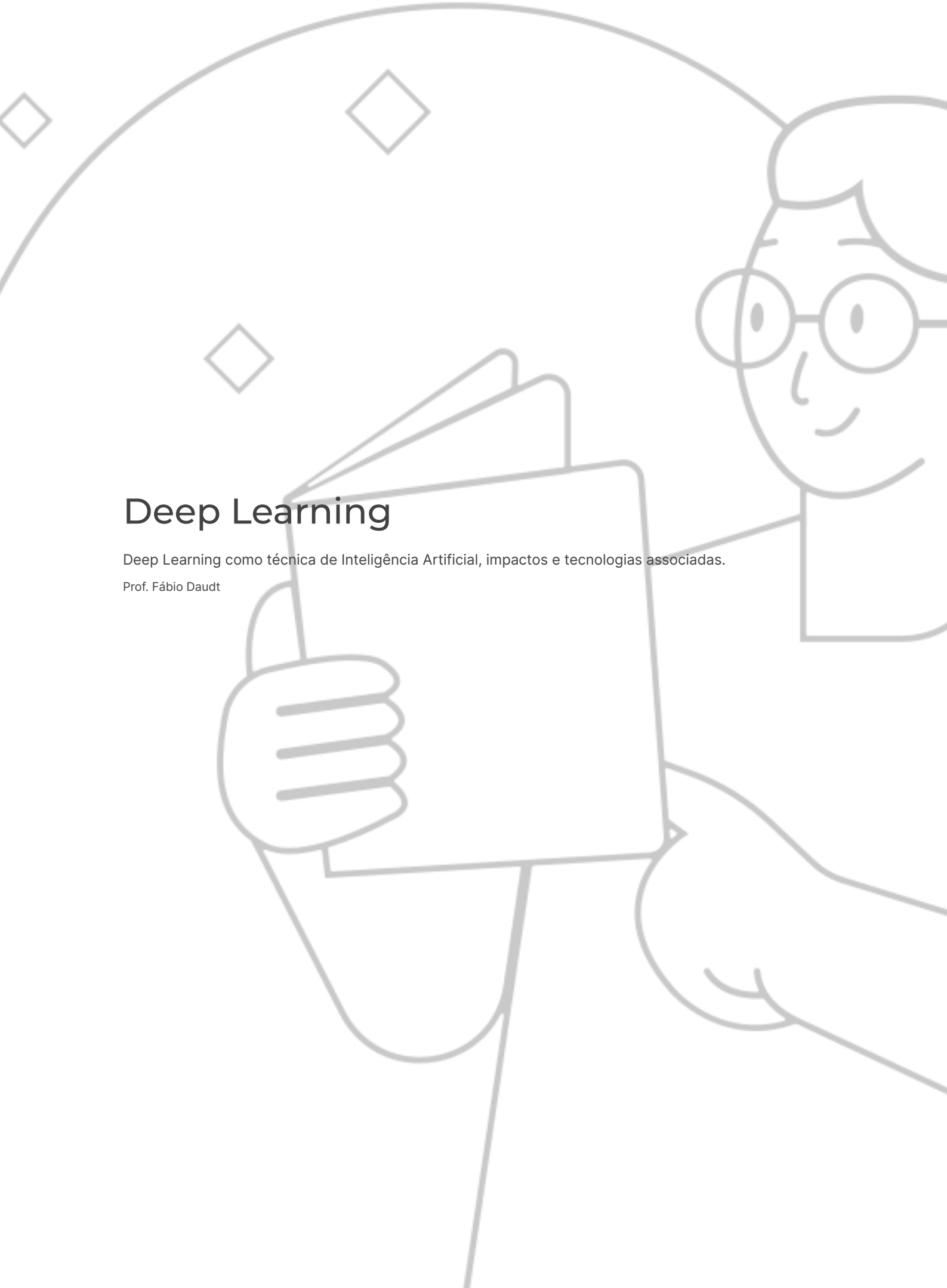


Deep Learning

Deep Learning como técnica de Inteligência Artificial, impactos e tecnologias associadas.

Prof. Fábio Daudt

Propósito

O conhecimento teórico e prático da técnica de Inteligência Artificial chamada de Deep Learning (aprendizagem profunda) aplicada no reconhecimento de voz, visão computacional e muitos outros é um importante diferencial para profissionais de tecnologia que pretendem trabalhar no desenvolvimento de produtos muito avançados que podem ser aplicados em diversas áreas.

Preparação

Antes de iniciar o estudo, verifique se possui acesso a algum editor de Python para executar o projeto proposto, tal como Jupyter notebook ou acesso ao Google Colab. Os dados a serem processados também estão incluídos no referido arquivo para **download**.

Objetivos

- Reconhecer os principais conceitos de Deep Learning e os fatores que a tornaram tão popular.
- Identificar os principais algoritmos de Deep Learning e os tipos de tarefas resolvidas.
- Identificar os principais frameworks de Deep Learning.
- Empregar um projeto prático de Deep Learning.

Introdução

As redes neurais artificiais que utilizam técnicas de Deep Learning estão sendo usadas para tratar diversas situações práticas, como reconhecimento de voz, visão computacional e muitos outros. Essa diversidade de aplicações aumentou o interesse no entendimento teórico de como elas são construídas e de como trabalham.

Outro importante fator que contribuiu para a popularização das redes neurais artificiais aplicando técnicas de Deep Learning é a disponibilidade de recursos computacionais, tais como linguagens de programação, bibliotecas e frameworks, que facilitaram bastante a implementação de programas para resolver problemas reais.

Neste conteúdo, abordaremos alguns dos conceitos fundamentais das redes neurais implementando algoritmo de Deep Learning, incluindo o estudo de suas arquiteturas e de algoritmos de aprendizagem.

Além disso, analisaremos uma área de aplicação chamada de reconhecimento de imagem, que tem muitas aplicações interessantes e que envolve o conhecimento de múltiplas disciplinas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

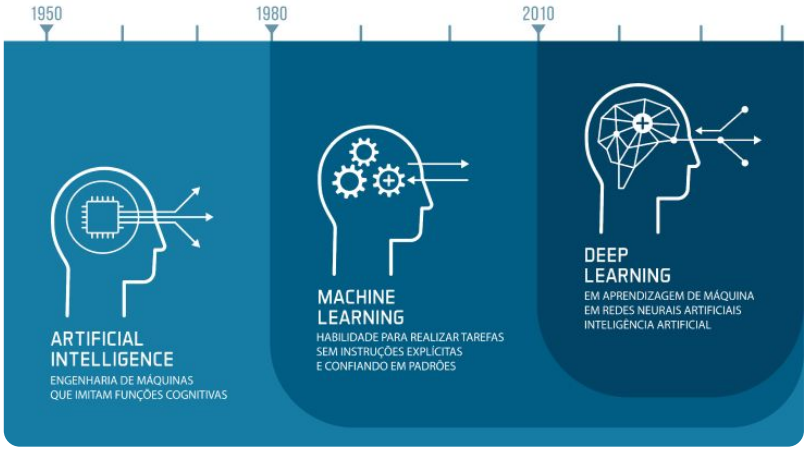
Entendendo Deep Learning

Deep Learning vs. Inteligência Artificial

Neste vídeo, comentaremos como funciona o Deep Learning vs. Inteligência Artificial.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

Nosso estudo vai analisar o Deep Learning (aprendizagem profunda), um subconjunto do conceito de Machine Learning (aprendizado de máquina) que, por sua vez, é um subconjunto de Inteligência Artificial (IA).



Vamos detalhar cada um a seguir:

Inteligência Artificial

É uma tecnologia que permite que uma máquina tome uma decisão ou ação inteligente e que um agente inteligente perceba cognitivamente seu ambiente e, correspondentemente, tente maximizar sua probabilidade de sucesso de uma ação alvo. Um agente inteligente corresponderia a um módulo de hardware, software, robô ou aplicativo.

Machine Learning

Consiste em técnicas e modelos mais avançados que podem aprender com dados e fornecer aplicativos de Inteligência Artificial. Outra definição diz que Machine Learning é a ciência de fazer os computadores agirem sem serem explicitamente programados.

Deep Learning

Emprega redes neurais artificiais de várias camadas para obter alta precisão em tarefas, como detecção de objetos, reconhecimento de fala, tradução de idiomas e outros avanços atuais que você ouve nas notícias. A beleza e o poder do Deep Learning é sua capacidade de aprender, extrair e traduzir automaticamente os recursos de conjuntos de dados, como fotos, vídeos ou textos, sem usar regras ou códigos codificados manualmente.

O Machine Learning tradicional, ou “não profundo”, depende mais da entrada humana para entender as distinções entre diferentes entradas de dados, uma vez que especialistas humanos precisam criar uma hierarquia de recursos, normalmente aprendendo com dados mais estruturados.

Exemplo

Se fossemos fornecer uma seleção de fotos de vários itens de fast food, como "pizza", "hambúrguer" ou "pastel". Um especialista humano nessas fotografias identificaria os traços que caracterizam cada imagem como uma determinada variedade de fast food. Um elemento identificador em cada imagem, por exemplo, pode ser o pão associado a cada tipo de alimento. Como alternativa, você pode acelerar o aprendizado por meio do aprendizado supervisionado simplesmente usando rótulos, como "pizza", "hambúrguer" ou "pastel".

Embora o Deep Learning possa usar conjuntos de dados rotulados, comumente chamados de aprendizado supervisionado, para orientar seu algoritmo, isso não é necessário. Ele tem a capacidade de receber entradas não estruturadas em sua forma bruta (como vídeos e fotos) e pode identificar automaticamente as características que diferenciam "pizza", "hambúrguer" e "pastel" umas das outras.

> As principais diferenças entre o Machine Learning e o Deep Learning são como o algoritmo aprende e o quantitativo de dados que cada algoritmo consome.

No Deep Learning, grande parte da extração de recursos do processo é automatizada, o que reduz a necessidade de intervenção humana manual e ele também tira proveito de conjuntos de dados massivos. Uma vez que 80% a 90% dos dados em uma organização são considerados não estruturados (imagens, vídeos e textos), essa capacidade se torna uma grande vantagem.

Deep Learning

Como funciona o Deep Learning

Neste video, apresentamos as principais reflexões sobre como funciona o Deep Learning.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

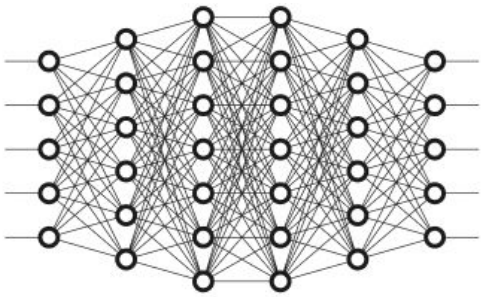
Os algoritmos de Deep Learning imitam o cérebro humano por meio do uso de entradas de dados, pesos e viés. Juntos, esses componentes identificam, categorizam e caracterizam com precisão os itens nos dados.

As redes neurais compõem a espinha dorsal dos algoritmos de Deep Learning, distinguindo-se dos demais algoritmos pelo número de camadas de nós ou pela profundidade das redes neurais.

> Portanto, um algoritmo de Deep Learning é uma rede neural que possui obrigatoriamente mais de três camadas, incluindo as entradas e as saídas, ou seja, uma rede neural que possui apenas duas ou três camadas é apenas uma rede neural básica.

As Deep Learnings são compostas por muitas camadas de nós interconectados, cada um dos quais aprimora a previsão ou categorização feita por aquele abaixo dela. A propagação direta refere-se ao movimento de cálculos através da rede. As camadas visíveis de uma rede neural profunda são suas camadas de entrada e saída.

O modelo de aprendizado profundo ingere os dados para processamento na camada de entrada e a previsão ou classificação final é realizada na camada de saída.



Rede neural Deep Learning.

A retropropagação (backpropagation) é um processo que usa técnicas como descida de gradiente (gradient descedent) para calcular os erros de previsão antes de alterar os pesos e vieses da função, voltando iterativamente pelas camadas em um esforço para treinar o modelo. Uma rede neural pode fazer previsões e fazer as correções necessárias para quaisquer falhas, graças à propagação direta e retropropagação trabalhando juntas. O algoritmo melhora continuamente a precisão ao longo do tempo.

Comentário

Vale dizer que o Deep Learning é uma técnica de Machine Learning que usa várias camadas internas (camadas ocultas) de unidades de processamento não lineares (neurônios) para realizar operações de aprendizagem supervisionadas ou não supervisionadas com os dados.

Assim, Deep Learning é uma rede **feedforward** com muitas camadas ocultas. Mas, qual seria o benefício de se ter muitas camadas ocultas?

Feedforward

Rede neural feedforward também chamada de rede de propagação direta. Sua principal característica é que o fluxo de processamento só ocorre em um único sentido que vai da camada de entrada para a camada de saída.

> Teoricamente, quanto maior o número de camadas ocultas, maior será o nível de acurácia do modelo.

Os fatores que tornaram as redes Deep Learning tão populares

Neste vídeo, comentaremos sobre os fatores que tornaram as redes Deep Learning tão populares.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Muitos aplicativos e serviços de Inteligência Artificial são baseados em Deep Learning, que aprimora a automação ao realizar tarefas mentais e físicas sem a necessidade de intervenção humana. O Deep Learning é a tecnologia que alimenta tanto as tecnologias estabelecidas (como assistentes digitais, controles remotos de TV ativados por voz e sistemas de recomendação de compras) quanto as tecnologias emergentes, como carros autônomos.

As redes Deep Learning existem já há algum tempo, porém se tornaram populares recentemente e alguns dos fatores que contribuíram para tanto são: computação em nuvem, maior disponibilidade de dados, escalas de desempenho com dados, inovação de hardware e os sistemas integrados de software e hardware, servidores e inovações de rede.

Computação em nuvem

O declínio dos custos computacionais é definitivamente uma das forças motrizes. Suporte para computação em nuvem, bem como dispositivo conectado a uma nuvem podem receber armazenamento de dados, análise de dados, funções de aplicativos e serviços de inteligência de controle. Isso inclui o que temos como computação em nuvem, em termos de SaaS (software como serviço), PaaS (plataforma como serviço) e IaaS (infraestrutura como serviço).

Maior disponibilidade de dados

Há maior qualidade de coleta de dados por meio de filtros e bancos de dados inteligentes, além da coleta de Big Data em tempo real por meio de smartphones, redes sociais e sensores IoT (Internet das Coisas). Além disso, os frameworks de Big Data, como o MapReduce e HDFS do Hadoop, o Hadoop Distributed File System, que permitem a extração rápida de dados estruturados, dados semiestruturados e dados não estruturados em tempo real, permitem, assim, que o algoritmo de Deep Learning trabalhe diretamente com esses resultantes. Dessa forma, os dados utilizados serão treinados pelo sistema e, em seguida, fornecerão resultados precisos sobre a estrutura geral dos dados que foram coletados dos bancos de dados.

Escalas de desempenho de dados

A performance das redes de Deep Learning aumenta de acordo com a quantidade de dados disponíveis na atualidade, de fontes estruturadas, semiestruturadas e não estruturadas, permitindo:

- Aprendizado de máquina aprimorado, aprendizado profundo e tecnologia de Big Data fazem melhor uso dos dados.
- Resultados de engenharia de recursos mais rápidos.
- Servidores e redes mais rápidos.
- Computação de distribuição poderosa.

Além disso, a computação de distribuição de Big Data, como a tecnologia MapReduce e HDFS do Hadoop, é fonte de direcionamento que permite resultados de engenharia de recursos mais rápidos.

Inovação de hardware

Temos também inovações de hardware, tais como:

- Suporte de CPUs e **GPUs** muito mais poderosas.
- CPUs de smartphones, que são CPUs multicores e as novas arquiteturas Big Data.
- Processamento eficiente de baixo consumo de energia em dispositivos móveis, que estão em smartphones, dispositivos de realidade aumentada, plataformas IoT que suportam poderosa computação distribuída.

GPUs

Um processador composto por muitos núcleos menores e mais especializados. Trabalhando em conjunto, os núcleos oferecem desempenho massivo quando uma tarefa de processamento pode ser dividida e executada em muitos núcleos

Sistemas integrados de software e hardware, servidores e inovações de rede

Nesse contexto, podemos destacar a confiabilidade aprimorada, pois existem aplicações compreensíveis e exemplos práticos que mostram como o Deep Learning pode ser usado, bem como hardware, software, modelos de biblioteca de fácil utilização e a existência de modelos abertos pré-treinados.

Prós e contras das redes Deep Learning

Vejamos a seguir alguns pontos positivos e negativos das redes Deep Learning:

Positivos

- Desempenho de última geração.
- Arquitetura facilmente adaptável a vários domínios de problemas.
- Redução da necessidade de engenharia de recursos.

Negativos

- Requer muitos dados de treinamento limpos.
- Requer alto poder computacional.
- Longo tempo de treinamento necessário.
- Alto custo de implementação.
- Dificuldade na interpretação dos resultados.

Verificando o aprendizado

Questão 1

Deep Learning, ou aprendizagem profunda, em português, nada mais é do que um ramo do Machine Learning, ou seja, é um aprendizado de máquina que visa “ensinar” as máquinas a agirem e interpretarem dados de uma maneira mais natural. Qual o relacionamento entre Inteligência Artificial (AI), Machine Learning (ML) e Deep Learning (DL)? Considere o símbolo ”>” como relacionamento “possui a subárea”.

A

AI > ML > DL

B

Não existe relacionamento entre AI, ML e DL

C

DL > AI > ML

D

ML > AI > DL

E

DL > ML > AI

A alternativa A está correta.

Inteligência Artificial (AI) possui algumas subáreas, entre elas a área de Machine Learning (ML). O Deep Learning é uma subárea de Machine Learning.

Questão 2

O engenheiro de Machine Learning é responsável pela definição, implementação e manutenção de modelos de ML. Por meio do aprendizado dos computadores, será possível encontrar soluções e escalar processos para facilitar a tomada de decisões. Quais foram os principais fatores na adoção massiva do Deep Learning na última década?

A

Educação em Deep Learning.

B

Computação mais rápida e eficiente (GPUs) e maior quantidade de dados.

C

Facilidade de implantação de projetos de Deep Learning.

D

Número de cientistas de dados.

E

Pouca quantidade de dados disponíveis atualmente.

A alternativa B está correta.

Definitivamente, o surgimento de computação rápida e eficiente (GPUs) e a disponibilidade de dados (Big Data) de fontes variadas impulsionaram o crescimento e adoção massiva do Deep Learning.

Convolutional Neural Networks (CNN)

Neste vídeo, apresentaremos a rede neural Convolutional Neural Networks (CNN), com destaque para as suas camadas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Aplicação da CNN

Uma rede neural convolucional (CNN) usa uma Rede Neural FeedForward. Essas redes aproveitam os princípios da álgebra linear, particularmente, a multiplicação de matrizes para identificar padrões dentro de uma imagem.

A CNN usa uma Multi-Layer Perceptrom (MLP) para fazer esse processo convolucional. As técnicas de aprendizado profundo da CNN tornaram-se conhecidas com base em sua eficácia no reconhecimento de imagem. As CNNs precisam de uma quantidade mínima de pré-processamento.

Saiba mais

As redes neurais convolucionais (CNN) são geralmente utilizadas para reconhecimento de imagem e vídeo, detecção de objetos, processamento de linguagem natural (classificação de documentos), reconhecimento de caracteres, jogos (xadrez e Go), reconhecimento de padrões e/ou visão computacional.

A visão computacional é um campo da Inteligência Artificial (IA) que permite que computadores e sistemas obtenham informações significativas de imagens digitais, vídeos e outras entradas visuais e, com base nessas entradas, possam agir. Essa capacidade de fornecer recomendações a distingue das tarefas de reconhecimento de imagem.

Camadas principais

As redes neurais convolucionais são diferenciadas de outras redes neurais por seu desempenho superior com entradas de sinal de imagem, fala ou áudio. Eles têm três tipos principais de camadas: camada convolucional (Convolutional layer), camada de pooling (Pooling Layer) e camada totalmente conectada (Fully-Connected Layer).

Camada convolucional (Convolutional Layer)

Enquanto as camadas convolucionais podem ser seguidas por camadas convolucionais adicionais ou camadas de agrupamento, a camada totalmente conectada é a camada final.

A cada camada, a CNN aumenta em sua complexidade, identificando maiores porções da imagem. As camadas anteriores se concentram em recursos simples, como cores e bordas.

À medida que os dados da imagem progridem pelas camadas da CNN, ela começa a reconhecer elementos ou formas maiores do objeto até finalmente identificar o objeto pretendido.

> A camada convolucional é a primeira camada de uma rede convolucional.

Ela é o bloco de construção central de uma CNN e é onde ocorre a maior parte da computação. Requer alguns componentes como dados de entrada, um filtro e um mapa de recursos (features).

Por exemplo, se a entrada for uma imagem colorida, composta por uma matriz de pixels em 3D, isso significa que a entrada terá três dimensões — altura, largura e profundidade — que correspondem ao RGB em uma imagem.

Também temos um detector de feição, também conhecido como kernel ou filtro, que percorrerá os campos receptivos da imagem, verificando se a feição está presente. Esse processo é conhecido como convolução.

Por fim, a camada convolucional converte a imagem em valores numéricos, permitindo que a rede neural interprete e extraia padrões relevantes.

Camada de pooling (Pooling Layer)

O agrupamento de camadas, também conhecido como downsampling, realiza a redução de dimensionalidade, reduzindo o número de parâmetros na entrada. Semelhante à camada convolucional, a operação de pooling varre um filtro em toda a entrada, mas a diferença é que esse filtro não possui pesos. Em vez disso, o kernel aplica uma função de agregação aos valores dentro do campo receptivo, preenchendo a matriz de saída. Existem dois tipos principais de pooling:

1. ### Max pooling
   À medida que o filtro se move pela entrada, ele seleciona o pixel com o valor máximo para enviar para a matriz de saída. Essa abordagem tende a ser usada com mais frequência em comparação ao average pooling.

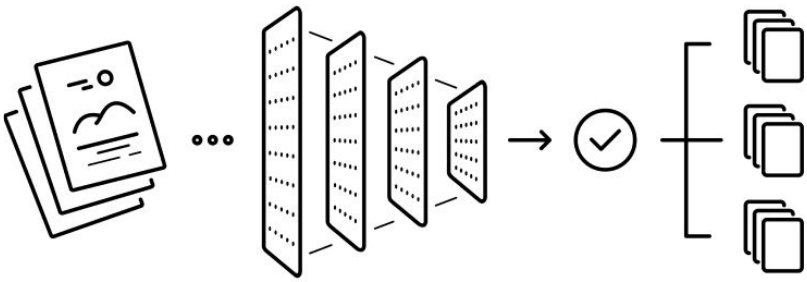
2. ### Average poooling
   À medida que o filtro se move pela entrada, ele calcula o valor médio dentro do campo receptivo para enviar para a matriz de saída. Embora muitas informações sejam perdidas na camada de pooling, elas ajudam a reduzir a complexidade, melhorar a eficiência e limitar o risco de overfitting em uma rede CNN. O ajuste excessivo (overfitting) é um comportamento indesejável de aprendizado de máquina que ocorre quando o modelo de aprendizado de máquina fornece previsões precisas para dados de treinamento, mas não para novos dados.

Camada totalmente conectada (Fully-Connected Layer – FC)

O nome da camada totalmente conectada se descreve adequadamente, pois é nela que cada nó na camada de saída se conecta diretamente a um nó na camada anterior. Inicialmente, os valores de pixel da imagem de entrada não estão diretamente conectados à camada de saída em camadas parcialmente conectadas.

Essa camada realiza a tarefa de classificação com base nas características extraídas das camadas anteriores e seus diferentes filtros. Enquanto as camadas convolucionais e de pooling tendem a usar funções ReLu, as camadas FC geralmente aproveitam uma função de ativação softmax para classificar as entradas adequadamente, produzindo uma probabilidade de 0 a 1.

Em redes neurais artificiais, a função de ativação de um nó define a saída desse nó dada uma entrada ou conjunto de entradas, sendo a ReLU uma função de ativação bem como a Softmax.



Camada totalmente conectada.

Recurrent Neural Network (RNN)

Neste vídeo, falaremos sobre a Recurrent Neural Network (RNN).

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

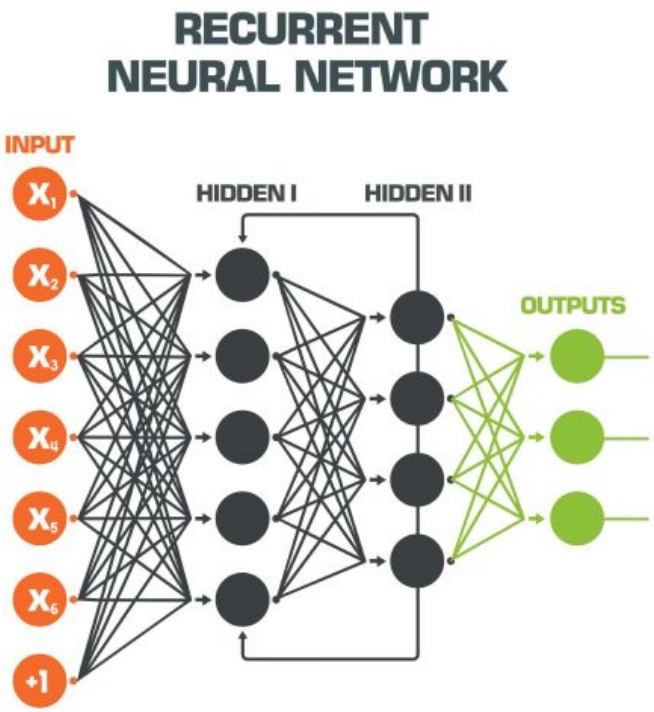
Aplicação da RNN

Uma rede neural recorrente (RNN) é um tipo de rede neural artificial que usa dados sequenciais ou dados de séries temporais. Esses algoritmos de aprendizado profundo são comumente usados para problemas ordinais ou temporais, como tradução de idiomas, processamento de linguagem natural, reconhecimento de fala e legenda de imagens.

Vamos observar algumas diferenças entre redes neurais:

- Assim como as redes neurais de feedforward e convolucionais (CNNs), as redes neurais recorrentes utilizam dados de treinamento para o aprendizado, ou seja, são considerados algoritmos de aprendizagem supervisionado. Eles se distinguem por sua “memória”, pois obtêm informações de entradas anteriores para influenciar a entrada e a saída atuais.
- Enquanto as redes neurais profundas tradicionais assumem que as entradas e saídas são independentes umas das outras, a saída das redes neurais recorrentes depende dos elementos anteriores dentro da sequência. Embora eventos futuros também sejam úteis para determinar a saída de uma determinada sequência, as redes neurais recorrentes unidirecionais não podem explicar esses eventos em suas previsões.

As redes neurais recorrentes (RNNs) são identificadas por seus loops de feedback. Esses algoritmos de aprendizado são aproveitados principalmente ao usar dados de séries temporais para fazer previsões sobre resultados futuros (previsões do mercado de ações ou previsão de vendas), chatbots, assistente de voz, previsão de tempo e geração de música.



Recurrent Neural Network (RNN).

Algumas variações de arquitetura das RNN

A seguir, apresentamos algumas das variações de arquiteturas das redes neurais recorrentes (RNN):

Long Short-Term Memory (LSTM)

A arquitetura Long Short-Term Memory (LSTM) tornou as redes neurais recorrentes muito populares quando revolucionou os programas de reconhecimento de fala. As redes LSTM são agora a base para muitos dos aplicativos mais bem sucedidos de hoje no domínio de reconhecimento de fala, domínio text-to-speech e domínio de reconhecimento de escrita manual.

> Essa é uma arquitetura RNN que foi criada com o propósito de ser uma solução para problema de vanishing gradiente ou problema da dissipação do gradiente, que é comum em redes neurais recorrentes (RNN).

Os gradientes são valores usados para atualizar os pesos das redes neurais, portanto o problema da dissipação do gradiente ocorre quando o gradiente diminui à medida que se propaga novamente ao longo do tempo. Se um valor de gradiente se torna extremamente pequeno, não contribui muito com o aprendizado.

O objetivo foi abordar o problema das dependências em longo prazo, ou seja, se o estado anterior que está influenciando a previsão atual não estiver no passado recente, o modelo RNN pode não ser capaz de prever com precisão o estado atual.

Exemplo

Se quisermos prever que: “Caroline é alérgica a leite, ela não pode comer manteiga”, então o contexto de uma alergia a leite pode ajudar a prever que o alimento que não pode ser consumido contenha leite. No entanto, se esse contexto for algumas frases anteriores, isso tornará difícil ou mesmo impossível para o RNN conectar as informações.

Para resolver essa questão, as redes LSTMs têm “células” nas camadas ocultas da rede neural, que possuem três portas – uma porta de entrada, uma porta de saída e uma porta de esquecimento. Essas portas controlam o fluxo de informações necessário para prever a saída na rede.

Bidirectional Recurrent Neural Networks (BRNN)

RNNs bidirecionais extraem dados futuros para melhorar a precisão deles, enquanto RNNs unidirecionais só podem ser extraídas de entradas anteriores para fazer previsões sobre o estado atual.

Gated Recurrent Units (GRUs)

Essa variante RNN funciona para resolver o problema de memória de curto prazo (short-term memory) das redes RNN, assim como as redes LSTMs. Ao invés de usar uma informação de regulação de “estado de célula”, usa estados ocultos e, em vez de três portas, tem duas portas, sendo uma porta de reinicialização e outra de atualização. Semelhante aos portões dentro das redes LSTMs, os portões de reinicialização e atualização controlam quantas e quais informações retêm.

Autoencoders

Redes Autoencoders e Generative Adversarial Networks (GANs)

Neste vídeo, apresentaremos os principais conceitos do funcionamento das redes Autoencoders e Generative Adversarial Networks (GANs).

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Uma rede Autoencoder é uma rede neural pré-treinada com uma camada aberta que usa um algoritmo de **aprendizado não supervisionado** e retropropagação.

Os Autoencoders têm forte semelhança com redes neurais Multi-Layer Perceptrons, pois possuem uma camada de entrada, camadas ocultas de neurônios e, em seguida, uma camada de saída. A principal diferença entre uma Multi-Layer Perceptron e uma rede de Autoencoder é a camada de saída em um Autoencoder, que tem o mesmo número de unidades que a camada de entrada.

Autoencoders são usadas para aprender representações compactadas de conjuntos de dados, bem como para reduzir a dimensionalidade de um conjunto de dados, compressão de imagens, detecção de anomalias, detecção de fraudes e extração de features.

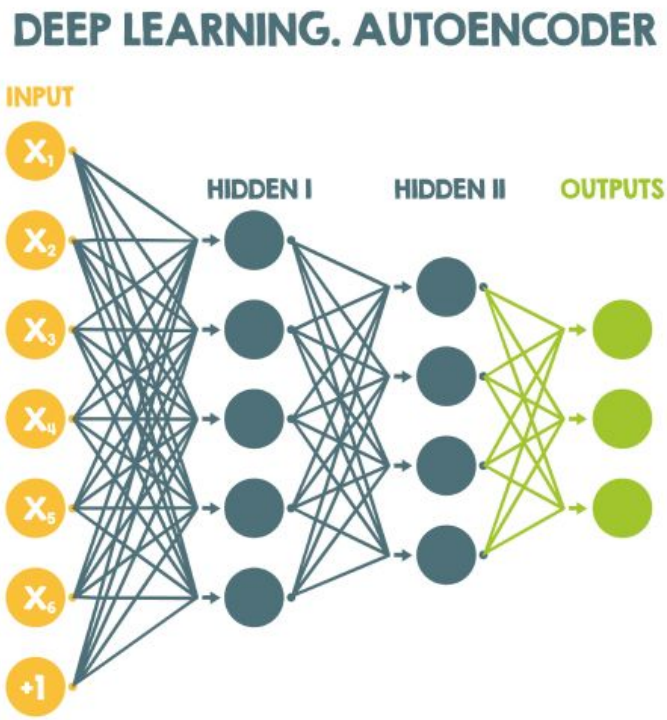
A **saída da rede** de Autoencoder é uma reconstrução dos dados de entrada da forma mais eficiente.

O Autoencoder é um **auto-associador** (AA), cuja finalidade é receber na saída a imagem mais precisa da entrada quanto possível.

Existem dois tipos de auto-associador (AA), as **geradoras** e as **sintetizadoras**. Uma máquina de Boltzmann restrita pertence ao primeiro tipo e um Autoencoder representa o segundo tipo, pois define um valor objetivo igual ao vetor de entrada, ou seja, y = x.

Aprendizado não supervisionado

O aprendizado não supervisionado consiste em treinar uma máquina a partir de dados que não estão rotulados e/ou classificados. Os algoritmos que fazem isso buscam descobrir padrões ocultos que agrupam as informações de acordo com semelhanças ou diferenças, por exemplo.



Redes Autoencoders.

Generative Adversarial Networks (GANs)

As redes GANs usam aprendizado não supervisionado para treinar dois modelos em paralelo, colocando, dessa forma, uma contra a outra, dando origem ao termo “adversária”.

Modelos de GANs são utilizadas para sintetizar novas imagens com base em outras imagens de treinamento, sendo muito utilizadas também para modelar outros domínios, tais como: som, vídeo e geração de imagens de descrições de texto.

> Um aspecto-chave dos GANs (e modelos generativos em geral) é como eles usam uma contagem de parâmetros que é significativamente menor do que o normal em relação à quantidade de dados nos quais estamos treinando a rede.

A rede é forçada a representar eficientemente os dados de treinamento, tornando-o mais eficaz na geração de dados semelhantes aos dados de treinamento.

Verificando o aprendizado

Questão 1

Uma empresa de análise de segurança e transporte deseja usar as imagens internas dos funcionários de seus clientes para classificar se os funcionários devem passar pelo portão de segurança nos escritórios dos clientes. Qual algoritmo eles devem usar para criar esse modelo de Deep Learning?

A

Long Short-Term Memory Network (LSTM)

B

Recurrent Neural Network (RNN)

C

Convolutional Neural Network (CNN)

D

Autoencoders

E

Adversarial Generative Networks (GANs)

A alternativa C está correta.

A Convolutional Neural Networks (CNN) é reconhecida pelos ótimos resultados em classificação de imagens e visão computacional.

Questão 2

Uma famosa empresa de entretenimento planeja sintetizar novas imagens de personagens com base em outras imagens de treinamento do seu respectivo banco de dados de imagens. Qual algoritmo eles devem usar para criar esse modelo de Deep Learning?

A

Recurrent Neural Network (RNN)

B

Convolutional Neural Network (CNN)

C

Autoencoders

D

Adversarial Generative Networks (GANs)

E

Long Short-Term Memory Network (LSTM)

A alternativa D está correta.

As redes Adversarial Generative Networks (GANs) são reconhecidas pelos ótimos resultados para sintetizar novas imagens com base em outras imagens de treinamento, sendo muito utilizadas também para modelar outros domínios, tais como: som, vídeo e geração de imagens de descrições de texto.

Ferramentas para Deep Learning

Uma ferramenta de deep learning pode ser um framework ou uma biblioteca. Por isso, antes de falarmos sobre as ferramentas, vamos definir o que é um framework e uma biblioteca.

Framework são ferramentas que estabelecem um intermédio entre a pessoa programadora e o código de uma aplicação, como uma espécie de interface de comunicação. São utilizados para automatizar tarefas, de modo a tornar o trabalho pesado mais abstrato e menos complexo para as pessoas da área.

Os frameworks provêem soluções que já foram implementadas em outro contexto e que foram amplamente testados e validadas por várias pessoas na comunidade, e assim é possível termos menos erros e bugs inesperados, promovendo assim, maior segurança. No geral, é possível obter uma padronização maior dos códigos, o que favorece a legibilidade e a manutenção. Um framework, geralmente, é uma estrutura muito bem documentada, com uma forma específica para ser implementada, o que garante organização e limpeza.

Uma biblioteca pode ser definida puramente como um conjunto de funções específicas: cálculos científicos, a parte de gráficos, visualização de dados, entre outras. O conceito de framework é mais complexo do que o de biblioteca.

Vejamos os principais frameworks e bibliotecas de deep learning.

TensorFlow

Neste vídeo, apresentaremos a ferramenta para Deep Learning TensorFlow.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O TensorFlow é uma plataforma completa que permite a criação e a implantação de modelos de Machine Learning, sendo baseada em código aberto e foi desenvolvido pelo Google Brain Team. Permite a programação em Python ou C++ e possui interface para Python, C, C++, Java, Go, podendo ser implantada em diferentes plataformas computacionais, tais como em várias TPUs, CPUs, GPUs.

Vejamos algumas características do TensorFlow:

Permite a preparação dos dados

O TensorFlow oferece várias ferramentas de dados para ajudar você a consolidar, limpar e pré-processar dados em escala. Ferramentas para validar e transformar grandes conjuntos de dados e ferramentas de IA responsáveis que auxiliam o desenvolvedor a descobrir e eliminar vieses em seus dados para produzir resultados justos e éticos de seus modelos.

Criação de modelos de Machine Learning

Opção de usar modelos pré-treinados ou criação de modelos personalizados.

Implantação de modelos

Opção de executar no local, no dispositivo, no navegador ou na nuvem. O TensorFlow oferece recursos robustos para implantar seus modelos em qualquer ambiente, tais como: servidores, dispositivos de borda, navegadores, dispositivos móveis, microcontroladores, CPUs, GPUs, FPGAs. O TensorFlow Serving pode executar modelos de Machine Learning em escala de produção nos processadores mais avançados do mundo, incluindo as Tensor Processing Units (TPUs) personalizadas do Google. Além disso, o TensorFlow Lite permite executar modelos em dispositivos móveis, dispositivos de computação de borda e até microcontroladores. Já o TensorFlow.js permite executar Machine Learning com apenas um navegador da Web.

Implementar MLOps

MLOps é uma abreviação para operações de Machine Learning, ou seja, Machine Learning Operations, que é um conjunto de práticas recomendadas para que as empresas executem a IA com sucesso. Permite executar modelos em produção e mantendo-os funcionando. A plataforma TensorFlow permite implementar as melhores práticas para automação de dados, rastreamento de modelos, monitoramento de desempenho e retreinamento de modelos.

Microsoft CNTK

Ferramentas para Deep Learning Microsoft CNTK e Keras

Neste vídeo, apresentaremos as ferramentas para Deep Learning Microsoft CNTK e Keras.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O Microsoft Cognitive Toolkit (CNTK) é um kit de ferramentas de código aberto para Deep Learning distribuído comercialmente. Ele descreve as redes neurais como uma série de etapas computacionais por meio de um grafo direcionado. O CNTK permite ao usuário realizar e combinar facilmente alguns tipos de redes de Deep Learning populares como:

- DNNs feed-forward – uma rede neural deconvolucional é uma rede neural que executa um modelo de convolução inversa. Alguns especialistas referem-se ao trabalho de uma rede neural deconvolucional como a construção de camadas a partir de uma imagem em uma direção ascendente, enquanto outros descrevem os modelos deconvolucionais como “engenharia reversa” dos parâmetros de entrada de um modelo de rede neural convolucional.
- Redes neurais convolucionais (CNNs), abordadas anteriormente.
- Redes neurais recorrentes (RNNs/LSTMs).

O CNTK implementa o aprendizado de descida de gradiente estocástico, ou seja, o método do gradiente estocástico (Stochastic Gradient Descent, SGD) com diferenciação automática e paralelização em várias GPUs e servidores.

Saiba mais

A palavra “estocástico” significa um sistema ou processo que está vinculado a uma probabilidade aleatória, portanto, em Stochastic Gradient Descent, algumas amostras são selecionadas aleatoriamente em vez de todo o conjunto de dados para cada iteração.

O CNTK pode ser incluído como uma biblioteca em seus programas Python, C# ou C++ ou usado como uma ferramenta autônoma de aprendizado de máquina por meio de sua própria linguagem de descrição de modelo (BrainScript). Além disso, você pode usar a funcionalidade de avaliação do modelo CNTK por meio de programas Java.

Uma recente nota no site do projeto CNTK diz que o Microsoft CNTK não é mais desenvolvido ativamente. A versão mais recente do CNTK é 2.7.

Keras

Keras é uma biblioteca de Deep Learning baseada em Python que oferece suporte à criação de redes neurais convolucionais (CNN) e redes neurais recorrentes (RNN) para execução no TensorFlow ou Theano.

Para sua utilização, é necessária a criação de códigos similares aos aplicados na linguagem Python e, por ser de código aberto para qualquer pessoa, pode ser utilizado gratuitamente.

O Keras foi desenvolvido com base no TensorFlow, sendo considerado de fácil utilização.

Veja alguns destaques do Keras:

Destaque 1

Permite a implementação de modelos de Machine Learning em grande escala. Sendo construído com base no TensorFlow 2, o Keras é um framework robusto que pode ser dimensionado para grandes clusters de GPUs ou TPU.

Destaque 2

Permite implementação em qualquer ambiente, possibilitando aproveitar os recursos completos de implantação da plataforma TensorFlow. Você pode exportar modelos Keras para JavaScript para executar diretamente no navegador, para TF Lite para executar em dispositivos iOS, Android e incorporados. Também é fácil servir modelos Keras por meio de uma API Web.

Um vasto ecossistema Keras é uma parte central do ecossistema TensorFlow 2, cobrindo todas as etapas do fluxo de trabalho de um projeto de Machine Learning, desde o gerenciamento de dados até o treinamento de hiperparâmetros e soluções de implantação.

Caffe

Ferramentas para Deep Learning Caffe, Theano e MXNet

Neste vídeo, apresentaremos as ferramentas para Deep Learning Caffe, Theano e MXNet.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Caffe é um framework aberto de Deep Learning que é compatível com interfaces como C, C++, Python e MATLAB, sendo desenvolvido pela Berkeley AI Research (BAIR) e por colaboradores da comunidade. Yangqing Jia criou o projeto durante seu doutorado na UC Berkeley.

Sua aplicabilidade na modelagem de redes neurais convolucionais (CNN) e sua velocidade o tornaram popular nos últimos anos. O framework é muito utilizado em tarefas de visão computacional, mas o suporte para as redes neurais recorrentes (RNN) é bem limitado.

Alguns destaques do Caffe:

- A biblioteca C++ vem com uma interface Python.
- A configuração define modelos sem hard-coding.
- Mais fácil de configurar e treinar, sem ter que construir na rede.
- O suporte para redes neurais recorrentes é bastante ruim.

Theano

Theano é uma biblioteca Python que permite definir, otimizar e avaliar expressões matemáticas, especialmente aquelas com arrays multidimensionais (numpy.ndarray). Foi construído com base na biblioteca NumPy.

Saiba mais

A biblioteca Theano permite atingir velocidades que rivalizam com implementações C feitas à mão para problemas envolvendo grandes quantidades de dados. Ele também pode superar C em uma CPU em muitas ordens de magnitude, aproveitando as GPUs recentes.

Theano combina aspectos de um sistema de álgebra computacional (CAS) com aspectos de um compilador otimizado. Ele também pode gerar código C personalizado para muitas operações matemáticas. Essa combinação de CAS com compilação otimizada é particularmente útil para tarefas nas quais expressões matemáticas complicadas são avaliadas repetidamente e a velocidade de avaliação é crítica. É utilizado em situações nas quais muitas expressões diferentes são avaliadas, uma vez que o Theano pode minimizar a quantidade de sobrecarga de compilação/análise e, ainda, fornecer recursos simbólicos, como diferenciação automática.

Theano tem impulsionado pesquisas científicas computacionalmente intensivas em larga escala desde 2007. Alguns destaques do Theano:

- Integração com o NumPy, pois uma interface semelhante à do NumPy, os numpy.ndarrays também são usados internamente em funções compiladas em Theano.
- Uso transparente de GPU, permitindo a execução de cálculos intensivos de dados até 140x mais rápido do que em uma CPU (suporte apenas para float32).
- Diferenciação simbólica eficiente, pois o Theano pode calcular derivadas para funções de uma ou várias entradas.
- Otimizações de velocidade e estabilidade.
- Geração dinâmica de código C, permitindo avaliar expressões mais rapidamente.

- Extenso teste de unidade e autoverificação: inclui ferramentas para detectar e diagnosticar bugs e/ou problemas potenciais.

PyTorch

PyTorch é um framework de Deep Learning de código aberto que acelera o caminho da prototipagem de pesquisa à implantação de produção.

Permite a criação de modelos de Deep Learning e executar cálculos de tensores de alta complexidade usando GPUs e CPUs, sendo um tensor um array n-dimensional.

Ele emprega CUDA junto com bibliotecas C/C++ para o processamento e para escalar a produção de modelos de construção e flexibilidade geral.

> A plataforma CUDA (Compute Unified Device Architecture) baseia-se em uma extensão da linguagem de programação C e consiste em um compilador e uma API (Application Programming Interface) de programação, permitindo a geração de programas para serem executados em GPUs.

O PyTorch é executado em Python, o que significa que qualquer pessoa com um conhecimento básico de Python pode começar a construir seus modelos de Deep Learning.

Nos últimos anos, o PyTorch teve um alto nível de adoção na comunidade, sendo considerado um concorrente do TensorFlow. Alguns destaques do PyTorch:

- Excelente em prototipagem rápida.
- Forte suporte para GPUs, pois programas paralelos podem ser implementados em várias GPUs.
- Fornece interface mais limpa, sendo mais fácil de usar.
- Facilita a troca de dados com bibliotecas externas, pois o ecossistema de ferramentas e bibliotecas estende o PyTorch, oferecendo suporte ao desenvolvimento em visão computacional, processamento de linguagem natural e outros.
- Pronto para produção.
- Treinamento distribuído.
- Suporte na nuvem, pois é suportado pelas principais plataformas de nuvem.

MXNet

MXNet (pronunciado como mix-net) é um framework de Deep Learning de código aberto que tem suporte para Python, R, C++ e Julia. Com o back-end escrito em C++ e CUDA, o MXNet pode escalar e trabalhar com uma infinidade de GPUs.

O MXNet suporta redes de Long Short-Term Memory (LTSM), juntamente com RNN e CNN, permitindo, também, que o usuário codifique em uma variedade de linguagens de programação (Python, C++, R, Julia e Scala, para citar algumas).

Resumindo

Isso significa que você pode treinar seus modelos de Deep Learning com uma linguagem de programação com a qual se sinta confortável.

O que torna o MXNet um dos frameworks de Deep Learning preferidos é a sua funcionalidade de treinamento distribuído. O MXNet foi projetado especificamente para alta eficiência, produtividade e flexibilidade. A

Amazon empregou o MXNet como sua biblioteca de referência para Deep Learning.

Alguns destaques do MXNet:

- Programação híbrida que fornece o melhor da programação imperativa e simbólica.
- Fornece treinamento distribuído.
- Suporta implantação em diferentes linguagens, como Java, Scala, R, Julia, C++, Perl e Clojure.
- Suporta clusters de GPU para aumentar escalabilidade.

Verificando o aprendizado

Questão 1

Machine Learning exige que você não pode criar tudo do zero, especialmente se estiver em um ambiente de negócios onde os prazos de entrega de projetos são curtos. Portanto, na maioria dos casos, você precisa de um framework para ajudar a colocar seus projetos em prática. Dentre as opções a seguir, qual framework de Deep Learning não é considerado de código aberto (open source)?

A

Microsoft CNTK (Cognitive Tookit)

B

Keras

C

MATLAB

D

TensorFlow

E

MXNet

A alternativa C está correta.

Todos os frameworks de Deep Learning apresentados são baseados em código livre (open source), sendo que o MATLAB é um software proprietário.

Questão 2

Existem alguns frameworks com aplicações voltadas especificamente para a Inteligência Artificial, com diversos recursos e algoritmos diferentes – como gradiente descendente e gradiente descendente estocástico, por exemplo. Nos frameworks, todos esses algoritmos já estão otimizados e os códigos que rodam essas funções já estão implementados de maneira que podem simplesmente ser importados e rodados automaticamente. Dentre os frameworks a seguir, qual deles foi desenvolvido e é mantido pelo Google?

A

PyTorch

B

TensorFlow

C

Caffe

D

Microsoft CNTK (Cognitive Toolkit)

E

Keras

A alternativa B está correta.

De todos os frameworks de Deep Learning apresentados, o TensorFlow foi o framework criado e mantido pelo Google.

Preparação do ambiente

Neste vídeo, vamos falar sobre a preparação do ambiente.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Instalando os pacotes

A fim de que você possa executar o código a seguir, utilize o Jupyter notebook ou Google Colab. De modo geral, o mundo da computação é colaborativo, sendo comum, quando encontramos erros nas bibliotecas utilizadas, copiarmos o log de erro e procurarmos em qualquer motor de busca. Provavelmente, alguma resposta será retornada em algum fórum de discussão, como StackOverflow, GitHub, Reddit, entre outros. Não só isso é uma prática comum na comunidade de desenvolvimento de software e computação, como também nos possibilita aprender cada vez mais.

O nome do projeto que vamos criar é “Detecção de Câncer de Pulmão em Imagens Médicas com Deep Learning”.

Atenção

Importante! As imagens usadas neste projeto contêm somente a lesão no pulmão, sendo que algumas indicam câncer e outras não. São apenas imagens de exemplo e nosso foco é na construção do modelo.

Inicialmente, você deverá atualizar os pacotes necessários à execução do exemplo prático a ser implementado, então execute o comando a seguir no terminal ou prompt de comando:

```
python
pip install -U nome_pacote
```

Para instalar a versão exata de um pacote, execute o comando a seguir no terminal ou prompt de comando:

```
python
pip install nome_pacote==versão_desejada
```

Depois de instalar ou atualizar o pacote, reinicie o Jupyter notebook.

Vamos à instalação do pacote watermark, sendo esse pacote usado para gravar as versões de outros pacotes usados neste Jupyter notebook.

```
python

pip install -q -U watermark
```

Os seguintes pacotes serão também necessários:

Matplotlib

É uma biblioteca Python de plotagem 2D, que auxilia a biblioteca matemática NumPy.

pip install -q -U matplotlib==3.2.1

OpenCV

É uma biblioteca de código aberto voltada para as áreas de visão computacional e aprendizado de máquina.

pip install -q -U cv2==4.2.0

NumPy

É uma biblioteca para a linguagem de programação Python, que suporta o processamento de grades, multidimensionais arranjos e matrizes, juntamente com uma grande coleção de funções matemáticas de alto nível para operar sobre essas matrizes.

pip install -q -U numpy==1.18.4

TensorFlow

É uma biblioteca de código aberto criada para aprendizado de máquina, computação numérica e muitas outras tarefas.

pip install -q -U tensorflow==2.2.0

Keras

É uma biblioteca de rede neural de código aberto escrita em Python.

pip install -q -U keras==2.3.1

imutils

Inclui uma série de funções convenientes para tornar as funções básicas de processamento de imagem, como tradução, rotação, redimensionamento, esqueletização, exibição de imagens Matplotlib, classificação de contornos, detecção de arestas.

pip install -q -U imutils

Importando os pacotes

Os pacotes são uma maneira de estruturar o “espaço de nomes” dos módulos Python, usando “nomes de módulo com pontos”. Por exemplo, o nome do módulo A.B designa um submódulo chamado B , em um pacote chamado A. Para importar um módulo, utilizamos o import. Vejamos o código a ser implementado no notebook:

```
python

# Imports
import os
import sys
import cv2
import random
import tensorflow
import keras
import itertools
import matplotlib as m
import numpy as np
import matplotlib.pyplot as plt
from keras import backend as K
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Activation, Flatten, Dense
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical
from keras.optimizers import Adam
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
import imutils
from imutils import paths
import argparse
from keras.models import load_model
# Versões dos pacotes usados neste jupyter notebook
%reload_ext watermark
%watermark –iversions
```

Tratativa das imagens

Neste vídeo, vamos falar sobre a tratativa das imagens.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Carregando as imagens

Vamos agora carregar as imagens disponibilizadas neste material, a fim de que possamos utilizá-las no processamento de redes neurais.

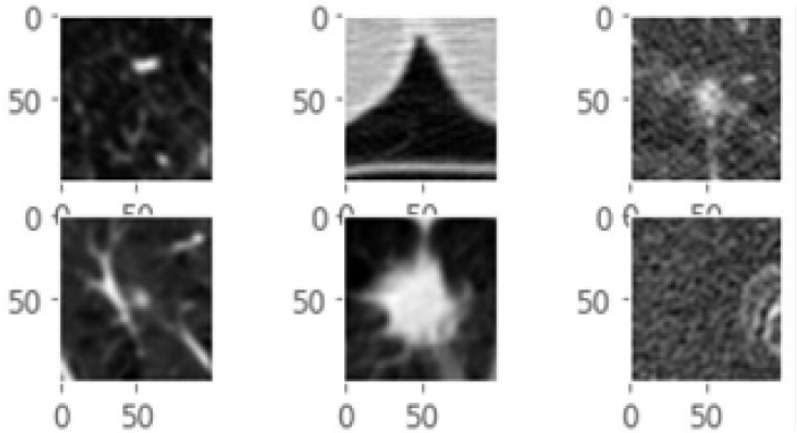
```python
# Diretório das imagens em disco
imagens_treino = "./dados/treino/"

# Listas para as imagens e labels
imagens = []
labels = []

# Lista as imagens e faz o shuffle (embaralha)
imagePaths = sorted(list(paths.list_images(imagens_treino)))
random.seed(42)
random.shuffle(imagePaths)

# Loop pelas imagens e leitura com OpenCV
for imagePath in imagePaths:

    # Leitura da imagem
    image = cv2.imread(imagePath)

    # Redimensionamento para 40x40 pixels
    image = cv2.resize(image, (40,40))

    # Converte a imagem para array
    image = img_to_array(image)

    # Adiciona à lista de imagens
    imagens.append(image)

    # Extrai o label
    label = imagePath[-7:-4]

    # Define o valor 0 ou 1 para o label
    if label == "pos":
        label = 1
    else:
        label = 0

    # Adiciona à lista de labels
    labels.append(label)

# Normalização das imagens
imagens = np.array(imagens, dtype = "float") / 255.0

# Carrega os labels
labels = np.array(labels)

# Labels
labels
```

Visualização das imagens

Depois de importar os dados, podemos visualizar as imagens. A seguir, mostramos o código:

```
python

# Plot das imagens
for i, images in enumerate(imagePaths[:6]):
    img = cv2.imread(images)
    img = cv2.resize(img, (100, 100))
    plt.subplot(3, 3, i + 1)
    plt.imshow(img)
    plt.grid(False)
plt.show()
```

Vejamos as imagens:



Print ilustrando as imagens

Pré-processamento das imagens

Nesse pré-processamento, vamos dividir as imagens em treino e teste, para treinar o modelo e depois testar sua performance.

```
python

# Divisão em treino/teste com proporção 75/25
(x_treino, x_teste, y_treino, y_teste) = train_test_split(imagens, labels, test_size =
0.25, random_state = 42)
# Shape
# Formato = número de imagens x altura x largura x número de canais de cores
x_treino.shape
# Shape
# Formato = número de imagens x altura x largura x número de canais de cores
x_teste.shape
```

Vamos deixar a variável de saída (label) como tipo categórico aplicando o One-Hot Encoding, que é uma transformação que fazemos nos dados para representarmos uma variável categórica de forma binária (indica presença ou ausência de um valor). Por exemplo, imagine que nosso conjunto de dados possui uma coluna chamada “sexo” com os valores M e F.

```
python

# One-Hot Encoding
y_treino = to_categorical(y_treino, num_classes = len(labels))
y_teste = to_categorical(y_teste, num_classes = len(labels))
```

Na sequência, vamos criar o gerador de imagens para o treinamento.

```
python

# Gerador de imagens
aug = ImageDataGenerator(rotation_range = 30,
            width_shift_range = 0.1,
            height_shift_range = 0.1,
            shear_range = 0.2,
            zoom_range = 0.2,
            horizontal_flip = True,
            fill_mode = "nearest")
```

Implementação do modelo

Neste vídeo, apresentaremos a implementação do modelo, incluindo a seleção do modelo, o treinamento do modelo, a avaliação do modelo, o relatório de classificação e o teste do modelo.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Seleção do modelo

Há uma infinidade de modelos de Machine Learning disponíveis, cada um voltado ao cumprimento de uma determinada função. Portanto, a escolha do modelo mais adequado deve ser feita de acordo com o objetivo proposto inicialmente. Nesse caso, foi escolhido uma rede CNN (Convolutional Neural Networks), pois é a mais adequada para classificação de imagens. Vejamos o código:

python

```
# Hiperparâmetros

# Número de épocas
epochs = 50

# Taxa de aprendizagem
lr = 1e-3

# Tamanho do batch
batch_size = 32

# Modelo CNN
class ModeloCNN:

    @staticmethod
    def build(width, height, depth, classes):

        # Cria a sequência de camadas
        model = Sequential()

        # Shape de entrada
        inputShape = (height, width, depth)

        # Formato das imagens
        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)

        # Primeira camada convolucional com ativação Relu e MaxPooling
        model.add(Conv2D(20, (5, 5), padding = "same", input_shape = inputShape))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

        # Segunda camada convolucional com ativação Relu e MaxPooling
        model.add(Conv2D(50, (5, 5), padding = "same"))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

        # Primeirca camada totalmente conectada com ativação Relu
        model.add(Flatten())
        model.add(Dense(500))
        model.add(Activation("relu"))

        # Camada de saída com classificação softmax
        model.add(Dense(classes))
        model.add(Activation("softmax"))

        return model

# Cria o modelo
modelo_cnn = ModeloCNN.build(width = 40, height = 40, depth = 3, classes = len(labels))

# Cria o otimizador com algoritmo Adam
otimizador = Adam(lr = lr, decay = lr / epochs)

# Compila o modelo com o otimizador, função de custo e métricas
modelo_cnn.compile(optimizer = otimizador, loss = "categorical_crossentropy", metrics =
["accuracy"])

# Sumário do modelo
modelo_cnn.summary()
```

Treinamento do modelo

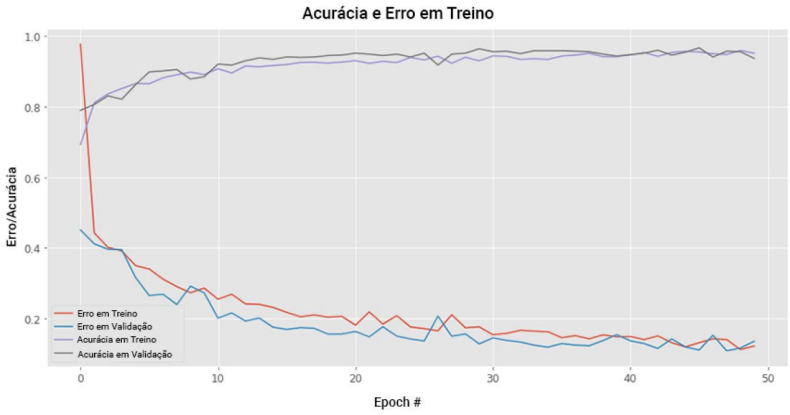
A etapa do treinamento é fundamental não apenas para preparar a máquina, mas para aprimorar constantemente suas habilidades de previsão. Dessa forma, a máquina efetivamente aprende com seus erros e torna-se cada vez mais aperfeiçoada. O treinamento pode ser considerado o principal pilar do Machine Learning.

```
python

# Treina o modelo

print("\nIniciando o Treinamento...\n")

hist = modelo_cnn.fit_generator(aug.flow(x_treino, y_treino, batch_size = batch_size),
                                validation_data = (x_teste, y_teste),
                                steps_per_epoch = len(x_treino) // batch_size,
                                epochs = epochs,
                                verbose = 1)

print("\nTreinamento Concluído.\n")
```

Avaliação do modelo

Uma maneira simples de observar o qução bom é um modelo de classificação é usando a acurácia. A acurácia indica uma performance geral do modelo. A acurácia mede o total de acertos considerando o total de observações. Vejamos o código correspondente:

```
python

# Calculamos a performance do modelo com dados de teste
score = modelo_cnn.evaluate(x_teste, y_teste, verbose = 1)
print("Acurácia do Modelo em Teste: %0.2f" % score[1])
```



Print ilustrando a acurácia do modelo.

Temos então o seguinte resultado para a acurácia:

Acurácia do Modelo em Teste: 0.94

Relatório de classificação

Vamos verificar a performance e o equilíbrio na previsão das classes do modelo.

```python
print(classification_report(predY, y_teste))
```

	precision	recall	f1-score	support
0.0	0.98	0.90	0.94	399
1.0	0.89	0.98	0.93	337
accuracy			0.94	736
macro avg	0.94	0.94	0.94	736
weighted avg	0.94	0.94	0.94	736

Print ilustrando o relatório de classificação.

Da imagem anterior, podemos concluir que o modelo está com excelente performance e equilibrado na previsão das classes.

Agora, vamos salvar o modelo em disco, pois usaremos para teste do modelo treinado e previsões. Vejamos o código:

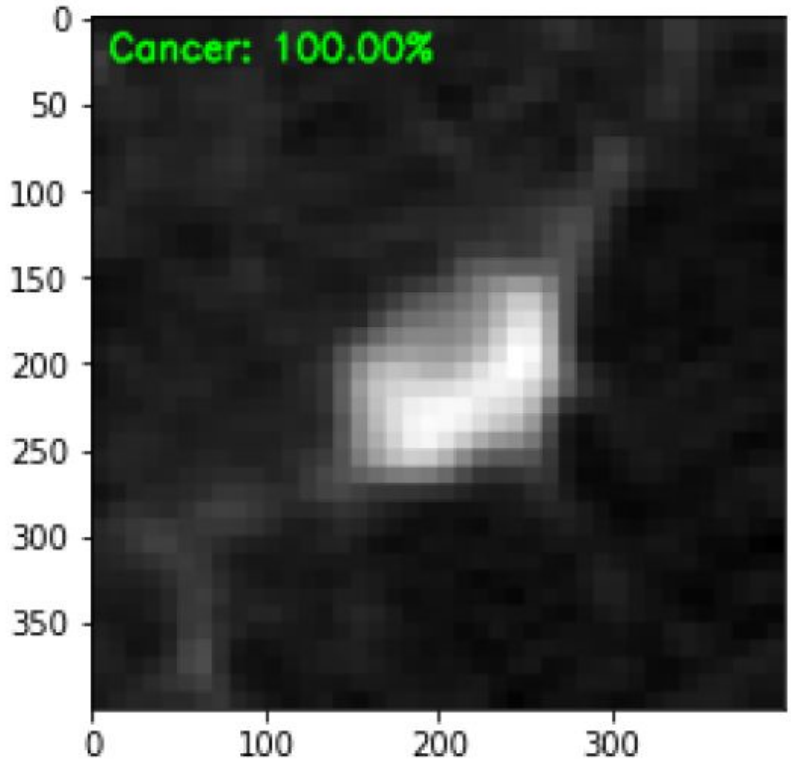
```python
modelName = "./modelo/modelo_cnn.model"
modelo_cnn.save(modelName)
```

Teste do modelo

Durante a fase do teste, o conjunto de teste é utilizado para determinar a performance da rede com dados que não foram previamente utilizados. A performance da rede, medida nessa fase, é uma boa indicação de sua performance real. Vejamos o código a seguir:

python

```
# Define o local da imagem (outras imagens estão disponíveis para #teste)
image = "./dados/teste/imagem2_cancer.png"

# Define o local do modelo
model = "./modelo/modelo_cnn.model"

#Carregamos e visualizamos uma imagem de teste.
# Carrega a imagem
image = cv2.imread(image)
orig = image.copy()
plt.imshow(image)
plt.show()

#Para cada nova imagem precisamos aplicar o mesmo pré-processamento #que foi usado no
treinamento.
# Pré-processamento da imagem
image = cv2.resize(image, (40, 40))
image = image.astype("float") / 255.0
image = img_to_array(image)
image = np.expand_dims(image, axis = 0)

# Carrega o modelo treinado e salvo anteriormente
model = load_model(model)

# Fazemos a previsão
score = model.predict(image)[0]

# Extraímos a classe da previsão
label = model.predict_classes(image)[0]

# De acordo com o label, definimos a classificação final
if label == 0:
    label = "No Cancer"
else:
    label = "Cancer"

#Calcular o nível de confiança do modelo
# Extrai a maior probabilidade das previsões do modelo
proba = max(score)

# Print
print("Nível de confiança = {0:.2f}%".format(proba*100))
# Output
output = imutils.resize(orig, width = 400)


# Verifica a probabilidade
if proba*100 > 95:
    out = "{}: {:.2f}%".format(label, proba * 100)
    color = (0,255,0)
else:
    out = "{}".format("Not detected")
    color = (255,0,0)

# Texto
text = cv2.putText(output, out, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX, 0.7, color, 2)

#Nível de confiança = 100.00%

#Por fim, criamos o plot mostrando a imagem e o label previsto pelo modelo.
plt.imshow(output)
plt.show()
```



Print ilustrando o resultado do teste do modelo.

A partir da imagem gerada, podemos concluir que o modelo detectou câncer.

Verificando o aprendizado

Questão 1

Deep Learning é um tipo de processo de Machine Learning, chamado aprendizado profundo, que usa nós ou neurônios interconectados em uma estrutura em camadas, semelhante ao cérebro humano. A rede neural cria um sistema adaptativo que os computadores usam para aprender com os erros e se aprimorar continuamente. Qual etapa da construção de um modelo de rede neural Deep Learning permite determinar a performance da rede com dados que não foram previamente utilizados?

A

Teste do modelo.

B

Avaliação do modelo.

C

Seleção do modelo.

D

Treinamento do modelo.

E

Relatório de classificação.

A alternativa A está correta.

No teste é utilizado um conjunto de teste no aprendizado de máquina, que é um conjunto de dados secundário, usado para testar um programa de aprendizado de máquina depois que ele foi treinado em um conjunto de dados de treinamento inicial.

Questão 2

O Deep Learning é a tecnologia base para ferramentas como o Google Translate (Google Tradutor) e a Cortana (assistente personalizado da Microsoft), por exemplo. Em suma, com enorme quantidade de poder computacional, as máquinas podem agora reconhecer objetos e traduzir voz em tempo real. Nesse contexto, analise as afirmativas a seguir.

I. Uma rede CNN (Convolutional Neural Networks) é a mais adequada para classificação de imagens.

II. A etapa do treinamento de um modelo de rede Deep Learning é fundamental não apenas para preparar a máquina, mas para aprimorar constantemente suas habilidades de previsão.

III. Conjunto de dados de treinamento é utilizado para determinar a performance da rede com dados que não foram previamente utilizados.

Assinale a seguir a alternativa cujas afirmativas relacionam-se com computação nas nuvens.

A

As opções I, II e III.

B

Apenas as opções II e III.

C

Apenas as opções I e II.

D

Apenas as opções I e III.

E

Apenas a opção III.

A alternativa C está correta.

O conjunto de teste é usado para possibilitar a determinação da performance da rede com dados que não foram utilizados previamente.

Considerações finais

Como vimos, Deep Learning é um subconjunto ou subárea do Machine Learning. Apresentamos os principais conceitos e características das redes de Deep Learning, assim como os fatores que as tornaram populares.

Vimos as principais redes de Deep Learning e suas respectivas características, tais como: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Autoencoders e as Generative Adversarial Networks (GANs).

As características dos principais frameworks de Deep Learning foram apresentadas. Os frameworks mais populares são: TensorFlow, Microsoft CNTK, Keras, PyTorch, Caffe, Theano e MXNet. Uma característica comum a esses frameworks é que são baseados em código livre. E, por fim, apresentamos um projeto passo a passo de Deep Learning de uma rede convolucional (CNN) para previsão de câncer de pulmão em imagens médicas.

Podcast

No podcast a seguir, faremos um breve resumo sobre o tema.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Complemente seus estudos pesquisando os seguintes assuntos:

- Criar modelos de Machine Learning no nível de produção com o TensorFlow.
- Convolutional Neural Networks, IBM Cloud Learn Hub.
- Recurrent Neural Networks, IBM Cloud Learn Hub.
- PyTorch – Da pesquisa à produção.

Referências

APACHE MXNet. Consultado na internet em: 09 nov. 2022.

CAFFE Deep Learning Framework. Consultado na internet em: 11 nov. 2022.

CHRISTOFIDIS, C. **Awesome Deep Learning**. Consultado na internet em 07 nov. 2022.

CHRISBASOGLU. **The Microsoft Cognitive Toolkit** – Cognitive Toolkit – CNTK. Microsoft. Consultado na internet em: 10 nov. 2022.

DEEP LEARNING BOOK. **Deep Learning Book**. Consultado na internet em: 11 nov. 2022.

KERAS. **Keras Documentation**. Consultado na internet em: 13 jan. 2023.

PYTORCH. **PyTorch**. Consultado na internet em: 09 nov. 2022.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. 4. ed. [S.l.] Prentice Hall, 2018.

THEANO. **Theano 1.1.2 documentation**. Consultado na internet em: 09 nov. 2022.

TRASK, A. W. **Grokking deep learning**. Shelter Island, NY: Manning, 2019.

TENSORFLOW. **TensorFlow**. Consultado na internet em: 13 jan. 2023.

WARDEN, P. **What is deep learning, and why should you care?** O'Reilly. Consultado na internet em: 07 nov. 2022.