequada do Python:

  ```
  python2.7 -m pip install some-python-package
  ```

  ```
  python3.4 -m pip install some-python-package
  ```
- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.

- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8157`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

  Problema: o MOTD tem um link incorreto para essas notas de release.

  Solução: se você chegou até aqui, já saberá.

  Este problema será corrigido em uma versão futura.

Detalhes das notas de release da Deep Learning AMI (Ubuntu) versão 1.0

AWS Deep Learning AMI

A AWS Deep Learning AMI é pré-compilada com o CUDA 8 e 9 e várias estruturas de deep learning. A DLAMI usa a plataforma Anaconda com Python2 e Python3 para alternar facilmente entre as estruturas.

Destaques da versão

1. Usou Ubuntu 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0.5
5. CuBLAS 8 e 9
6. glibc 2.18
7. OpenCV 3.2.0

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v0.12.0

  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate caffe2_p27`
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: v2.2
  - Justificativa: última versão
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate cntk_p27`
    - Para Anaconda Python 3+ - `source activate cntk_p36`
- Keras: Keras - Biblioteca de deep learning para Python)

  Integração do Tensorflow com v2.0.9
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`

  Integração do MXNet integração com v1.2.2
  - Justificativa: versão estável
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate mxnet_p27`
    - Para Anaconda Python 3+ - `source activate mxnet_p36`
- PyTorch: PyTorch é um pacote python que fornece dois recursos de alto nível: computação Tensor (como numpy) com aceleração forte de GPU e redes neurais profundas criadas em um sistema autograd baseado em fita
  - Ramificação/tag usada: v0.2
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate pytorch_p27`
    - Para Anaconda Python 3+ - `source activate pytorch_p36`
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: v1.4
  - Justificativa: estável e bem testado
  - Para ativar:
    - Para Anaconda Python 2.7+ - `source activate tensorflow_p27`
    - Para Anaconda Python 3+ - `source activate tensorflow_p36`
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.

- Ramificação/tag usada: v0.9
- Justificativa: estável e bem testado
- Para ativar:
  - Para Anaconda Python 2.7+ - `source activate theano_p27`
  - Para Anaconda Python 3+ - `source activate theano_p36`

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI para todas essas estruturas de deep learning instaladas, exceto o Caffe2:

1. Apache MXNet
2. Caffe2 (apenas Python 2.7)
3. CNTK
4. PyTorch
5. Tensorflow
6. Theano

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN.

Drivers de GPU instalados

- CuDNN 7
- NVIDIA 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Guia do desenvolvedor do AWS Deep Learning AMI

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ubuntu/src/bin.

Regiões da Deep Learning AMI (Ubuntu)

As Deep Learning AMI (Ubuntu)s estão disponíveis nas seguintes regiões:

- Leste dos EUA (Ohio): ec2-us-east-2
- Leste dos EUA (Norte da Virgínia): ec2-us-east-1
- Oeste dos EUA (Norte da Califórnia): ec2-us-west-1
- Oeste dos EUA (Oregon): ec2-us-west-2
- Ásia-Pacífico (Seul): ec2-ap-northeast-2

- Ásia-Pacífico (Cingapura): ec2-ap-southeast-1
- Ásia-Pacífico (Tóquio): ec2-ap-northeast-1
- UE (Irlanda): ec2-eu-west-1

Referências

- Apache MXNet
- Caffe2
- CNTK
- Keras
- PyTorch
- TensorFlow
- Theano

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos da Deep Learning AMI (Ubuntu)

- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8157`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

  Problema: o MOTD tem um link incorreto para essas notas de release.

Solução: se você chegou até aqui, já saberá.

Este problema será corrigido em uma versão futura.

Arquivo de release de origem da AWS Deep Learning AMI

Tópicos

Deep Learning AMI com código-fonte (CUDA 9, Ubuntu) Versão: 2.0

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com CUDA9 e MXNet e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou Deep Learning Base AMI (Ubuntu) como a AMI de base
2. Atualizou TensorFlow mestre com suporte para com CUDA9/Volta
3. MXNet atualizado para v1.0
4. Atualizou PyTorch para v0.3.0
5. Adicionou o Keras 2.0.9 com suporte a TensorFlow como back-end padrão

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v1.0
  - Justificativa: estável e bem testado
  - Diretórios de origem:
    - /home/ubuntu/src/mxnet
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Diretórios de origem:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Keras: Keras - Biblioteca de deep learning para Python)

A integração do Tensorflow com o Tensorflow v2.0.9. é o back-end padrão.

  - Justificativa: versão estável
  - Diretórios de origem:
    - /home/ec2-user/src/keras
- PyTorch: PyTorch é um pacote python que fornece dois recursos de alto nível: computação Tensor (como numpy) com aceleração forte de GPU e redes neurais profundas criadas em um sistema autograd baseado em fita.
  - Ramificação/tag usada: v0.3.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/pytorch
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag Master
  - Justificativa: estável e bem testado
  - Diretórios de origem:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Apache MXNet
2. Caffe2
3. Keras
4. PyTorch
5. Tensorflow

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN.

Drivers de GPU instalados

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ubuntu/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Disponibilidade da região da AMI

Disponível nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Referências

- Apache MXNet
- Caffe2
- CNTK
- Keras
- PyTorch
- TensorFlow
- Theano

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Problema: NCCL não é totalmente compatível Tentativas de usar o NCCL com todas as instâncias, mas P3 levará a um travamento.

  Solução: não use o NCCL em instâncias que não sejam a P3.
- Problema: os testes do PyTorch não funcionam. ~ /src/bin/testPyTorch – instala o ambiente de teste que não é compatível com o pytorch 0.3.0

  Solução: N/D
- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8057`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

Versão da Deep Learning AMI Ubuntu: 2.4_Oct2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou Ubuntu 16.04 (ami-d15a75c7) como a AMI de base

2. Compatibilidade com o CUDA 8
3. Atualizações da estrutura para Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.11.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag v1.0
  - Justificativa: compatível com o cuda8.0 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.0
  - Justificativa: estável e bem testado
  - Observação: esta é uma versão experimental e pode haver alguns problemas. Disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe2
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.3.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow

    - Para Python3+- /home/ubuntu/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
    - Ramificação/tag usada: ramificação master
    - Justificativa: nenhum outro ramificação estável ou tag disponível
    - Source_Directories:
        - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
    - Ramificação/tag usada: tag v2.0
    - Justificativa: versão estável
    - Source_Directories:
        - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
    - Ramificação/tag usada: tag v2.0.8
    - Justificativa: versão estável
    - Source_Directories:
        - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 375.66
- CUDA 8.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ubuntu/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2

Região	Code
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK
- Os ambientes do conda keras1.2_p3 e keras1.2_p2 são fornecidos com a versão de apenas CPU do MXNet.

  Para usar Keras com um back-end de MXNet para treinamento em GPUs, você pode resolver esse problema executando o seguinte:

  ```
  pip install mxnet-cu80
  ```

  no ambiente do conda.

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

  ```
  import mxnet as mx
  import tensorflow as tf
  ```

  no mesmo processo do Python pode causar um problema.

Arquivo de release do Ubuntu da DLAMI

Tópicos

Versão da Deep Learning CUDA 9 AMI Ubuntu: 1.0

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com CUDA9 e MXNet e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou Ubuntu 16.04 (ami-d15a75c7) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0
5. MXNet com suporte do CUDA9

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.12.0 Release Candidate
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2

- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag Master
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. MXNet
2. Caffe2
3. Tensorflow

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN.

Drivers de GPU instalados

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ubuntu/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.

- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8157`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

Versão da Deep Learning AMI Ubuntu: 2.3_Sep2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou Ubuntu 16.04 (ami-d15a75c7) como a AMI de base
2. Compatibilidade com o CUDA 8
3. Atualizações da estrutura para Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.11.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag v1.0
  - Justificativa: compatível com o cuda8.0 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.0
  - Justificativa: estável e bem testado
  - Observação: esta é uma versão experimental e pode haver alguns problemas. Disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe2
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado

  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.3.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: mestre (com suporte do MXNet)
  - Justificativa: versão estável (1.2.2)
  - Source_Directories:
    - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 375.66
- CUDA 8.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ubuntu/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)

- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Ubuntu: 2.2_August2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou Ubuntu 16.04 (ami-d15a75c7) como a AMI de base
2. Compatibilidade com o CUDA 8

3. Atualizações da estrutura para Tensorflow(v1.2.0), Caffe2(v0.7.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.10.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag v1.0
  - Justificativa: compatível com o cuda8.0 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.7.0
  - Justificativa: pré-lançamento
  - Observação: esta é uma versão experimental e pode haver alguns problemas. Disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe2
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.2.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3

    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0
  - Justificativa: última versão
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 375.66
- CUDA 8.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Ubuntu: 1.5_June2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. MXNet compilado com suporte do S3(USE_S3=1).
2. Usou a AMI de base mais recente para Ubuntu 14.04 (ami-b1143ba7).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.10.0
  - Justificativa: estável e bem testado

  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.7.0
  - Justificativa: pré-lançamento
  - Observação: esta é uma versão experimental e pode haver alguns problemas. Disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe2
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.1.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0.rc1

  - Justificativa: última versão
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.

- Também testado em g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK
- Há um problema conhecido com o operador Dropout ao ser executado em tipos de instância de GPU. Consulte Github Issue para obter os detalhes e a solução.

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Ubuntu: 1.4_June2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.10.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu

- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.7.0
  - Justificativa: pré-lançamento
  - Observação: esta é uma versão experimental e pode haver alguns problemas. Disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe2
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.1.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0.rc1
  - Justificativa: última versão
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe

2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.

- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK
- Há um problema conhecido com o operador Dropout ao ser executado em tipos de instância de GPU. Consulte Github Issue para obter os detalhes e a solução.

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
```

```
        import tensorflow as tf
```

no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Ubuntu: 1.3_Apr2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.9.3
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.6.0
  - Justificativa: pré-lançamento
  - Observação: esta é uma versão experimental e pode haver alguns problemas. Disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe2
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/Theano

- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.0.1
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0.rc1
  - Justificativa: última versão
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
```

```
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

/home/ubuntu/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)

- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK
- Há um problema conhecido com o operador Dropout ao ser executado em tipos de instância de GPU. Consulte Github Issue para obter os detalhes e a solução.

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Ubuntu: 1.2

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.

  - Ramificação/tag usada: tag v0.9.3
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.0.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0beta12.0
  - Justificativa: última versão
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:

- /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK
- Há um problema conhecido com o operador Dropout ao ser executado em tipos de instância de GPU. Consulte Github Issue para obter os detalhes e a solução.

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Ubuntu: 1.1

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.9.3
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc4
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v0.12.1
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3

    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0beta7.0
  - Justificativa: última versão
  - Source_Directories:
    - /home/ubuntu/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.1
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ubuntu/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 352.99
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Versão da Deep Learning AMI Ubuntu: 1.0

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.9.3
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: ramificação master
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/caffe
    - Para Python3+ - /home/ubuntu/src/caffe3
    - Para Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ubuntu/src/caffe_cpu
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ubuntu/src/theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v0.12.1
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ubuntu/src/tensorflow
    - Para Python3+- /home/ubuntu/src/tensorflow3

    - Para Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
    - Ramificação/tag usada: ramificação master
    - Justificativa: nenhum outro ramificação estável ou tag disponível
    - Source_Directories:
        - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
    - Ramificação/tag usada: tag v2.0beta7.0
    - Justificativa: última versão
    - Source_Directories:
        - /home/ubuntu/src/cntk

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ubuntu/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ubuntu/src/anaconda3/bin
    source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 352.99
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ubuntu/src/bin/testAll : testa todas as estruturas

/home/ubuntu/src/bin/testMXNet: testa MXNet

/home/ubuntu/src/bin/testTheano : testa Theano

/home/ubuntu/src/bin/testTensorFlow: testa TensorFlow

/home/ubuntu/src/bin/testTorch : testa Torch

/home/ubuntu/src/bin/testCNTK : testa CNTK

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.

Ubuntu AMI

Deep Learning AMIs com base em Ubuntu estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.

- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Arquivo de release do Amazon Linux da DLAMI

Tópicos

Deep Learning AMI com código-fonte (CUDA 9, Amazon Linux) Versão: 2.0

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com CUDA9 e MXNet e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou Deep Learning Base AMI (Amazon Linux) como a AMI de base
2. Atualizou TensorFlow mestre com suporte para com CUDA9/Volta
3. MXNet atualizado para v1.0
4. Atualizou PyTorch para v0.3.0
5. Adicionou o Keras 2.0.9 com suporte a TensorFlow como back-end padrão

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v1.0
  - Justificativa: estável e bem testado
  - Diretórios de origem:
    - /home/ec2-user/src/mxnet
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Diretórios de origem:
    - Para Python2.7+- /home/ec2-user/src/caffe2

    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Keras: Keras - Biblioteca de deep learning para Python)

  A integração do Tensorflow com o Tensorflow v2.0.9. é o back-end padrão.
  - Justificativa: versão estável
  - Diretórios de origem:
    - /home/ec2-user/src/keras
- PyTorch: PyTorch é um pacote python que fornece dois recursos de alto nível: computação Tensor (como numpy) com aceleração forte de GPU e redes neurais profundas criadas em um sistema autograd baseado em fita.
  - Ramificação/tag usada: v0.3.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/pytorch
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag Master
  - Justificativa: estável e bem testado
  - Diretórios de origem:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Apache MXNet
2. Caffe2
3. Keras
4. PyTorch
5. Tensorflow

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN.

Drivers de GPU instalados

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Disponibilidade da região da AMI

Disponível nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Referências

- Apache MXNet
- Caffe2
- CNTK
- Keras
- PyTorch

- TensorFlow
- Theano

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Problema: NCCL não é totalmente compatível Tentativas de usar o NCCL com todas as instâncias, mas P3 levará a um travamento.

  Solução: não use o NCCL em instâncias que não sejam a P3.
- Problema: os testes do PyTorch não funcionam. ~ /src/bin/testPyTorch – instala o ambiente de teste que não é compatível com o pytorch 0.3.0

  Solução: N/D
- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8057`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

Versão da Deep Learning AMI Amazon Linux: 3.3_Oct2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. A AMI do Linux agora é compilada com a Amazon Linux AMI base 2017.09
2. Compatibilidade com o CUDA 8
3. Atualizações da estrutura para Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.11.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag 1.0
  - Justificativa: compatível com o cuda8.0 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.0
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.3.0
  - Justificativa: estável e bem testado

  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0
  - Justificativa: última versão
  - Source_Directories:
    - /home/ec2-user/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag v2.0.8
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ec2-user/src/keras

Suporte do Python 2.7 e do Python 3.4

O Python 2.7 e o Python 3.4 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1

- NVIDIA 375.66
- CUDA 8.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

Região	Code
Leste dos EUA (Ohio)	ec2-us-east-2
Leste dos EUA (Norte da Virgínia)	ec2-us-east-1
Oeste dos EUA (Oregon)	ec2-us-west-2
Pequim (China)	cn-north-1

Região	Code
Ásia Pacífico (Mumbai)	ec2-ap-south-1
Ásia-Pacífico (Seul)	ec2-ap-northeast-2
Ásia-Pacífico (Cingapura)	ec2-ap-southeast-1
Ásia-Pacífico (Sydney)	ec2-ap-southeast-2
Ásia-Pacífico (Tóquio)	ec2-ap-northeast-1
UE (Frankfurt)	ec2-eu-central-1
UE (Irlanda)	ec2-eu-west-1

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo: sudo ./testCNTK
- Os ambientes do conda keras1.2_p3 e keras1.2_p2 são fornecidos com a versão de apenas CPU do MXNet.

  Para usar Keras com um back-end de MXNet para treinamento em GPUs, você pode resolver esse problema executando o seguinte:

  ```
  pip install mxnet-cu80
  ```

  no ambiente do conda.

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

  ```
  import mxnet as mx
  ```

```
import tensorflow as tf
```

no mesmo processo do Python pode causar um problema.

Versão da Deep Learning CUDA 9 AMI Amazon Linux: 1.0

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com CUDA9 e MXNet e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Usou o Amazon Linux 2017.09 (ami-8c1be5f6) como a AMI de base
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0
5. Compatibilidade com o CUDA 9
6. MXNet com suporte do CUDA9

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.12.0 Release Candidate
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.1
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag Master
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. MXNet
2. Caffe2
3. Tensorflow

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN.

Drivers de GPU instalados

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)

- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Problemas conhecidos

- Problema: as versões de pip e Python não são compatíveis com a Deep Learning AMI (DLAMI) do Amazon Linux, especificamente o pip, que deveria instalar as associações do Python 2, mas instala as associações do Python 3.

  Esse é um problema conhecido documentado no site do pip. Este problema será corrigido em uma versão futura.

  Solução: use o comando relevante a seguir para instalar o pacote da versão adequada do Python:

  ```
  python2.7 -m pip install some-python-package
  ```

  ```
  python3.4 -m pip install some-python-package
  ```
- Problema: os tutoriais fornecidos pela estrutura ou por terceiros podem ter dependências do Python não instaladas na DLAMI.

  Solução: você precisará instalá-las no ambiente ativado com conda ou pip.
- Problema: erro de módulo não encontrado ao executar exemplos do Caffe2.

  Solução: as dependências opcionais do Caffe2 são necessárias para alguns tutoriais
- Problema: os recursos de download do modelo do Caffe2 resultam em 404. Os modelos alteraram os locais desde a versão v0.8.1. Atualize models/download.py para usar a atualização do master.
- Problema: matplotlib pode renderizar apenas png.

  Solução: instale o Pillow e reinicie o kernel.
- Problema: a alteração do código-fonte do Caffe2 parece não funcionar.

  Solução: altere o PYTHONPATH para usar o local da instalação `/usr/local/caffe2` em vez da pasta de compilação.
- Problema: erros de net_drawer do Caffe2.

  Solução: use o patch de registrador encontrado nesta confirmação.
- Problema: o exemplo do Caffe2 mostra um erro relativo a LMDB (não é possível abrir o DB etc.)

  Solução: isso exigirá uma compilação do código-fonte depois da instalação do sistema LMDB, como: `sudo apt-get install liblmdb-dev`
- Problema: desconexões do SSH ao usar o servidor Jupyter travam a porta local. Ao tentar criar um túnel para o servidor, você vê `channel_setup_fwd_listener_tcpip: cannot listen to port: 8157`.

  Solução: use `lsof -ti:8057 | xargs kill -9` em que 8057 é a porta local usada. Em seguida, tente criar o túnel para seu servidor Jupyter novamente.

Versão da Deep Learning AMI Amazon Linux: 3.1_Sep2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. Compatibilidade com o CUDA 8
2. Atualizações da estrutura para Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.11.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag 1.0
  - Justificativa: compatível com o cuda8.0 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.8.0
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano

- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.3.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0
  - Justificativa: última versão
  - Source_Directories:
    - /home/ec2-user/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: master
  - Justificativa: versão estável (1.2.2 com suporte do MXNet)
  - Source_Directories:
    - /home/ec2-user/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 375.66
- CUDA 8.0

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância P2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)

- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo: sudo ./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Amazon Linux: 2.3_June2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Destaques da versão

1. MXNet compilado com suporte do S3(USE_S3=1).
2. Correções de segurança aplicadas para problema de segurança de Stakclash. (https://alas.aws.amazon.com/ALAS-2017-845.html)

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos

de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.10.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.7.0
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.1.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:

    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0.rc1
  - Justificativa: última versão
  - Source_Directories:
    - /home/ec2-user/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ec2-user/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Baseado em p2.16xlarge.

- Também testado em g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Amazon Linux: 2.2_June2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.10.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.

  - Ramificação/tag usada: tag v0.7.0
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.9.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.1.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0.rc1
  - Justificativa: última versão
  - Source_Directories:
    - /home/ec2-user/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ec2-user/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano

4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.

- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Amazon Linux: 2.1_Apr2017

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.9.3
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu
- Caffe2: o Caffe2 é uma estrutura entre plataformas criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: tag v0.7.0
  - Justificativa: estável e bem testado
  - Observação: disponível apenas para Python2.7
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe2
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.0.1
  - Justificativa: estável e bem testado
  - Source_Directories:

    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
    - Ramificação/tag usada: ramificação master
    - Justificativa: nenhum outro ramificação estável ou tag disponível
    - Source_Directories:
        - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
    - Ramificação/tag usada: tag v2.0.rc1
    - Justificativa: última versão
    - Source_Directories:
        - /home/ec2-user/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
    - Ramificação/tag usada: tag 1.2.2
    - Justificativa: versão estável
    - Source_Directories:
        - /home/ec2-user/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57

- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

/home/ec2-user/src/bin/testCaffe2: testa Caffe2

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.
- Caffe2: com base neste exemplo no repositório do Caffe2. O limite da precisão da validação é de 90%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

  no mesmo processo do Python pode causar um problema.

Versão da Deep Learning AMI Amazon Linux: 2.0

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: tag v0.9.3
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: tag rc5
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:

    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - Para Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - Para CPU_ONLY: /home/ec2-user/src/caffe_cpu
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: tag v1.0.0
  - Justificativa: estável e bem testado
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/tensorflow
    - Para Python3+- /home/ec2-user/src/tensorflow3
    - Para Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - Para Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação master
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: tag v2.0beta12.0
  - Justificativa: última versão
  - Source_Directories:
    - /home/ec2-user/src/cntk
- Keras: Keras - Biblioteca de deep learning para Python)
  - Ramificação/tag usada: tag 1.2.2
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ec2-user/src/keras

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet

5. CNTK

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
        source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 367.57
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.
- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.

- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Problemas conhecidos

- Necessidade de usar sudo para executar o script testCNTK.

  por exemplo, sudo./testCNTK

Não suportado

- O funcionamento de várias estruturas juntas no mesmo processo Python não foi testado.

  Por exemplo, um fragmento de código, como o seguinte:

```
import mxnet as mx
        import tensorflow as tf
```

Versão da Deep Learning AMI Amazon Linux: 1.5

Deep Learning Amazon Machine Image

As Deep Learning AMIs são pré-compiladas com estruturas populares de deep learning e também contêm a plataforma Anaconda (Python2 e Python3).

Estruturas de deep learning pré-compiladas

- MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande

variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: ramificação master
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: o Caffe é uma estrutura de deep learning criada com expressão, velocidade e modularidade em mente. Desenvolvida pela Berkeley Vision and Learning Center (BVLC) e por colaboradores da comunidade.
  - Ramificação/tag usada: ramificação master
  - Justificativa: compatível com o cuda7.5 e o cudnn 5.1
  - Source_Directories:
    - Para Python2.7+- /home/ec2-user/src/caffe
    - Para Python3+- /home/ec2-user/src/caffe3
    - Para Python3+- /home/ec2-user/src/caffe3
- Theano: o Theano é uma biblioteca do Python que permite definir, otimizar e avaliar expressões matemáticas envolvendo matrizes multidimensionais de forma eficiente.
  - Ramificação/tag usada: tag rel-0.8.2
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: ramificação r0.10
  - Justificativa: estável e bem testado
  - Source_Directories:
    - /home/ec2-user/src/tensorflow
- Torch: Torch é uma estrutura de computação científica com amplo suporte para algoritmos de aprendizagem de máquina que coloca GPUs primeiro. É fácil de usar e eficiente graças a uma linguagem de script fácil e rápida, LuaJIT, e uma implementação de C/CUDA subjacente.
  - Ramificação/tag usada: ramificação rel-0.8.2
  - Justificativa: nenhum outro ramificação estável ou tag disponível
  - Source_Directories:
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - é um toolkit unificado de deep-learning da Microsoft Research.
  - Ramificação/tag usada: v2.0beta2.0
  - Justificativa: versão estável
  - Source_Directories:
    - /home/ec2-user/src/cntk

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.5 são compatíveis com a AMI para as seguintes estruturas de deep learning:

1. Caffe

2. Tensorflow
3. Theano
4. MXNet

Suporte de tipos de instância de CPU

A AMI oferece suporte aos tipos de instâncias de CPU para todas as estruturas. O MXNet é criado com suporte para a biblioteca Intel MKL2017 DNN. Se desejar usar o binário do caffe para a instância de CPU, você deverá usar o binário dentro de /home/ec2-user/src/caffe_cpu/

Suporte do CNTK Python

Você pode executar o CNTK para Python em um ambiente do conda. Para fazer isso:

```
cd /home/ec2-user/src/anaconda3/bin
    source activate cntk-py34
```

Drivers de GPU instalados

- CuDNN 5.1
- NVIDIA 352.99
- CUDA 7.5

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Documentação do EC2 para execução de instância G2

Testar as estruturas

As estruturas de deep learning foram testadas com dados do MNIST. A AMI contém scripts para treinamento e teste com MNIST para cada uma das estruturas. O teste verifica se a precisão da validação está acima de um limite específico. O limite é diferente para cada uma das estruturas.

Os scripts estão disponíveis no diretório /home/ec2-user/src/bin.

Os seguintes scripts testam as diversas estruturas:

/home/ec2-user/src/bin/testAll: testa todas as estruturas

/home/ec2-user/src/bin/testMXNet: testa MXNet

/home/ec2-user/src/bin/testTheano: testa Theano

/home/ec2-user/src/bin/testTensorFlow: testa TensorFlow

/home/ec2-user/src/bin/testTorch: testa Torch

/home/ec2-user/src/bin/testCNTK: testa CNTK

Os testes a seguir foram executados em cada uma das estruturas:

- MXNet: este exemplo no repositório do MXNet. O limite da precisão da validação testada é de 97%.
- Tensorflow: este exemplo no repositório do keras. O limite da precisão da validação testada é de 95%.
- Theano: o mesmo exemplo acima. O limite da precisão da validação é de 95%.

- Torch: este exemplo na árvore do Torch. O limite da precisão da validação é de 93%.
- Caffe: este exemplo no repositório do Caffe. O limite da precisão da validação é de 98%.
- CNTK: este exemplo no repositório do CNTK. O limite da precisão da validação é de 97%.

Amazon Linux AMI

As Deep Learning AMIs com base no Amazon Linux estão disponíveis nas seguintes regiões:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

Referências

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Ambientes de teste

- Criado e testado em g2.2xlarge.
- Também testado em g2.8xlarge, p2.16xlarge, c4.4xlarge.

Arquivo de releases do Windows da AWS Deep Learning AMI

Tópicos

Detalhes das notas de release da Deep Learning AMI (Windows 2016) versão 1.0

AWS Deep Learning AMI

A AWS Deep Learning AMI é pré-compilada com o CUDA 8 e 9 e várias estruturas de deep learning.

Destaques da versão

1. CUDA 8 e 9
2. CuDNN 6 e 7
3. NCCL 2.0.5
4. CuBLAS 8 e 9
5. OpenCV 3.2.0
6. SciPy 0.19.1

7. Conda 5.01

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v0.12.0
  - Compilado com CUDA 8 e cuDNN 6.1
  - Justificativa: estável e bem testado
  - Diretório de origem:

    C:\MXNet
- Caffe: o Caffe é uma estrutura criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: ramificação do Windows, confirmação nº 5854
  - Compilado com CUDA 8 e cuDNN 5
  - Justificativa: estável e bem testado
  - Diretório de origem:

    C:\Caffe
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: v1.4
  - Compilado com CUDA 8 e cuDNN 6.1
  - Justificativa: estável e bem testado
  - Diretório de origem:

    C:\ProgramData\Anaconda3\envs\tensorflow\Lib\site-packages

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI para todas essas estruturas de deep learning instaladas, exceto o Caffe2:

1. Apache MXNet
2. Caffe
3. Tensorflow

Suporte de tipos de instância

Ainda não há suporte para o tipo de instância P3.

A AMI é compatível com todos os outros tipos de instâncias de CPU para todas as estruturas.

Drivers de GPU instalados

- CuDNN 6 e 7
- Nvidia 385.54

- CUDA 8 e 9

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Iniciar uma instância do Windows

Regiões da Deep Learning AMI (Windows 2016)

As AMIs da Windows 2016 estão disponíveis nas seguintes regiões:

- Leste dos EUA (Ohio): ec2-us-east-2 - ami-6cc1ef09
- Leste dos EUA (Norte da Virgínia): ec2-us-east-1 - ami-d50381af
- GovCloud: ec2-us-gov-west-1 - ami-d5018db4
- Oeste dos EUA (Norte da Califórnia): ec2-us-west-1 - ami-0abf876a
- Oeste dos EUA (Oregon): ec2-us-west-2 - ami-d3be62ab
- Ásia-Pacífico (Mumbai): ec2-ap-south-1 - ami-6f1e5000
- Ásia-Pacífico (Seul): ec2-ap-northeast-2 - ami-9375d2fd
- Ásia-Pacífico (Cingapura): ec2-ap-southeast-1 - ami-5c64323f
- Ásia-Pacífico (Sydney): ec2-ap-southeast-2 - ami-a0ca20c2
- Ásia-Pacífico (Tóquio): ec2-ap-northeast-1 - ami-c6fe42a0
- Canadá (Central): ec2-ca-central-1 - ami-4b82392f
- UE (Frankfurt): ec2-eu-central-1 - ami-9e7efcf1
- UE (Irlanda): ec2-eu-west-1 - ami-8aee58f3
- UE (Londres): ec2-eu-west-2 - ami-09edf26d
- AS (São Paulo): ec2-sa-east-1 - ami-10bffa7c

Referências

- Apache MXNet
- Caffe
- TensorFlow

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Detalhes das notas de release da Deep Learning AMI (Windows 2012 R2) versão 1.0

AWS Deep Learning AMI

A AWS Deep Learning AMI é pré-compilada com o CUDA 8 e 9 e várias estruturas de deep learning.

Destaques da versão

1. CUDA 8 e 9
2. CuDNN 6 e 7

3. NCCL 2.0.5
4. CuBLAS 8 e 9
5. OpenCV 3.2.0
6. SciPy 0.19.1
7. Conda 5.01

Estruturas de deep learning pré-compiladas

- Apache MXNet: o MXNet é uma biblioteca de código aberto flexível, eficiente, portátil e escalável para deep learning. Ele oferece suporte aos modelos de programação declarativos e imperativos, em uma grande variedade de linguagens de programação, o que o torna eficiente mas simples para codificar aplicativos de deep learning. O MXNet é eficiente e inerentemente compatível com programações paralelas automáticas de partes de código-fonte que podem ser paralelizadas sobre um ambiente distribuído. MXNet também é portátil, usando otimizações de memória que permitem que seja executado em celulares para servidores completos.
  - Ramificação/tag usada: v0.12.0
  - Compilado com CUDA 8 e cuDNN 6.1
  - Justificativa: estável e bem testado
  - Diretório de origem:

    C:\MXNet
- Caffe: o Caffe é uma estrutura criada com expressão, velocidade e modularidade em mente.
  - Ramificação/tag usada: ramificação do Windows, confirmação nº 5854
  - Compilado com CUDA 8 e cuDNN 5
  - Justificativa: estável e bem testado
  - Diretório de origem:

    C:\Caffe
- TensorFlow: o TensorFlow™ é uma biblioteca de software de código aberto para computação numérica que usa gráficos de fluxo de dados.
  - Ramificação/tag usada: v1.4
  - Compilado com CUDA 8 e cuDNN 6.1
  - Justificativa: estável e bem testado
  - Diretório de origem:

    C:\ProgramData\Anaconda3\envs\tensorflow\Lib\site-packages

Suporte do Python 2.7 e do Python 3.5

O Python 2.7 e o Python 3.6 são compatíveis com a AMI para todas essas estruturas de deep learning instaladas, exceto o Caffe2:

1. Apache MXNet
2. Caffe
3. Tensorflow

Suporte de tipos de instância

Ainda não há suporte para o tipo de instância P3.

A AMI é compatível com todos os outros tipos de instâncias de CPU para todas as estruturas.

Drivers de GPU instalados

- CuDNN 6 e 7
- Nvidia 385.54
- CUDA 8 e 9

Executar a instância de deep learning

Escolha o tipo da AMI na lista a seguir na região de sua escolha e siga as etapas em:

Iniciar uma instância do Windows

Regiões da Deep Learning AMI (Windows 2012 R2)

As AMIs da Windows 2012 R2 estão disponíveis nas seguintes regiões:

- Leste dos EUA (Ohio): ec2-us-east-2 – ami-93c0eef6
- Leste dos EUA (Norte da Virgínia): ec2-us-east-1 - ami-3375f749
- GovCloud: ec2-us-gov-west-1 - ami-87018de6
- Oeste dos EUA (Norte da Califórnia): ec2-us-west-1 - ami-04bf8764
- Oeste dos EUA (Oregon): ec2-us-west-2 - ami-d2be62aa
- Ásia-Pacífico (Mumbai): ec2-ap-south-1 - ami-221f514d
- Ásia-Pacífico (Seul): ec2-ap-northeast-2 - ami-1975d277
- Ásia-Pacífico (Cingapura): ec2-ap-southeast-1 - ami-5d67313e
- Ásia-Pacífico (Sydney): ec2-ap-southeast-2 - ami-a2f71dc0
- Ásia-Pacífico (Tóquio): ec2-ap-northeast-1 - ami-96fe42f0
- Canadá (Central): ec2-ca-central-1 - ami-1a863d7e
- UE (Frankfurt): ec2-eu-central-1 - ami-cf78faa0
- UE (Irlanda): ec2-eu-west-1 - ami-3eea5c47
- UE (Londres): ec2-eu-west-2 - ami-caeef1ae
- AS (São Paulo): ec2-sa-east-1 - ami-aabefbc6

Referências

- Apache MXNet
- Caffe
- TensorFlow

Ambientes de teste

- Baseado em p2.16xlarge.
- Também testado em p2.xlarge, c4.4xlarge.

Histórico do documento do Guia do desenvolvedor do AWS Deep Learning AMI

A tabela a seguir descreve a documentação desta versão da AWS Deep Learning AMI.

- Versão da API: mais recente
- Última atualização da documentação: 11 de dezembro de 2017

Alteração	Descrição	Data
AMIs do Linux v3.0, além da apresentação do MXNet Model Server, TensorFlow Serving e TensorBoard	Tutoriais para AMIs do Conda com novos recursos de fornecimento e visualização de modelos usando MXNet Model Server v0.1.5, TensorFlow Serving v1.4.0 e TensorBoard v0.4.0. Recursos de AMI e estrutura em CUDA descritos na visão geral do Conda e CUDA. Últimas notas de release movidas para https://aws.amazon.com/releasenotes/	25 de janeiro de 2018
AMIs do Linux v2.0	AMIs de base, origem e Conda atualizadas com NCCL 2.1. AMIs de origem e Conda atualizadas com MXNet v1.0, PyTorch 0.3.0 e Keras 2.0.9.	11 de dezembro de 2017
Duas opções de AMI do Windows adicionadas	AMIs do Windows 2012 R2 e 2016 lançadas: adicionadas ao guia de seleção de AMI e adicionadas às notas de release.	30 de novembro de 2017
Versão da documentação inicial	Descrição detalhada da alteração com o link para o tópico ou a seção que sofreu a alteração.	15 de novembro de 2017

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.