

UNIVERSIDADE JOSÉ DO ROSÁRIO VELLANO - UNIFENAS
ALEXANDRE SOARES NUNES

REDE NEURAL CONVOLUCIONAL PROFUNDA

**Alfenas
2017**

ALEXANDRE SOARES NUNES

REDE NEURAL CONVOLUCIONAL PROFUNDA

Trabalho de Conclusão de Curso (TCC)
apresentado à Universidade José do Rosário Vellano, como parte das exigências
do Curso de Ciência da Computação para
obtenção de Título de Bacharel em Ciência
da Computação.

Orientador: Marcos Alberto de Carvalho

**Alfenas
2017**

Alexandre Soares Nunes

REDE NEURAL CONVOLUCIONAL PROFUNDA

Trabalho de Conclusão de Curso (TCC) apresentado à Universidade José do Rosário Vellano, como parte das exigências do Curso de Ciência da Computação para obtenção de Título de Bacharel em Ciência da Computação.

Apresentado em 11 de dezembro de 2017, Alfenas-MG

Marcos Alberto de Carvalho
Orientador

Flávia Aparecida Oliveira Santos
Examinadora

Jaqueleine Correa Silva de Carvalho
Examinadora

Dedico este trabalho primeiramente à Deus, por ser essencial em minha vida, meu guia, socorro presente nas horas de angústias. Ao meus pais, Antônio Carlos Nunes e Sebastiana de Fátima Soares, por serem meu chão, meu esteio e meu porto. Sempre estiveram lá quando eu precisei. Ao meu irmão Leandro, pois, sempre esteve junto e apoiando aos meus pais enquanto eu estive fora. A todos os meus professores, em especial, ao meu orientador e professor Marcos Alberto de Carvalho, por todos os ensinamentos que me passou e todas as orientações durante o projeto e, ao professor Francisco Amorim Pereira que me incentivou e orientou no processo inicial do meu intercâmbio.

Resumo

O presente trabalho analisou os desempenhos de uma rede neural convolucional profunda, conhecendo os obstáculos e limites deste novo conceito que é uma evolução das redes neurais tradicionais. As técnicas convencionais de treinamentos de redes neurais tiveram sua capacidade de processar dados naturais restrinidas, devido a enorme quantidade de informações geradas por segundo mundialmente. Por esse motivo e associado a evolução das placas gráficas, desenvolveu-se as redes neurais profundas para aprender novos padrões e se adaptar as mudanças atuais. Os treinamentos foram feitos em duas plataformas, sendo: uma com a utilização apenas do CPU e outra contou também com o poder computacional de uma GPU. Em todos os testes foram feitos utilizando os mesmos algoritmos em ambas as plataformas e bases de imagens conhecidas como MNIST e CIFAR 10. Quando utilizada a GPU para os aprendizados dos padrões das bases de imagens trabalhadas, observou-se que há uma melhora significativa no tempo dos aprendizados. Porém, só houve um aprimoramento na acurácia com a adição de novas camadas internas na rede.

Palavras-chave: GPU, Aprendizado Profundo, MNIST, Rede Neural Convolucional Profunda.

Abstract

In the present work, it was analized the performance of a deep convolutional neural network by knowing the obstacles and limits of this new concept that is an evolution of the neural networks. Conventional neural network training techniques have been limited in its capability to process natural data due to the huge amount of information generated per second globally. For this reason and associated with the evolution of graphics cards, it has been developed the deep neural networks to learn new patterns and adapt to current changes. The training was done in two platforms, being: the first one using CPU and other one using the computing power of a GPU. All tests was done using the same algorithms in both platforms and known image databases, such as: MNIST and CIFAR 10. When used the GPU to the learning the image's patterns, it has been observed that there is a significant improvement in time to learn. However, the accuracy only has been improved by adding new intern layers in the net.

Key-words: GPU, Deep Learning, MNIST, Deep Convolutional Neural Network.

Lista de ilustrações

| | |
|--|----|
| Figura 1 – Neurônios | 15 |
| Figura 2 – Perceptron | 17 |
| Figura 3 – ADALINE - Adaptive Linear Element | 18 |
| Figura 4 – Funções de Ativação | 19 |
| Figura 5 – Linha do tempo da Inteligência Artificial | 22 |
| Figura 6 – Exemplo de aplicação da convolução na imagem | 23 |
| Figura 7 – Rede Neural Convolutiva | 23 |
| Figura 8 – Ponte colorida | 25 |
| Figura 9 – Ponte em escala de cinza | 25 |
| Figura 10 – Parte da imagem — Ponte | 26 |
| Figura 11 – Parte da imagem em forma de função. | 26 |
| Figura 12 – Convolução | 27 |
| Figura 13 – Resultado parcial (ilustrativo) da convolução | 27 |
| Figura 14 – Função MaxPooling (ilustrativa) | 28 |
| Figura 15 – Unidade Central de Processamento. | 29 |
| Figura 16 – Comparativo entre as arquiteturas de uma CPU x uma GPU | 30 |
| Figura 17 – Comparativo de buscas sobre Tensorflow nos últimos cinco anos. | 32 |
| Figura 18 – Laptop — Meganote 4129 | 34 |
| Figura 19 – Computador | 35 |
| Figura 20 – Placa Gráfica — Galax OC GTX 1060 | 36 |
| Figura 21 – MNIST | 37 |
| Figura 22 – Exemplos da base CIFAR 10 | 38 |
| Figura 23 – Treinamento Cifar10 — Computador sem GPU | 40 |
| Figura 24 – Treinamento Cifar10 — Computador com GPU | 40 |
| Figura 25 – Função de Cross- Entropy no treinamento CIFAR 10. | 41 |
| Figura 26 – Perda no treinamento CIFAR 10. | 41 |
| Figura 27 – Teoria do Caos | 42 |
| Figura 28 – Verificando instalação/versão do Python | 48 |
| Figura 29 – Código MNIST — parte 1 | 52 |
| Figura 30 – Código MNIST — parte 2 | 53 |

Lista de tabelas

| | |
|--|----|
| Tabela 1 – Tempo médio gasto nos treinamentos. | 39 |
|--|----|

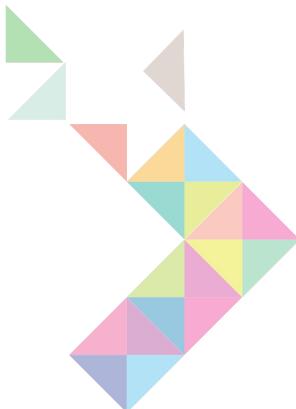
Lista de abreviaturas e siglas

| | |
|-------|--|
| ALU | Arithmetic Logic Unit |
| API | Application Programming Interface |
| CIFAR | Canadian Institute for Advanced Research |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| CU | Control Unit |
| CUDA | Compute Unified Device Architecture |
| DNA | Ácido Desoxirribonucleico |
| FPU | Float Point Unit |
| GPU | Graphics Processing Unit |
| IA | Inteligência Artificial |
| RGB | Red Green Blue |

Sumário

| | | |
|---------|---|----|
| 1 | INTRODUÇÃO | 11 |
| 1.1 | Contexto | 12 |
| 1.2 | Problema | 12 |
| 1.3 | Objetivo | 13 |
| 2 | REVISÃO LITERÁRIA | 14 |
| 2.1 | Inteligência Artificial | 14 |
| 2.1.1 | <i>Neurônio</i> | 15 |
| 2.1.2 | <i>Perceptron</i> | 16 |
| 2.1.3 | <i>O Elemento Linear Adaptativo (Adaline)</i> | 17 |
| 2.2 | Backpropagation | 18 |
| 2.3 | Aprendizado Profundo | 21 |
| 2.3.1 | <i>Rede Neural Convolutiva</i> | 22 |
| 2.3.1.1 | O aprendizado | 24 |
| 2.4 | Unidades de Processamento | 28 |
| 2.4.1 | <i>Unidade central de processamento (CPU)</i> | 28 |
| 2.4.2 | <i>Unidade de processamento gráfico (GPU)</i> | 30 |
| 2.5 | Python | 30 |
| 2.5.1 | <i>Tensorflow</i> | 31 |
| 3 | MATERIAIS E MÉTODOS | 33 |
| 3.1 | Equipamentos | 33 |
| 3.2 | Materiais | 36 |
| 3.2.1 | MNIST | 37 |
| 3.2.2 | CIFAR 10 | 37 |
| 3.2.3 | Curso de rede neural convolutiva | 38 |
| 3.3 | Metodologia | 38 |
| 4 | RESULTADOS E DISCUSSÕES | 39 |
| 4.1 | Resultados | 39 |
| 4.2 | Discussão | 41 |
| 5 | CONCLUSÃO | 43 |
| | Referências | 45 |

| | |
|---|-----------|
| APÊNDICES | 47 |
| APÊNDICE A – Instalação do Python | 48 |
| APÊNDICE B – Instalação do Tensorflow | 49 |
| B.1 Instalação da versão Tensorflow-CPU | 49 |
| B.2 Instalação da versão Tensorflow-GPU | 49 |
| B.3 Testando a instalação | 50 |
| APÊNDICE C – MNIST Tutorial | 52 |



FASTFORMAT

Você precisar comprar esse documento para remover a marca d'água.
Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.
10-page documents are free.

1 INTRODUÇÃO

A tecnologia de aprendizado de máquina (*machine learning*) ajuda muitos aspectos da sociedade moderna: desde pesquisas na *web* até filtragem de conteúdo em redes sociais para recomendações em *sites* de comércio eletrônico e cada vez mais presente em produtos de consumo como câmeras e smartphones. Os sistemas de aprendizado de máquina são usados para identificar objetos em imagens, transcrever discurso para texto, compartilhar itens de notícias, postagens ou produtos com os interesses dos usuários e selecionar os resultados relevantes da pesquisa. Cada vez mais, essas aplicações fazem uso de uma classe de técnicas chamadas aprendizagem profunda (*Deep Learning*).

As técnicas convencionais de aprendizagem de máquina foram limitadas em sua capacidade de processar dados naturais em sua forma bruta. Durante décadas, a construção de um sistema de reconhecimento de padrões ou de aprendizagem de máquina exigiu uma engenharia cuidadosa e conhecimentos de domínio consideráveis para projetar um extrator de características que transformasse os dados brutos (como os valores de *pixel* de uma imagem) em uma representação interna adequada ou em um vetor de características a partir do qual o subsistema de aprendizagem, muitas vezes um classificador, pudesse detectar ou classificar padrões na entrada.

O aprendizado por representação é um conjunto de métodos que permitem que uma máquina seja alimentada com dados brutos e, automaticamente, consegue descobrir as representações necessárias para detecção ou classificação. Os métodos de aprendizagem profunda são métodos de aprendizagem por representação com vários níveis, obtidos através da composição de módulos simples, mas não-lineares, que transforma a representação de nível (começando com a entrada bruta) em uma representação maior e ligeiramente em um nível mais alto de abstração. Com a composição de tais transformações, funções muito complexas podem ser aprendidas. Para tarefas de classificação, as camadas mais altas de representação ampliam os aspectos da entrada que são importantes para a discriminação e eliminam de variações irrelevantes. Uma imagem, por exemplo, vem na forma de uma série de valores de pixels e os recursos aprendidos na primeira camada de representação geralmente representam a presença ou ausência de arestas em orientações e locais particulares na imagem. A segunda camada tipicamente detecta padrões de arranjos particulares de arestas, independentemente de pequenas variações nas posições das arestas. A terceira camada pode reunir padrões de combinações maiores que correspondem a partes de objetos familiares e as camadas subsequentes detectariam objetos como combinações dessas partes. O aspecto fundamental do aprendizado profundo é que essas camadas de recursos não são projetadas por engenheiros humanos: são aprendidas a partir de

dados usando um procedimento de aprendizado de propósito geral.

O aprendizado profundo está fazendo avanços importantes na resolução de problemas que resistiram às melhores tentativas da comunidade de inteligência artificial (IA) por muitos anos. Tornou-se muito bom na descoberta de estruturas intrincadas em dados multidimensionais e, portanto, é aplicável a muitos domínios da ciência, das empresas e do governo. Além de bater recordes no reconhecimento de imagem e no reconhecimento de fala superou outras técnicas de aprendizagem de máquina para prever a atividade potencial de moléculas de drogas, analisando dados de aceleração de partículas, reconstruindo circuitos, e prevendo os efeitos de mutações em DNA não codificante em genes e doenças. Talvez, de forma mais surpreendente, o aprendizado profundo tenha produzido resultados extremamente promissores para várias tarefas na compreensão da linguagem natural, particularmente classificação tópica, análise de sentimentos, perguntas e respostas e tradução linguística. Acredita-se que o aprendizado profundo terá muitos mais êxitos no futuro próximo, porque requer muito pouca engenharia manual, para que possa-se aproveitar facilmente o aumento do poder computacional e dos dados disponíveis. Novos algoritmos de aprendizagem e arquiteturas que estão sendo desenvolvidas para redes neurais profundas, e acelerará esse progresso.

1.1 Contexto

O momento atual da tecnologia é a utilização da inteligência artificial para o aprimoramento e otimização do cotidiano, seja ele na agricultura, no comércio ou nas áreas da saúde. Seja em qualquer ambiente que a tecnologia esteja vinculada, a inteligência artificial está se mostrando uma ótima aliada. Empresas como: Google, Microsoft, Facebook e Tesla estão investindo milhões em estudos e pesquisa para alavancar ainda mais a evolução da IA. No início de 2016, o grupo de pesquisa do programa AlphaGo da Google venceu 10 jogos contra o campeão europeu Fan Hui gerando muitas expectativas para uso de sistemas inteligentes em um futuro próximo, uma vez que essa façanha estava sendo esperada pelo menos até 2020 (GIBNEY, 2016). Consegiu-se um avanço ~~no tempo de cinco anos~~. Aplicações como o Google Fotos e seus reconhecimentos faciais estão sendo possíveis graças apenas as melhorias nos algoritmos e modelos de treinamentos desenvolvidos nos anos 60 e 70.

1.2 Problema

Com a expansão da Internet, cresceu o número de usuário e suas aplicações aumentando o tráfego de informações, bem como, a necessidade de processamento de tais informações. Além do “boom” da produção de informações, o mercado de

equipamentos evolui consideravelmente. Independentemente destas transformações, os modelos de treinamentos padrões não estão trazendo um aprendizado eficiente, pois, os processadores não conseguem realizar suas tarefas de modo a oferecer um “custo-benefício” aceitável.

1.3 Objetivo

Buscando conhecer os limites e obstáculos desse novo conceito de aprendizagem de máquina e avaliar os recursos de *hardware* necessários para o aprendizado profundo, este trabalho foi desenvolvido com técnicas específicas para buscar os resultados mais verdadeiros possíveis.

Ainda que este conceito de aprendizagem de máquina não tenha sido desenvolvido recentemente, os estudos em redes neurais convolucionais profundas estão em alta nos últimos sete anos. Assim como toda tecnologia, as redes convolucionais também possuem suas limitações e dificuldades de desenvolvimento e implantações.

Além das dificuldades técnicas encontradas para implementação das redes convolucionais, há também limitação no poder de processamento dos equipamentos físicos. Houve um avanço no desenvolvimento de *hardwares* nas últimas décadas, contudo, existe uma enorme diferença entre os equipamentos disponíveis para as grandes empresas e para os usuários comuns.

Visando responder estas duas questões, este trabalho tem o objetivo de verificar qual o equipamento mínimo para um treinamento de uma rede neural convolucional contra balanceando custos e tempo para tal treinamento. Além de explicar para o leitor os conceitos básicos de forma simples, de tal maneira que ao final deste texto poderá entender o que é uma rede neural convolucional, bem como construir e treinar-la.

Você precisar comprar esse documento para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.

10-page documents are free.

2 REVISÃO LITERÁRIA

2.1 Inteligência Artificial

O termo inteligência artificial foi dito pela primeira vez em 1956 na conferência de Dartmouth, nos Estados Unidos da América, e organizado por John MacCarthy. Hoje conhecido mundialmente como “o pai da inteligência artificial”. Muitas pesquisas foram feitas na área, porém, dois trabalhos chamaram a atenção do meio científico, as pesquisas de Frank Rosenblatt em Perceptrons (ROSENBLATT, 1957) e os trabalhos publicados por Bernard Widrow e Ted Hoff conhecido como ADALINE (*Adaptive Linear Element*) (WIDROW; LEHR, 1990). Tanto o Perceptron, quanto ADALINE são conceitos de uma rede neural simples.

Uma rede de *feed-forward* de camada única consiste em um ou mais neurônios de saída, onde cada um dos quais está conectado com um fator de ponderação ou pesos a todas as entradas. No caso mais simples, a rede possui apenas duas entradas e uma única saída. A entrada do neurônio é a soma ponderada das entradas mais o termo das *bias*. A saída da rede é formada pela ativação do neurônio de saída, que é alguma função da entrada:

$$y = F \left(\sum_{i=1}^2 w_i x_i + \theta \right) \quad (2.1)$$

A função de ativação F pode ser linear, de modo que se possa ter uma rede linear ou não linear. Considera-se então a função de limite (*threshold function*): A saída da rede, portanto, é 1 ou -1 dependendo da entrada. A rede então pode ser usada para uma tarefa de classificação. Pode decidir se um padrão de entrada pertence a uma das duas classes. Se a entrada total for positiva, o padrão será atribuído à classe 1, se a entrada total for negativa, a amostra será atribuída à classe -1. A separação entre às duas classes neste caso é uma linha reta, dada pela equação:

10-page documents are free.

$$w_1 x_1 + w_2 x_2 + \theta = 0 \quad (2.2)$$

Há duas maneiras de se descrever os métodos de aprendizagem para esses tipos de redes: a regra de aprendizagem “Perceptron” e a regra “LMS” (*Least Mean Square* ou Mínimo Quadrado Médio). Ambos os métodos são procedimentos iterativos que ajustam os pesos. Uma amostra de aprendizagem é apresentada à rede. Para cada peso, o novo valor é calculado adicionando uma correção ao valor antigo.

2.1.1 Neurônio

Dentro de nossas cabeças, pesando cerca de 1,5 kg, o cérebro é um órgão vivo surpreendente composto por bilhões de pequenas células. Isso nos permite sentir o mundo ao nosso redor, pensar e conversar. O cérebro humano é o órgão mais complexo do corpo, e possivelmente o mais complexo na terra(SQUIRE et al., 2008a). Em 1873, o cientista e físico Camilo Golgi publicou os resultados de suas pesquisas expondo os primeiros neurônios. Porém, quatorze anos depois em 1887, o cientista e neuroanatomista Santiago Ramón y Cajal baseado nos trabalhos de Golgi e corrigindo algumas falhas, apresentou o que se tornou mundialmente conhecido como o primeiro neurônio (FIG.1) .

Figura 1 – Neurônios



<https://static01.nyt.com/images/2017/02/17/science/cajal-pyramidal-neuron/cajal-pyramidal-neuron-master315.jpg>

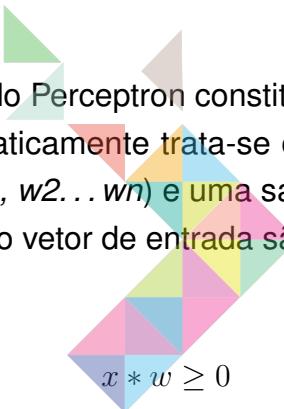
Os neurônios possuem uma arquitetura que consiste em um corpo celular e dois conjuntos de compartimentos adicionais chamados 'Processos'. Um desses conjuntos é chamado de axônios — seu trabalho é transmitir informações de um neurônio para outros, os quais estão conectados. O outro conjunto é chamado dendritos — seu trabalho é receber as informações que estão sendo transmitidas pelos axônios de outros

neurônios. Ambos os processos participam nos contatos especializados chamados sinapses(SQUIRE et al., 2008b) .

Há três categorias de neurônios: os neurônios sensoriais, os neurônios motores e os neurônios intermediários. Estes três tipos de neurônios são especializados e responsáveis na detecção e resposta de estímulos interno e esterno ao corpo humano. Quando há um excesso na intensidade de estímulos químicos, mecânicos, térmicos ou sonoro, podendo então causar danos aos tecidos do corpo humano, logo os neurônios sensoriais especializados enviam um sinal até os sensores motores, que por sua vez são responsáveis por controlar os músculos, realizando por sua vez ações necessárias para conter os danos. Entre os neurônios sensoriais e motores estão os sensores intermediários, que identificam qual o sinal e para quais neurônios os sinais devem ser entregues. Baseando-se neste conceito, os perceptrons foram criados.

2.1.2 *Perceptron*

Por convenção, o método Perceptron constitui-se de um conjunto de amostras de aprendizagem que matematicamente trata-se de um vetor de entrada x ($x_0, x_1, x_2 \dots x_n$), por pesos w ($w_0, w_1, w_2 \dots w_n$) e uma saída desejada $d(x)$ (ROSENBLATT, 1957). Os valores individuais do vetor de entrada são multiplicados pelos pesos,

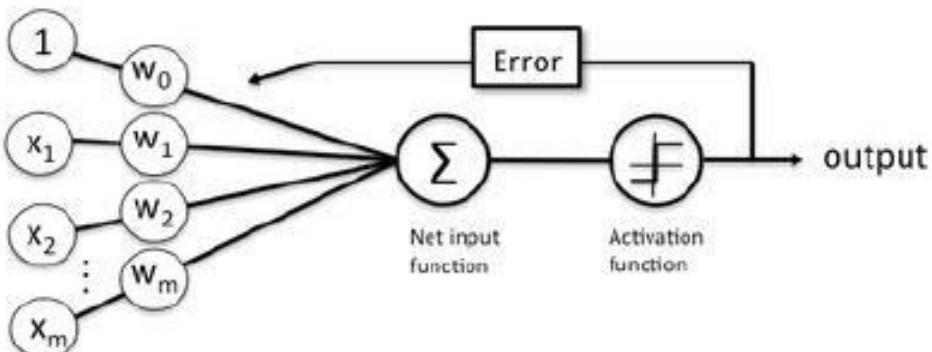


de forma que para uma tarefa de classificação, o $d(x)$ é geralmente 1 ou 0. A regra de aprendizagem do *perceptron* é muito simples e pode ser afirmada da seguinte maneira:

Você precisar comprar esse documento para remover a marca d'água.

- 1) Comece com pesos aleatórios para as conexões;
- 2) Selecione um vetor de entrada x do conjunto de amostras de treino;
You need to buy this document to remove the watermark.
- 3) Se $y \neq d(x)$ (o *perceptron* dá uma resposta incorreta), modifique todas as conexões w_i de acordo com: $\Delta w_i = d(x) x_i$;
- 4) Volte para o item 2.

Figura 2 – Perceptron



<https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html>

Um exemplo clássico é a função “E” que pode ser facilmente implementada e calculada com Perceptron. Inicialmente, a função “E” possui as seguintes entradas e suas respectivas saídas:

$$\begin{aligned}
 (X_1, X_2) &\rightarrow Y \\
 (0, 0) &\rightarrow 0 \\
 (0, 1) &\rightarrow 0 \\
 (1, 0) &\rightarrow 0 \\
 (-1, 1) &\rightarrow 1
 \end{aligned} \tag{2.4}$$

FASTFORMAT

Após a implementação, acrescentam-se os pesos junto as entradas, então ao calcular-se gera-se o resultado esperado. Os valores passam a ficar assim:

Você precisar comprar esse documento para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

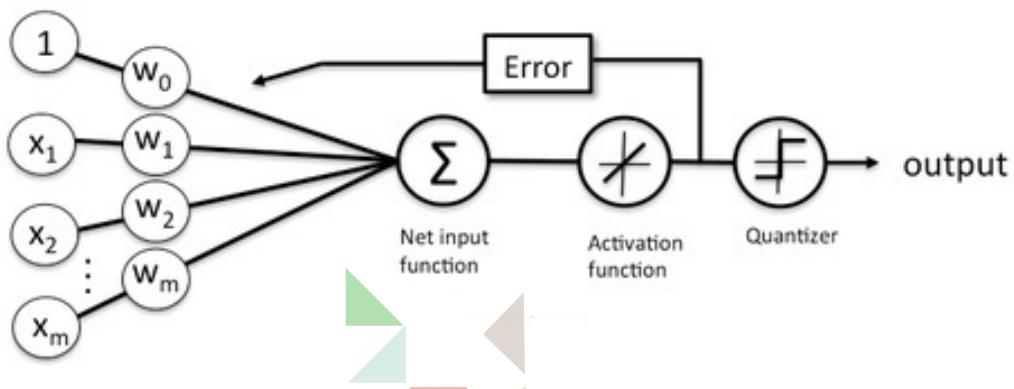
$$\begin{aligned}
 (0, 0, 1) &\rightarrow 0 \\
 \text{You need to buy this document to remove the watermark.} \\
 (0, 1, 1) &\rightarrow 0 \\
 \text{10-page documents are free.} \\
 (1, 0, 1) &\rightarrow 0 \\
 (-1, 1, 1) &\rightarrow 1
 \end{aligned} \tag{2.5}$$

2.1.3 O Elemento Linear Adaptativo (Adaline)

Uma importante generalização do algoritmo de treinamento do *perceptron* foi apresentada por Widrow e Hoff como o procedimento de aprendizado do mínimo quadrado médio (LMS), também conhecido como a regra delta. A principal diferença

funcional com a regra de treinamento de *perceptron* é a forma como a saída do sistema é usada na regra de aprendizagem. A regra de aprendizagem do *perceptron* usa a saída da função de limite (-1 ou 1) para aprender. A regra delta usa a saída líquida sem mapeamento adicional em valores de saída. A regra de aprendizado foi aplicada a ADALINE, desenvolvido por Bernard Widrow e Ted Hoff(WIDROW; LEHR, 1993) . Em uma implementação física simples:

Figura 3 – ADALINE



Essa implementação consiste em um conjunto de resistências controláveis conectadas a um circuito que pode resumir as correntes causadas pelos sinais de tensão de entrada.

2.2 Backpropagation

FASTFORMAT

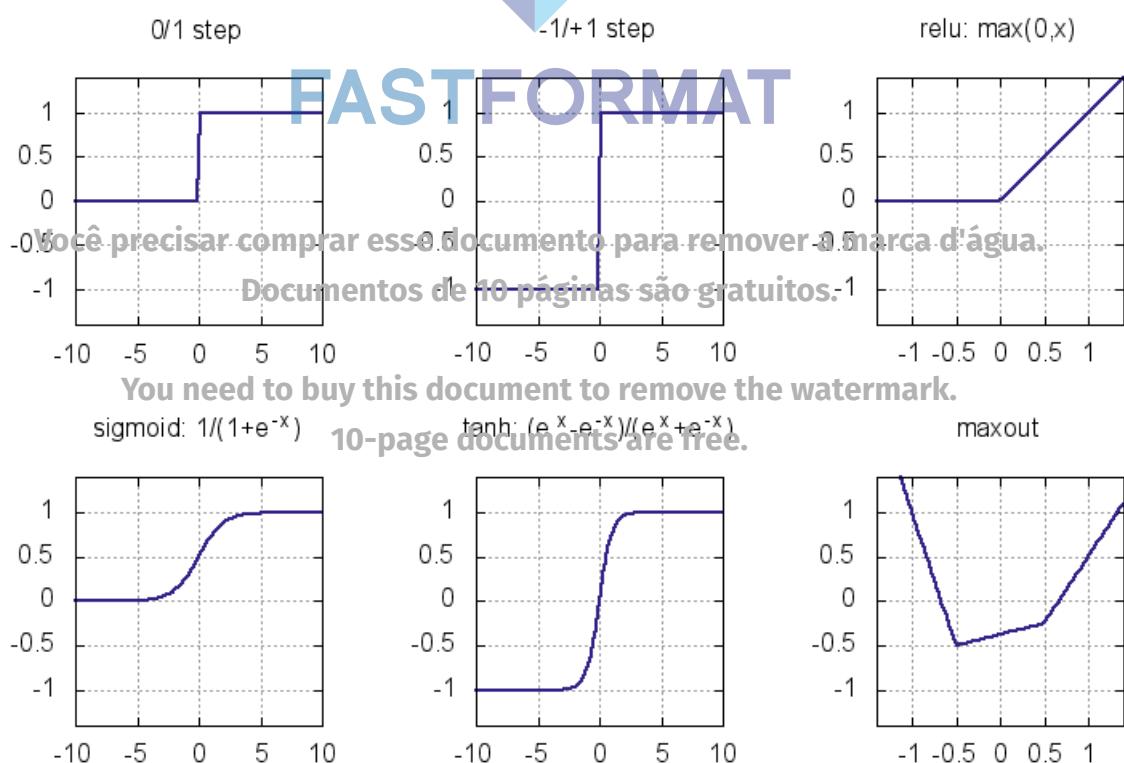
Desde o início do reconhecimento de padrões, o objetivo dos pesquisadores foi substituir características manipuladas manualmente em redes multicamadas treináveis, mas apesar de sua simplicidade, a solução não foi amplamente compreendida até meados dos anos 80. Como se pode verificar, as arquiteturas multicamadas podem ser treinadas por gradiente descendente estocástico simples. Enquanto os módulos são relativamente funções de suavização de suas entradas e de seus pesos internos, podem-se calcular os gradientes usando o procedimento de backpropagation. A ideia de que isso poderia ser feito, e que funcionou, foi descoberta de forma independente por vários grupos diferentes durante a década de 1970 e 1980(ROSENBLATT, 1957).

O procedimento de backpropagation para calcular o gradiente de uma função objetiva em relação aos pesos de uma pilha multicamada modular não é mais do que uma aplicação prática da regra de cadeia para derivadas. A visão chave é que a derivada (ou gradiente) do objetivo em relação à entrada de um módulo pode ser calculada trabalhando para trás a partir do gradiente em relação à saída desse módulo (ou a entrada do módulo subsequente). A equação de backpropagation pode ser

aplicada repetidamente para propagar gradientes através de todos os módulos, a partir da saída na parte superior (onde a rede produz sua previsão) até o final (onde a entrada externa é alimentada). Uma vez que esses gradientes foram computados, é enviado direto para calcular os gradientes em relação aos pesos de cada módulo.

De acordo com Werbos (1990), muitas aplicações de aprendizado profundo usam arquiteturas de rede neural feedforward, que aprendem um mapeamento de entrada de tamanho fixo (por exemplo, uma imagem) para uma saída de tamanho fixo (por exemplo, uma probabilidade para cada uma de várias categorias). Para ir de uma camada para a outra, um conjunto de unidades calcula uma soma ponderada de suas entradas da camada anterior e passa o resultado por uma função não linear. Atualmente, a função não linear mais popular é a unidade linear retificada (ReLU) FIG.4, que é simplesmente o retificador meia-onda. Nas décadas passadas, as redes neurais utilizaram não linearidades mais suaves, como $\tanh(z)$ ou $1 / (1 + \exp(-z))$, mas a ReLU geralmente aprende muito mais rapidamente em redes com muitas camadas, permitindo o treinamento de uma rede supervisionada profunda sem pré-treinamento não supervisionado. As unidades que não estão na camada de entrada ou saída são convencionalmente chamadas unidades ocultas. As camadas ocultas podem ser vistas como unidades de distorção da entrada de forma não linear, de modo que as categorias tornam-se linearmente separáveis pela última camada.

Figura 4 – Funções de Ativação



No final da década de 1990, as redes neurais e a backpropagation foram largamente abandonadas pela comunidade de aprendizagem mecânica e ignoradas pelas comunidades de reconhecimento de fala e de computação visual(KRIZHEVSKY; SUTSKEVER; HINTON, 2017). Era amplamente pensado que a aprendizagem útil, multiestágio, extractores de recursos com pouco conhecimento prévio era inviável. Comumente, pensava-se que o gradiente descendente simples ficaria preso em configurações locais de peso mínimo, para as quais nenhuma mudança pequena reduziria o erro médio.

Na prática, os mínimos locais raramente são um problema com grandes redes. Independentemente das condições iniciais, o sistema quase sempre atinge soluções de qualidade muito similar. Resultados teóricos e empíricos recentes sugerem fortemente que os mínimos locais não são um problema sério em geral. Em vez disso, a paisagem é empacotada com um número combinatório de pontos de sela onde o gradiente é zero e a superfície se curva para cima na maioria das dimensões e curva-se para baixo no restante. A análise parece mostrar que os pontos de sela com apenas algumas orientações de curvatura para baixo estão presentes em números muito grandes, mas quase todos têm valores muito similares da função objetiva. Por isso, não importa muito qual desses pontos de sela o algoritmo fica preso(WERBOS, 1990).

De acordo com Ranzato et al. (2007), o interesse em redes de feedforward profundo foi revivido em torno de 2006, por um grupo de pesquisadores reunidos pelo Instituto Canadense de Pesquisa Avançada (CIFAR). Os pesquisadores apresentaram procedimentos de aprendizagem sem supervisão que poderiam criar camadas de detectores de recursos sem exigir dados rotulados. O objetivo em aprender cada camada de detectores de recursos foi para poder reconstruir ou modelar as atividades dos detectores de recursos (ou entradas brutas) na camada abaixo. Por “pré-treinamento” várias camadas de detectores de recursos progressivamente mais complexos usando este objetivo de reconstrução, os pesos de uma rede profunda poderiam ser inicializados em valores sensíveis. Uma camada final de unidades de saída poderia então ser adicionada ao topo da rede e todo o sistema profundo poderia ser ajustado usando o padrão backpropagation. Isso funcionou notavelmente bem para reconhecer dígitos manuscritos ou para detectar pedestres, especialmente quando a quantidade de dados rotulados era muito limitada.

Segundo Raina, Madhavan e Ng (2009), a primeira grande aplicação desta abordagem de pré-treinamento foi no reconhecimento de fala e foi possível graças ao advento das unidades de processamento de gráficos (GPUs) rápidas que eram convenientes para o programa e permitiram que os pesquisadores treinassem redes 10 ou 20 vezes mais rápido. Em 2009, a abordagem foi utilizada para mapear janelas temporais curtas de coeficientes extraídos de uma onda sonora para um conjunto de

probabilidades para os vários fragmentos de fala que podem ser representados pelo quadro no centro da janela. Ele alcançou resultados recordes em um benchmark de reconhecimento de fala padrão que usava um pequeno vocabulário e foi rapidamente desenvolvido para dar resultados recorde em uma grande tarefa de vocabulário. Até 2012, as versões das redes profundas desenvolvidas a partir de 2009 estavam sendo desenvolvidas por muitos dos principais grupos de fala e já estavam sendo implantadas em telefones Android. Para conjuntos de dados menores, o pré-treinamento não supervisionado ajuda a evitar o super-ajuste, levando a uma generalização significativamente melhor quando o número de exemplos rotulados é pequeno ou em uma configuração de transferência onde temos muitos exemplos para algumas tarefas de "fonte", mas muito poucas para algumas tarefas de "alvo". Uma vez que a aprendizagem profunda foi reabilitada, descobriu-se que a fase de pré-treinamento só era necessária para pequenos conjuntos de dados.

Havia, no entanto, um tipo particular de rede profunda, feedforward que era muito mais fácil de treinar e generalizar muito melhor do que redes com conectividade completa entre camadas adjacentes. Esta foi a rede neural convolutiva. Alcançou muitos sucessos práticos durante o período em que as redes neurais não eram as favoritas e recentemente foi amplamente adotada pela comunidade de computação visual(LECUN et al., 1990).

2.3 Aprendizado Profundo

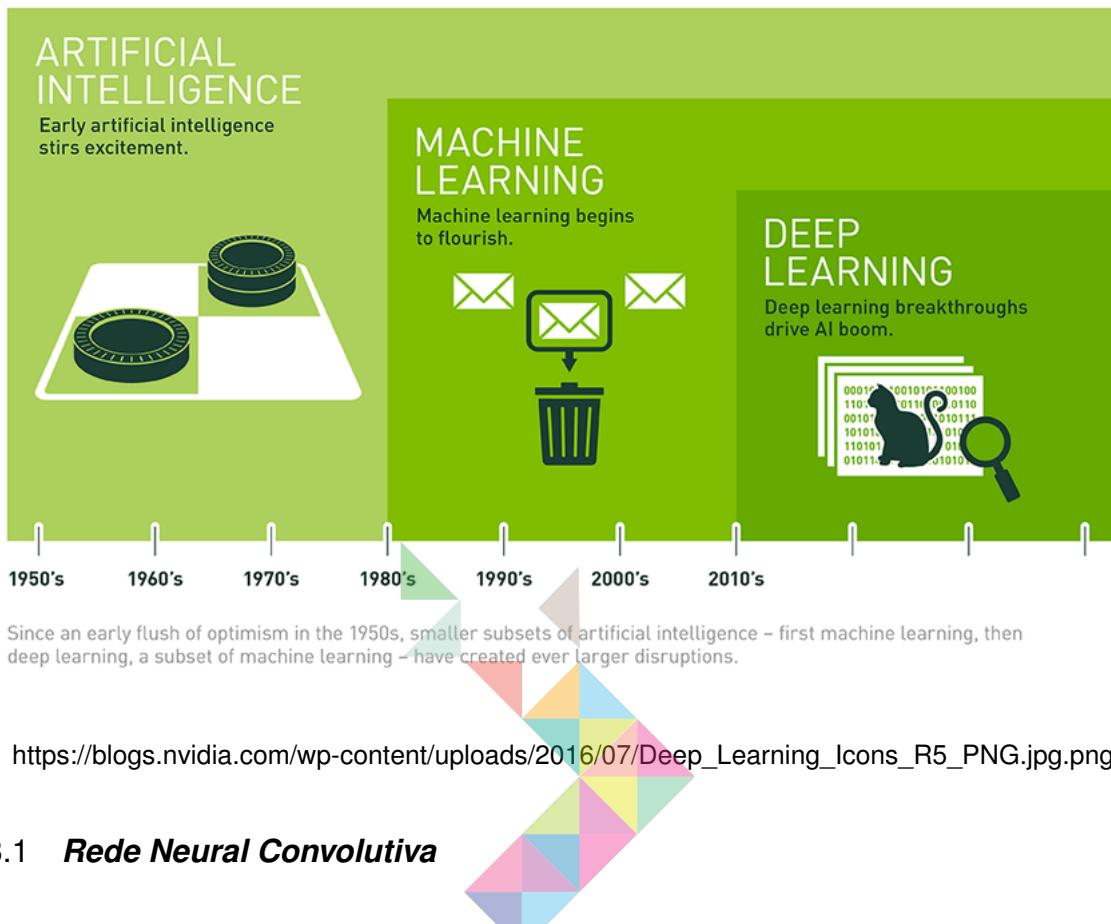
Deng e Yu (2014) descrevem que o aprendizado profundo ("Deep Learning") como uma classe de técnicas de aprendizagem de máquina, onde muitas camadas de estágios de processamento de informações em arquiteturas supervisionadas hierárquicas são exploradas para o aprendizado de recursos não supervisionados e para análise / classificação de padrões. A essência do aprendizado profundo é calcular características hierárquicas ou representações dos dados observacionais, onde as características ou fatores de nível superior são definidos a partir de níveis mais baixos. A família dos métodos de aprendizagem profunda tem crescido cada vez mais rica, abrangendo as de redes neurais, modelos probabilísticos hierárquicos e uma variedade de algoritmos de aprendizagem de recursos supervisionados e não supervisionados.

You need to buy this document to remove the watermark.

10-page documents are free.

Seguindo os pensamentos de Turing (1950), descritos em seu artigo, a partir da década de 50 os pesquisadores se empenham em pesquisas no âmbito do aprendizado de máquina . A figura 5 ilustra como foi um pouco a evolução da inteligência artificial ao passar dos anos chegando até o novo conceito de aprendizado profundo.

Figura 5 – Linha do tempo da Inteligência Artificial

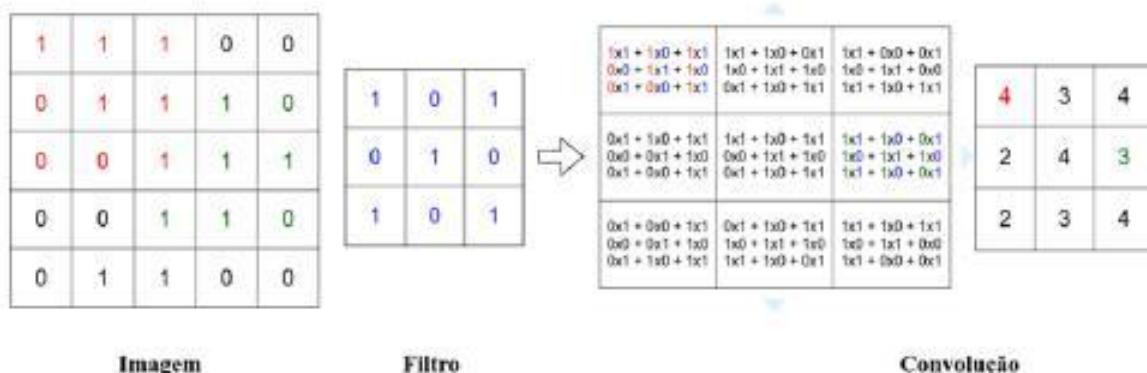


2.3.1 Rede Neural Convolutiva

As redes neurais convolutivas são projetadas para processar dados que vêm na forma de vários vetores, por exemplo, uma imagem colorida composta por três matrizes 2D contendo intensidade de pixels nos três canais de cores, sendo o padrão vermelho-verde-azul (*RGB*). Muitas modalidades de dados estão na forma de matrizes múltiplas: 1D para sinais e sequências, incluindo linguagem; 2D para imagens ou espectrogramas de áudio; e 3D para imagens em vídeo ou volumétricas. Existem quatro ideias chave para o ConvNets que aproveitam as propriedades dos sinais naturais: conexões locais, pesos compartilhados, agrupamento e uso de muitas camadas.

Segundo Ferreira (2017) uma rede neural convolutiva interpreta uma imagem como o somatório da multiplicação dos elementos da imagem. Pois trata-se de uma matriz de informações. Ao multiplicar-se estes valores próximos da matriz pelo filtro, gera-se então uma terceira matriz com o resultado da convolução. A figura (FIG.6) abaixo ilustra este processo de multiplicação:

Figura 6 – Exemplo de aplicação da convolução na imagem.

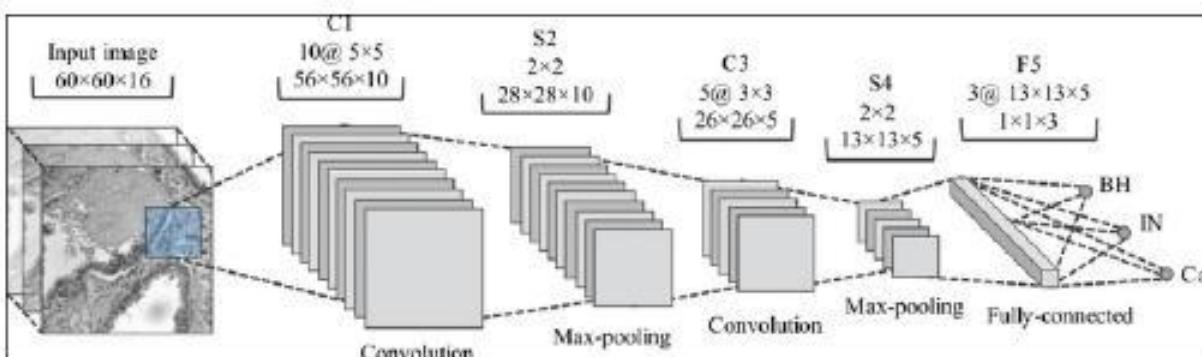


<http://www.gpec.ucdb.br/pistori/orientacoes/dissertacoes/alessandro2017.pdf>

A arquitetura de um ConvNet (FIG.7) típico é estruturada como uma série de etapas. Os primeiros estágios são compostos por dois tipos de camadas: camadas convolutivas e camadas de agrupamento. As unidades em uma camada convolutivas são organizadas em mapas de recursos, dentro dos quais cada unidade está conectada a manchas locais nos mapas de recursos da camada anterior através de um conjunto de pesos chamados banco de filtro. O resultado dessa soma ponderada local passa então por uma não linearidade, como uma ReLU (KRIZHEVSKY; SUTSKEVER; HINTON, 2017). Todas as unidades em um mapa de recursos compartilham o mesmo banco de filtros. Os diferentes mapas de recursos em uma camada usam diferentes bancos de filtros. A razão para esta arquitetura é dupla. Primeiro, em dados de matriz, como imagens, grupos locais de valores são muitas vezes altamente correlacionados, formando motivos locais distintos que são facilmente detectados. Em segundo lugar, as estatísticas locais de imagens e outros sinais são invariantes para a localização. Em outras palavras, se um motivo pode aparecer em uma parte da imagem, pode aparecer em qualquer lugar, daí a ideia de unidades em diferentes locais compartilhando os mesmos pesos e detectando o mesmo padrão em diferentes partes da matriz. Matematicamente, a operação de filtragem realizada por um mapa de recursos é uma convolução discreta, daí o nome.

10-page documents are free.

Figura 7 – Rede Neural Convolutiva



http://www.jpathinformatics.org/articles/2017/8/1/images/JPatholInform_2017_8_1_1_201108_f3.jpg

Embora o papel da camada convolucional seja para detectar conjunções locais de recursos da camada anterior, o papel da camada de agrupamento (pooling) é fundir recursos semanticamente semelhantes em um. Como as posições relativas das características que formam um motivo podem variar um pouco, a detecção confiável do motivo pode ser feita através da granulação da posição de cada característica. Uma unidade de agrupamento típica calcula o máximo de um trecho local de unidades em um mapa de recursos (ou em alguns mapas de recursos) (Krizhevsky; Sutskever; Hinton, 2017). As unidades de agrupamento vizinhas recebem entrada destes trechos que são deslocados por mais de uma linha ou coluna, reduzindo assim a dimensão da representação e criando uma invariância para pequenas mudanças e distorções. São empilhados dois ou três estágios de convolução, não linearidade e agrupamento, seguidos de camadas mais convolucionais e completamente conectadas. Os gradientes de Backpropagation através de um ConvNet são tão simples quanto através de uma rede regular e profunda, permitindo que todos os pesos em todos os bancos de filtros sejam treinados.

As redes neurais profundas exploram a propriedade de que muitos sinais naturais são hierarquias de composição, nas quais os recursos de nível superior são obtidos através da composição de níveis mais baixos. Nas imagens, as combinações locais de bordas formam motivos, os motivos se reúnem em partes e as peças formam objetos. Hierarquias semelhantes existem na fala e no texto de sons para telefones, fonemas, sílabas, palavras e frases. O agrupamento permite que as representações variem muito pouco quando os elementos na camada anterior variam em posição e aparência.

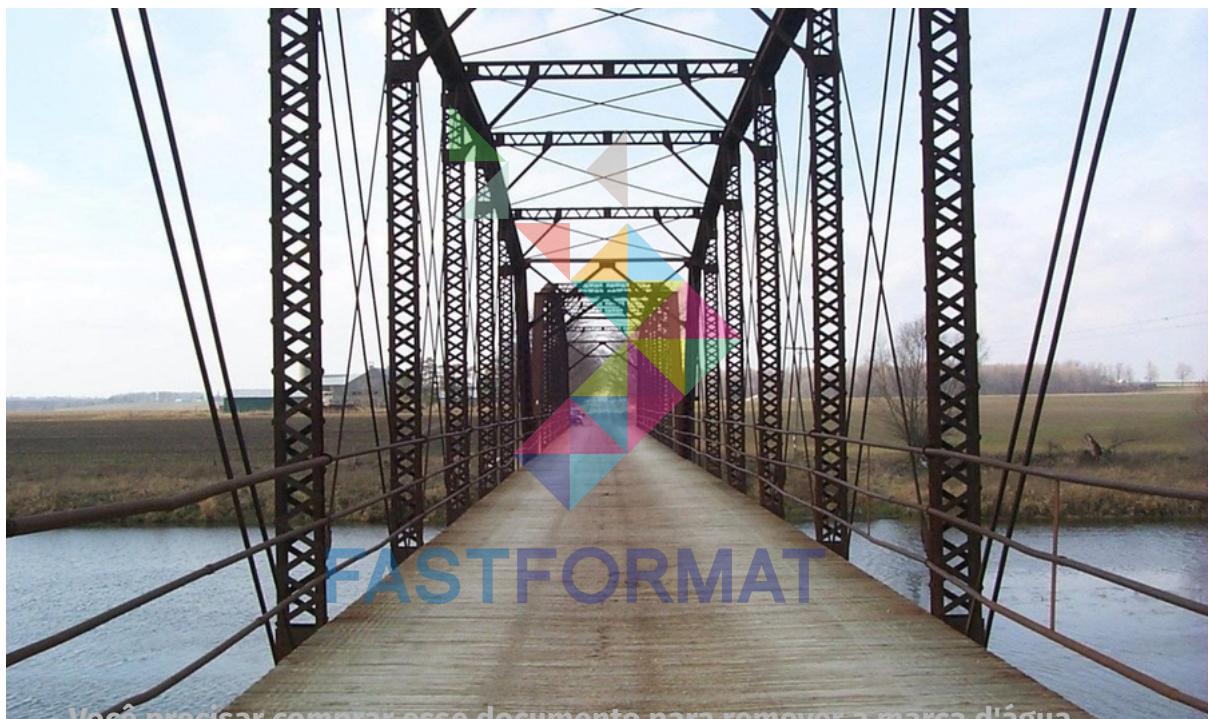
Houve inúmeras aplicações de redes convolutivas voltando ao início da década de 1990, começando com redes neurais de tempo para reconhecimento de fala e leitura de documentos. O sistema de leitura de documentos utilizou uma ConvNet treinada em conjunto com um modelo probabilístico que implementou restrições linguísticas. No final dos anos 90, este sistema estava lendo mais de 10% de todas as verificações nos Estados Unidos. Uma série de sistemas de reconhecimento de caracteres ópticos e de

reconhecimento de caracteres baseados em ConvNet foram posteriormente implantados pela Microsoft. Os ConvNets também foram experimentados no início da década de 1990 para a detecção de objetos em imagens naturais, incluindo rostos e mãos, e para reconhecimento facial.

2.3.1.1 O aprendizado

A rede convolutiva processa cada *pixel* ou conjunto deles aplicando os filtros conforme projetados. Para efeito de exemplificação, será aplicado um filtro de escala de cinza na imagem abaixo:

Figura 8 – Ponte colorida



You need to buy this document to remove the watermark.

Resultando na seguinte imagem:

10-page documents are free.

Figura 9 – Ponte em escala de cinza

Ao analisar mais profundamente esta imagem consegue-se verificar que se trata apenas de uma função matemática (FIG. 11), como pode-se observar ao isolar uma parte da imagem (FIG. 10).

Figura 10 – Parte da imagem — Ponte

OBS.: Imagem aumentada para melhor visualização.

Própria (2017)

Você precisar comprar esse documento para remover a marca d'água.

Digumento Parte 10 da imagem em cognição.



Própria (2017)

Ao realizar a convolução, a rede reconhece padrões na imagem conforme os filtros aplicados. Esse reconhecimento se dá pela operação matemática entre duas funções f e g , produzindo uma terceira função, que pode ser interpretada como uma função modificada de f .

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) * g(x - \tau)d\tau \quad (2.6)$$

No processamento de imagem, onde a imagem é definida como uma função bidimensional, a convolução é útil para detecção de bordas, alisamento de imagem, extração de atributos, entre outras aplicações. Uma outra maneira de entender a convolução é onde cada valor do pixel da imagem é multiplicado por um filtro e somado os resultados das multiplicações. Essa operação evidencia os pontos onde a parte da imagem corresponde ao filtro aplicado, determinando então os pontos característicos da imagem. Abaixo a imagem representando a convolução:

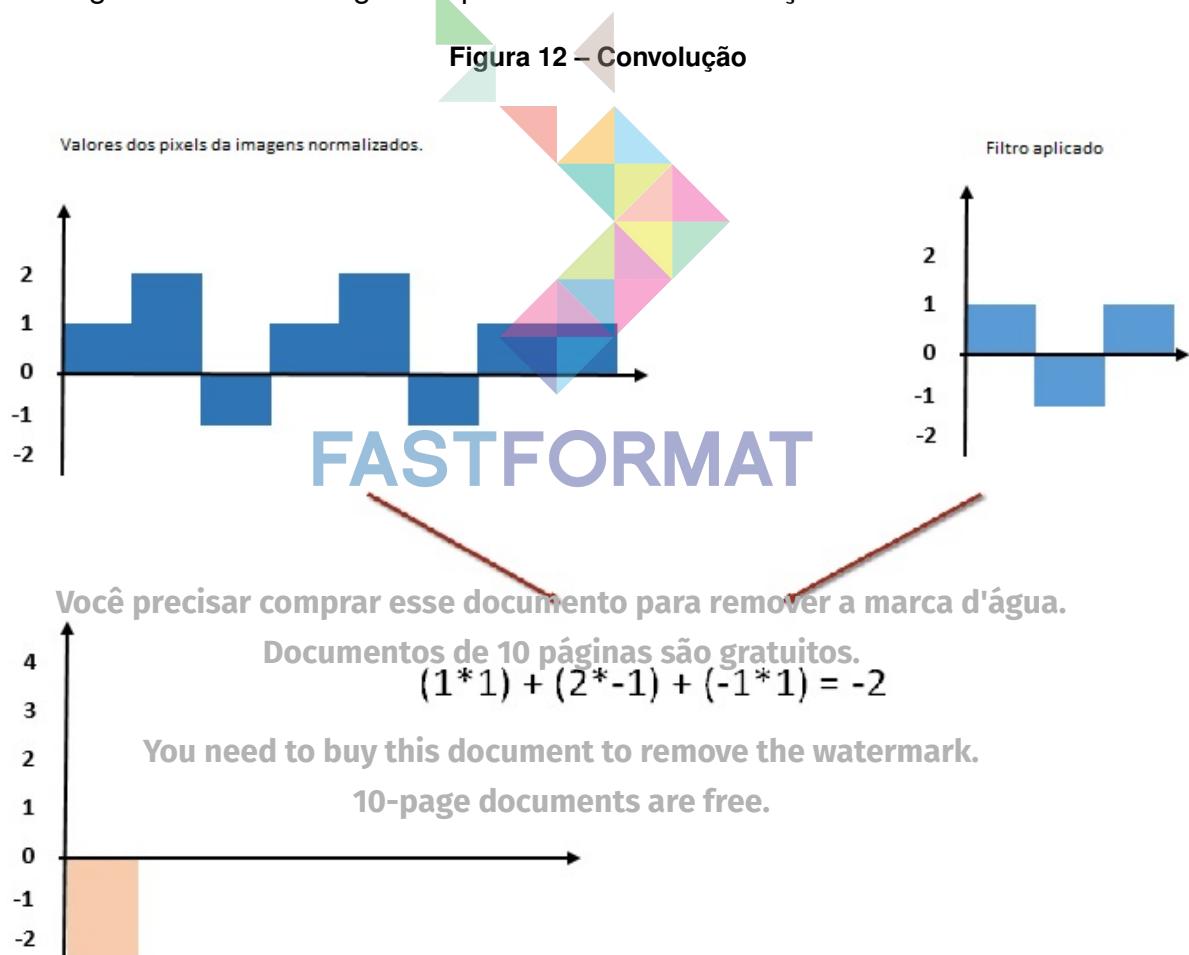
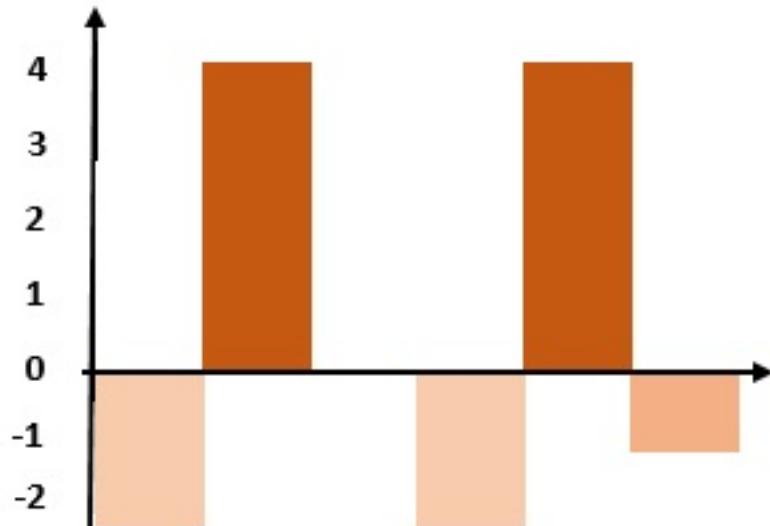


Figura 13 – Resultado parcial (ilustrativo) da convolução

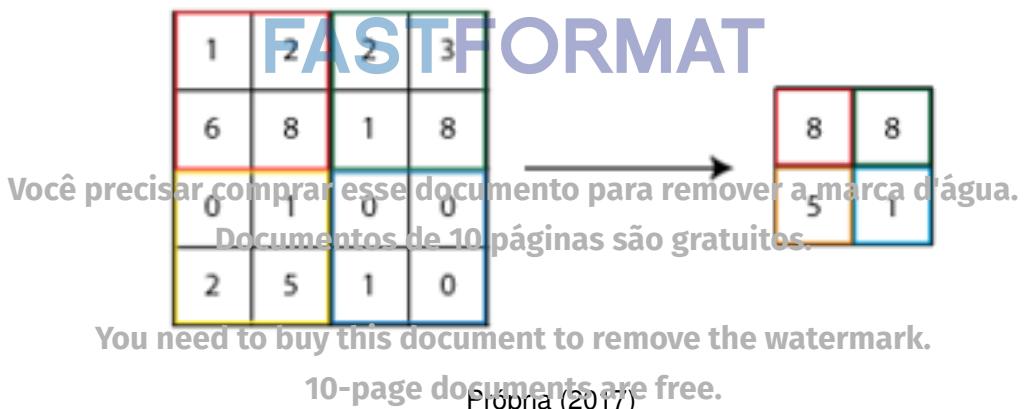


Própria (2017)

Após a convolução, ocorre a função de *MaxPooling* (Maximização).

A função de *MaxPooling* potencializa os resultados calculados na convolução:

Figura 14 – Função MaxPooling (ilustrativa)

MaxPooling

O aprendizado destes filtros é realizado na maioria das redes por meio do método gradiente de descendente. Onde os valores iniciais dos pesos são gerados randomicamente e após finalizar cada passagem pelas camadas, são calculados os erros através da função Softmax. A função Softmax é aplicada para interpretar as saídas como probabilidades e através de backpropagation os pesos dos filtros são levemente alterados para ajustar a rede de tal maneira que ela aprenda os padrões apresentados.

2.4 Unidades de Processamento

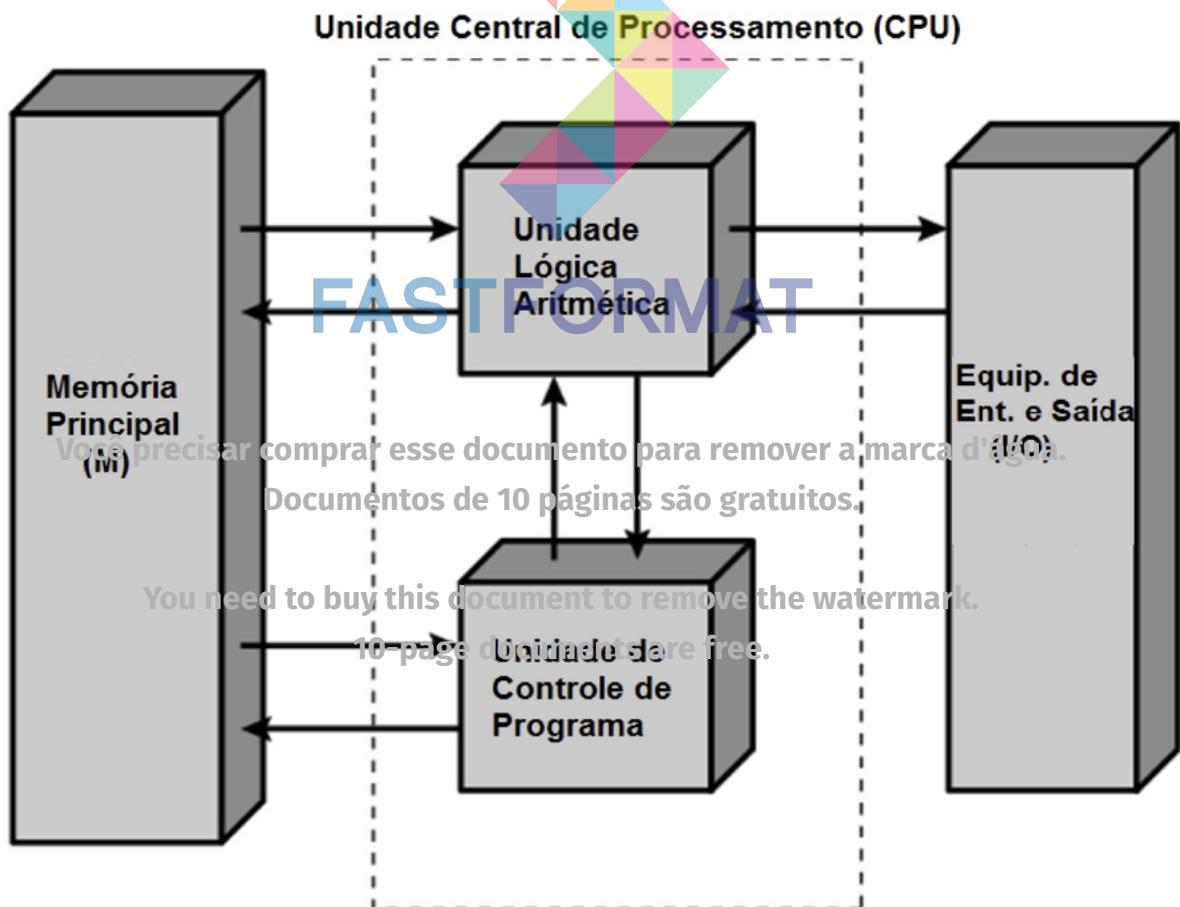
2.4.1 Unidade central de processamento (CPU)

O CPU é a abreviatura da unidade de processamento central. Às vezes referido simplesmente como o processador central ou mais comumente chamado de processador. O CPU é o cérebro do computador onde a maioria dos cálculos ocorre. Para a potência de computação, o CPU é o elemento mais importante de um sistema.

Existem basicamente dois componentes em um CPU, sendo:

- A unidade de lógica aritmética (ALU), que executa operações aritméticas e lógicas.
- A unidade de controle (CU), que extrai instruções da memória e decodifica e executa-as, ligando a ALU quando necessário.

Figura 15 – Unidade Central de Processamento.



A CPU executa o sistema operacional e as aplicações, recebendo constantemente a entrada do usuário ou programas de software ativos. Ele processa os dados e produz saída, que pode ser armazenada por um aplicativo ou exibida na tela.

A CPU contém pelo menos um processador, que é o chip real dentro da CPU que executa cálculos. Durante muitos anos, a maioria das CPUs tinha apenas um processador, mas agora é comum que uma única CPU possua pelo menos dois processadores ou “núcleos de processamento”. Uma CPU com dois núcleos de processamento é chamada de CPU dual-core e os modelos com quatro núcleos são chamados de CPUs quad-core. As CPUs de gama alta podem ter seis processadores (hexa-core) ou mesmo oito (octa-core). Um computador também pode ter mais de uma CPU, cada qual possui vários núcleos. Por exemplo, um servidor com duas CPUs hexa-core possui um total de 12 processadores(ARRUDA, 2011) .

Enquanto as arquiteturas do processador diferem entre os modelos, cada processador dentro de uma CPU normalmente possui seu próprio ALU, FPU,CU, registro e cache L1. Em alguns casos, os núcleos de processamento individuais podem ter seu próprio cache L2, embora também possam compartilhar o mesmo cache L2.

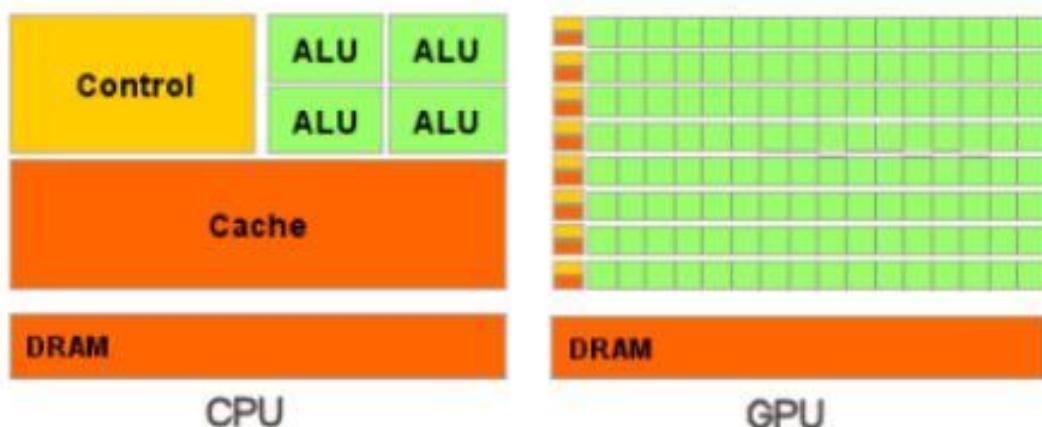
2.4.2 *Unidade de processamento gráfico (GPU)*

Unidades de processamento de gráficos (GPUs) são coprocessadores que tradicionalmente executam a renderização de informações de gráficos bidimensionais e tridimensionais para exibição em uma tela. Em particular, os jogos de computador exigem renderização em tempo real cada vez mais realista de dados gráficos e, portanto, as GPUs tornaram-se cada vez mais poderosas unidades de computação especializada altamente paralelas. Não demorou muito até que os programadores percebessem que esse poder computacional também podem ser usados para tarefas que não seja computação gráfica e treinamento de redes neurais.

Documentos de 10 páginas são gratuitos.

A capacidade de calcular pequenas tarefas em paralelo em suas centenas e/ou até milhares de (FIG.16) núcleos , trouxe para o aprendizado de máquina um avanço enorme. Especificamente para as redes convolutivas, onde os cálculos e aplicações dos filtros são basicamente os mesmos durante todo o processo de treinamento.

Figura 16 – Comparativo entre as arquiteturas de uma CPU x uma GPU



http://www.ixbt.com/video3/images/cuda/cpu_vs_gpu.png

O paradigma de programação mudou quando os dois principais fabricantes de GPU, NVIDIA e AMD, mudaram a arquitetura de hardware de uma tubulação de renderização gráfica dedicada para uma plataforma de computação multi-core, implementaram algoritmos de sombreado da transformação em um software executado nesses núcleos e explicitamente suportou cálculos de uso geral em GPUs, oferecendo linguagens de programação e cadeias de ferramentas de desenvolvimento de software.

2.5 Python

O Python é uma linguagem de programação de alto nível, projetada para ser fácil de ler e simples de implementar. É de código aberto, o que significa que é de uso gratuito, mesmo para aplicações comerciais. O Python pode ser executado em sistemas Mac, Windows e Unix e também foi portado para máquinas virtuais Java e .NET (PYTHON).

Você precisará comprar esse documento para remover a marca d'água.

Python é considerado uma linguagem de script, como Ruby ou Perl e é frequentemente usado para criar aplicativos da Web e conteúdo da Web dinâmico. Também é suportado por uma série de programas (geradores de imagens, 2D e 3D), permitindo que os usuários criem plug-ins personalizados e extensões com o Python. Exemplos de aplicativos que suportam uma API Python incluem GIMP, Inkscape, Blender e Autodesk Maya.

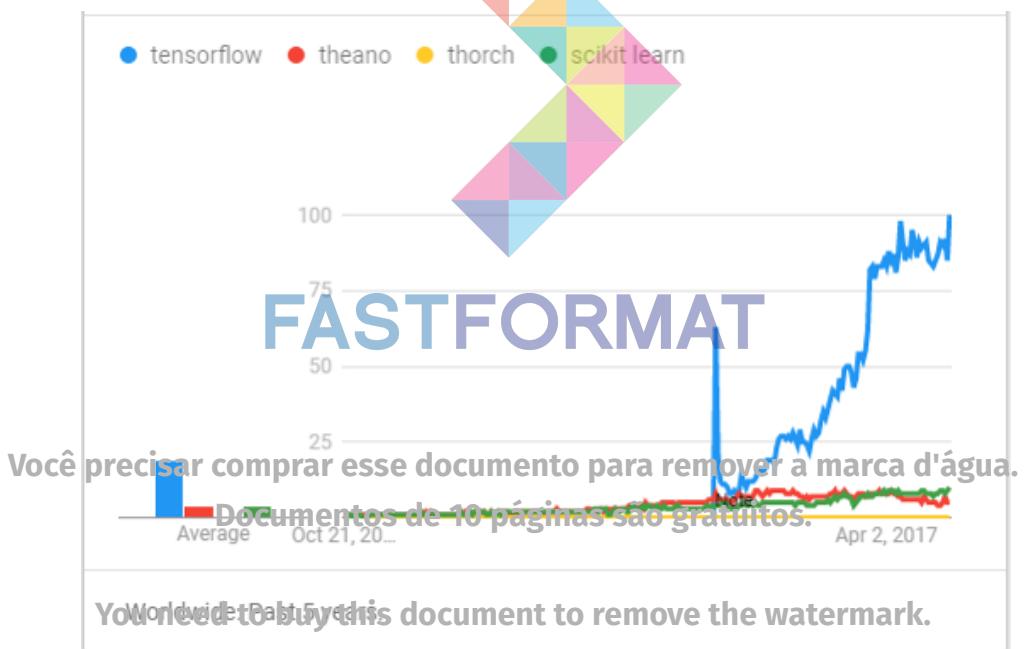
Os scripts escritos em Python (arquivos .PY) podem ser analisados e executados imediatamente. Eles também podem ser salvos como programas compilados (arquivos .PYC), que geralmente são usados como módulos de programação que podem ser referenciados por outros programas Python.

2.5.1 Tensorflow

Tensorflow é uma biblioteca de *software* de código aberto. Foi originalmente desenvolvido por pesquisadores e engenheiros que trabalham no *Google Brain Team* dentro da organização de pesquisa de Aprendizado de Máquina da Google. Possui uma arquitetura flexível permite implantar computação para uma ou mais CPUs ou GPUs em uma área de trabalho, servidor ou dispositivo móvel com uma única API. Especializado para treinamento e implementação de redes neurais profundas, o Tensorflow é utilizado por exemplo: em aplicativos que incluem reconhecimento de imagem, tradução automatizada, classificação de imagens, detecção de *spam* em correios eletrônicos. Alguns dos aplicativos mais populares que utilizam o Tensorflow é: o Google Fotos, o Google Tradutor, o Uber e o SnapChat(TENSOLOW,).

O Tensorflow foi desenvolvido pelo Google com o intuito de facilitar e agilizar o desenvolvimento de sistema que utilizem de inteligência artificial e se tornou uma das bibliotecas mais aplicadas conforme mostra a figura 17.

Figura 17 – Comparativo de buscas sobre Tensorflow nos últimos cinco anos.



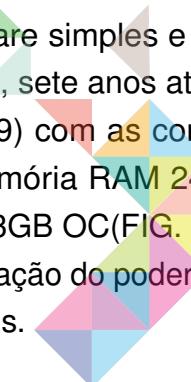
Oferece APIs para linguagens como Python, C++, Java e outros.

3 MATERIAIS E MÉTODOS

3.1 Equipamentos

Para tal projeto optou-se por utilizar a linguagem Python. Por trazer várias facilidades e funções nativas, a linguagem aprimora a computação e economiza em tempo no desenvolvimento e treinamentos das redes neurais. Assim como o Python, foi utilizado uma biblioteca Tensorflow para otimização nos treinamentos. Para armazenamento de algumas informações optou-se pelo banco de dados MySQL.

Haja vista que este trabalho tem o objetivo de verificar os limites de hardwares e do conceito de rede neurais convolucionais, houve a utilização de duas máquinas para treinamentos e testes. Inicialmente os treinamentos foram feitos em laptop comum, cuja as configurações de hardware limita-se em um processador Intel I5 540m, 8 GB de memórias RAM DDR3 1600MHz e 500 GB HDD Segate. Este laptop (FIG. 18) foi escolhido pois possui um hardware simples e antigo, uma vez que os componentes foram lançados em 2010, ou seja, sete anos atrás. Para os testes finais será utilizado um computador desktop (FIG. 19) com as configurações a seguir: um processador AMD ryzen 7 1700, 8 GB de memória RAM 2400 MHz, 240 GB SSD HiperX e uma placa de vídeo Galax GTX 1060 3GB OC(FIG. 20). Esta última configuração citada foi selecionada com o intuito da utilização do poder de paralelismo disponível pelas GPUs(Graphical Processing Units) atuais.



FASTFORMAT

Você precisar comprar esse documento para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.

10-page documents are free.

Figura 18 – Laptop — Meganote 4129



FASTFORMAT

Você precisar comprar esse documento para remover a marca d'água.

Própria (2017)
Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.

10-page documents are free.

Figura 19 – Computador



Própria (2017)

Figura 20 – Placa Gráfica — Galaxy OC GTX 1060



Você precisar comprar esse documento para remover a marca d'água.
http://www.galax.com/media/catalog/product/cache/20/image/500x500/9df78eab33525d08d6e5fb8d27136e95/g/a/galax_gtx1060_oc_3gb_box_card_-600x600.png

3.2 Materiais

You need to buy this document to remove the watermark.
10-page documents are free.

Durante todos os testes e treinamentos foram utilizadas bases de imagens e algoritmos conhecidos com o intuito de averiguar a real necessidade da utilização de GPUs nos treinamentos, optou-se por utilizar três base de imagens e modelos de algoritmos, apresentadas a seguir:

3.2.1 MNIST

A base MNIST é uma coleção de imagens de mais de 60 000 números escritos à mão para treinamento e 10 000 imagens para testes e avaliação do aprendizado. Também conhecida com o “Hello, World!” de pesquisadores que iniciam no universo do aprendizado do reconhecimento de padrões de imagens (LECUN,).

Figura 21 – MNIST



<https://kuanhoong.files.wordpress.com/2016/01/mnistdigits.gif?w=300&h=199>

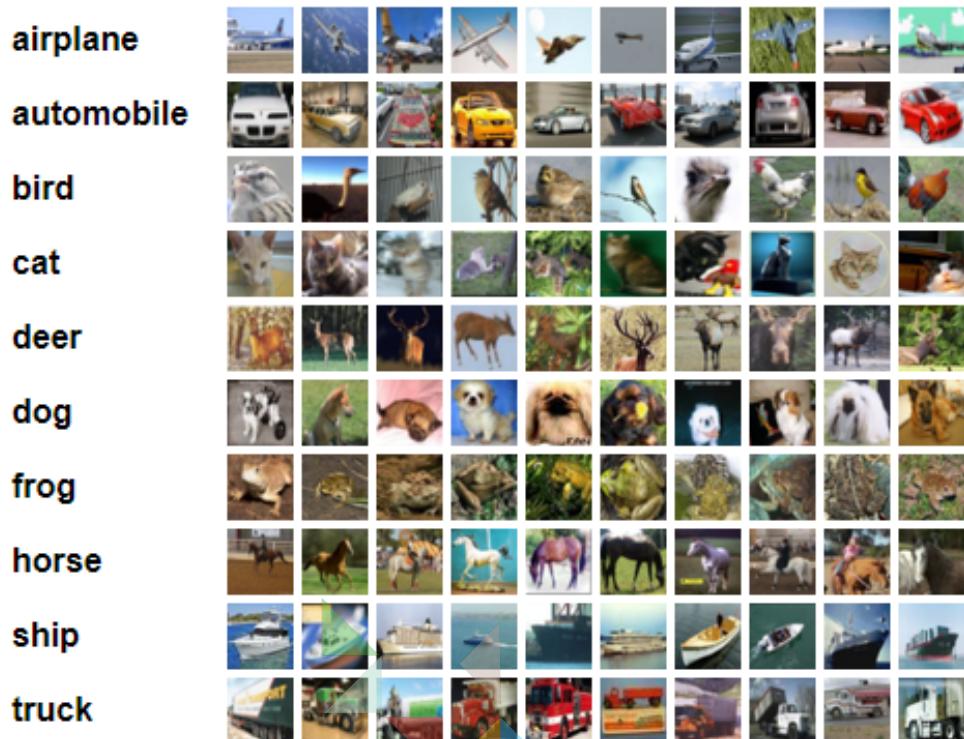
3.2.2 CIFAR 10

Também uma base de imagens muito conhecida por conter mais de 60 000 pequenas imagens de 32×32 pixels, catalogadas em 10 categorias, sendo 6 000 imagens por classe(KRIZHEVSKY; NAIR; HINTON,). Abaixo as dez classes:

FASTFORMAT

- Automóvel
- Caminhão
- Pássaro Documentos de 10 páginas são gratuitos.
- Avião You need to buy this document to remove the watermark.
- Cervo 10-page documents are free.
- Gato
- Cachorro
- Sapo
- Navio
- Cavalo

Figura 22 – Exemplos da base CIFAR 10



<https://www.cs.toronto.edu/~kriz/cifar.html>

3.2.3 Curso de rede neural convolutiva

Baseado no já citados exemplos, este material faz parte do curso de Aprendizado Profundo ministrado por Kirill Eremenko(EREMENKO,). Conta com 8 000 imagens classificadas em duas categorias (cão e gato) para treinamento e 2 000 imagens de teste.

3.3 Metodologia

comprar esse documento para remover a marca d'água.
Documentos de 10 páginas são gratuitos.

Os treinamentos foram feitos em ambos os computadores. Durante o processo de treinamento foram anotadas todas as informações necessárias para tais avaliações e comparações de modo a chegar em um resultado coeso. Ao longo dos treinamentos foram registradas as utilizações computacionais tanto dos processadores quanto da placa gráfica, assim como tempo de treinamento e utilização de memória.

4 RESULTADOS E DISCUSSÕES

4.1 Resultados

Após todos os testes efetuados durante o tempo do projeto, registrou-se uma diminuição significativa de tempo gasto nos treinamentos que foram feitos utilizando a GPU. Contudo, só houve melhora considerável no aprendizado quando foram acrescentadas camadas internas na rede convolutiva, aumentou-se a precisão da acurácia e aumentou proporcionalmente o tempo no treinamento.

Conforme descrito na tabela abaixo, houve uma economia media de 80% (porcento) no tempo gasto para os treinamentos realizados nas três bases.

Tabela 1 – Tempo médio gasto nos treinamentos.

| Base de Imagens | Tempo Médio de Treinamento | |
|------------------------|-----------------------------------|-----------|
| | Sem GPU | Com GPU |
| MNIST | 02 horas | 0,1 horas |
| CIFAR 10 | 38 horas | 07 horas |
| CURSO | 25 horas | 05 horas |

Própria (2017)

Além da otimização no tempo expedido, observou-se que os treinamentos feitos com o computador utilizando a GPU, apesar de estar executando o mesmo código, gastou em média (95%) a menos de segundos por lotes (batch). Assim como no tempo por lotes, houve uma desaceleração devido ao melhoramento de carregamento de imagens por lotes, em média 4 000% a mais. Resultando em um aprendizado mais rápido, porém, não mais preciso.

You need to buy this document to remove the watermark.

10-page documents are free.

Figura 23 – Treinamento Cifar10 — Computador sem GPU

```

2017-12-04 19:04:22.638836: step 200, loss = 3.65 (85.1 examples/sec; 1.504 sec/batch)
2017-12-04 19:04:37.755262: step 210, loss = 3.80 (84.7 examples/sec; 1.512 sec/batch)
2017-12-04 19:04:52.902889: step 220, loss = 3.83 (84.5 examples/sec; 1.515 sec/batch)
2017-12-04 19:05:07.332914: step 230, loss = 3.72 (88.7 examples/sec; 1.443 sec/batch)
2017-12-04 19:05:21.965740: step 240, loss = 3.94 (87.5 examples/sec; 1.463 sec/batch)
2017-12-04 19:05:36.598565: step 250, loss = 3.69 (87.5 examples/sec; 1.463 sec/batch)
2017-12-04 19:05:51.184591: step 260, loss = 3.67 (87.8 examples/sec; 1.459 sec/batch)
2017-12-04 19:06:05.599016: step 270, loss = 3.68 (88.8 examples/sec; 1.441 sec/batch)
2017-12-04 19:06:20.122642: step 280, loss = 3.49 (88.1 examples/sec; 1.452 sec/batch)
2017-12-04 19:06:34.724268: step 290, loss = 3.65 (87.7 examples/sec; 1.460 sec/batch)
2017-12-04 19:06:49.201093: step 300, loss = 3.57 (88.4 examples/sec; 1.448 sec/batch)
2017-12-04 19:07:03.272318: step 310, loss = 3.75 (91.0 examples/sec; 1.407 sec/batch)
2017-12-04 19:07:17.359142: step 320, loss = 3.47 (90.9 examples/sec; 1.409 sec/batch)
2017-12-04 19:07:31.461567: step 330, loss = 3.52 (90.8 examples/sec; 1.410 sec/batch)
2017-12-04 19:07:45.579592: step 340, loss = 3.56 (90.7 examples/sec; 1.412 sec/batch)
2017-12-04 19:07:59.666417: step 350, loss = 3.35 (90.9 examples/sec; 1.409 sec/batch)
2017-12-04 19:08:13.909242: step 360, loss = 3.49 (89.9 examples/sec; 1.424 sec/batch)
2017-12-04 19:08:28.230067: step 370, loss = 3.36 (89.4 examples/sec; 1.432 sec/batch)
2017-12-04 19:08:42.285692: step 380, loss = 3.38 (91.1 examples/sec; 1.406 sec/batch)

```

Própria (2017)

Figura 24 – Treinamento Cifar10 — Computador com GPU

```

2017-12-04 19:07:27.985145: step 250, loss = 3.53 (5119.3 examples/sec; 0.025 sec/batch)
2017-12-04 19:07:28.250815: step 260, loss = 3.62 (4818.0 examples/sec; 0.027 sec/batch)
2017-12-04 19:07:28.531779: step 270, loss = 3.56 (4555.7 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:28.813397: step 280, loss = 3.54 (4545.2 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:29.110322: step 290, loss = 3.45 (4310.9 examples/sec; 0.030 sec/batch)
2017-12-04 19:07:29.578799: step 300, loss = 3.61 (2732.3 examples/sec; 0.047 sec/batch)
2017-12-04 19:07:29.860432: step 310, loss = 3.49 (4544.9 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:30.126086: step 320, loss = 3.55 (4818.3 examples/sec; 0.027 sec/batch)
2017-12-04 19:07:30.422683: step 330, loss = 3.52 (4315.6 examples/sec; 0.030 sec/batch)
2017-12-04 19:07:30.719919: step 340, loss = 3.63 (4306.3 examples/sec; 0.030 sec/batch)
2017-12-04 19:07:30.985257: step 350, loss = 3.41 (4824.0 examples/sec; 0.027 sec/batch)
2017-12-04 19:07:31.262836: step 360, loss = 3.67 (4611.3 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:31.544126: step 370, loss = 3.49 (4550.5 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:31.841370: step 380, loss = 3.47 (4306.2 examples/sec; 0.030 sec/batch)
2017-12-04 19:07:32.122337: step 390, loss = 3.41 (4555.7 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:32.591146: step 400, loss = 3.33 (2730.3 examples/sec; 0.047 sec/batch)
2017-12-04 19:07:32.872770: step 410, loss = 3.79 (4545.1 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:33.154068: step 420, loss = 3.31 (4550.3 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:33.435347: step 430, loss = 3.17 (4550.6 examples/sec; 0.028 sec/batch)
2017-12-04 19:07:33.700684: step 440, loss = 3.23 (4824.1 examples/sec; 0.027 sec/batch)

```

Você precisar comprar esse documento para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

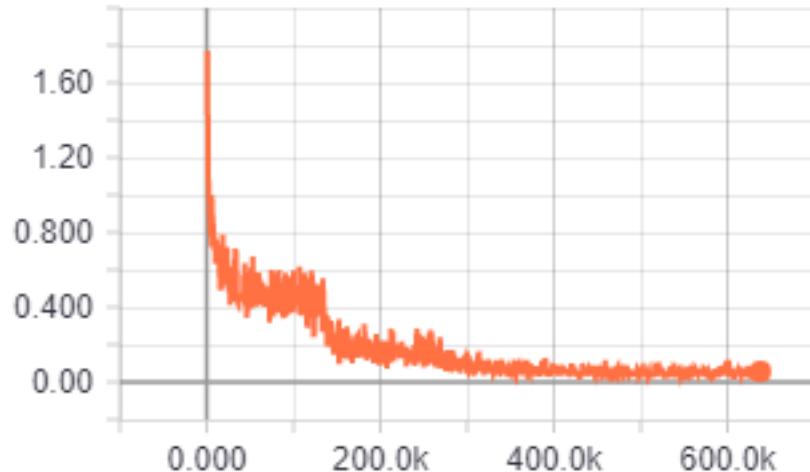
O Tensorflow disponibiliza ferramentas para acompanhamento durante os treinamentos. Gráficos e Histogramas que possibilitam a visualização e monitoramento dos algoritmos de aprendizagem podem ser acessados em servidor local. Basta a execução do comando:

```
"tensorboard - --logdir 'diretório'"
```

na janela de comandos do sistema. No qual deve-se indicar o diretório onde estão salvos os arquivos. Como pode-se inferir a partir dos gráficos apresentados abaixo, as redes neurais convolucionais profundas possuem um aprendizado rápido.

Figura 25 – Função de Cross- Entropy no treinamento CIFAR 10.

`cross_entropy_raw_`



Própria (2017)

Figura 26 – Perda no treinamento CIFAR 10.

`total_loss_raw_`



FASTFORMAT

Você precisar comprar esse documento para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

0.000 100.0k 200.0k 300.0k

You need to buy this document to remove the watermark.

Própria (2017)

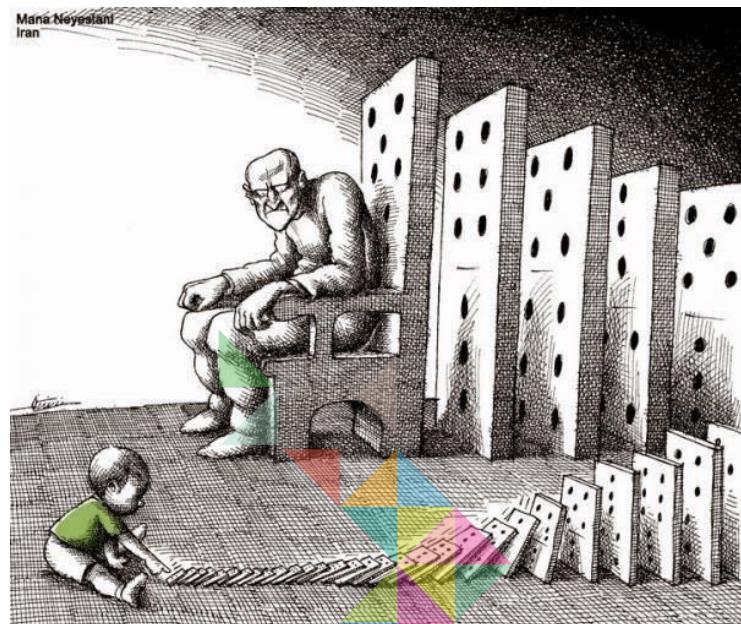
10-page documents are free.

4.2 Discussão

Os resultados encontrados no presente estudo sugerem que quanto maior for o poder de processamento do computador, mais eficiente e preciso serão os resultados dos treinamentos. Contudo, há a necessidade de ressaltar os gastos para tal desempenho. Equipamentos físicos possuem, de modo global, um custo proporcional ao seu desempenho e lançamento. Segundo Moore (1965), a tecnologia teria um crescimento de 100% a cada 18 meses e/ou o seu custo cairia pela metade neste

período. Houve sim um crescimento exponencial da tecnologia, quanto comparado ao seu preço de mercado, não obstante, ao previsto por Moore. Portanto, antes de qualquer investimento é necessário realizar cálculos de orçamento e averiguar se o custo dos equipamentos necessários são correspondentes ao real ganho no desempenho dos resultados estimados.

Figura 27 – Teoria do Caos



https://pm1.narvii.com/6356/a90e4d3a4753512abc4bbf5655fa60f5bf77c551_hq.jpg

Há de se pensar também a tecnologia apenas evoluiu graças a todos os estudos realizados em meados do século XX, pois segundo a Teoria do Caos, “o bater das asas de uma borboleta no Brasil pode gerar um furacão no Texas!”, ou seja, as decisões tomadas pela sociedade alteram completamente o futuro. Por esse motivo, a comunidade científica precisa descobrir o caminho mais sensato que possa conciliar o avanço tecnológico e a preservação e a diligência com o meio ambiente.

You need to buy this document to remove the watermark.

10-page documents are free.

5 CONCLUSÃO

A tecnologia está em constante evolução e crescimento, alcançando áreas jamais antes imaginadas, seja na saúde, no comércio ou na vida cotidiana de seus usuários. Sistemas de reconhecimento facial, detecção de spams em correios eletrônicos, identificação de fraudes em transações eletrônicas ou até mesmo segurança domiciliar, estão cada vez mais populares e de fácil acesso. Isso tudo está sendo possível devido a utilização da inteligência artificial. Pois, apesar do aprendizado ser complexo, demandar uma engenharia adaptada para cada tipo de problema e requerer hardwares potentes, sistemas que usufruem dos benefícios do aprendizado de máquina estão auferindo centenas de novos usuários todos os dias.

As grandes empresas estão focadas em conquistar novos horizontes e investem milhares em pesquisas e programas como por exemplo o AlphaGo e carros autônomos. Apesar disso, a tecnologia possui limitações, seja em equipamentos físicos ou na lógica de algoritmos não eficazes. Modelos iniciais de redes neurais não estão sendo capazes de consumir a quantidade de informações geradas e se adaptarem aos novos problemas. Sendo uma evolução das redes neurais, as redes neurais convolucionais profunda estão se mostrando mais eficientes e flexíveis.

Assim como as redes convolucionais, o Python está se popularizando novamente por ser uma linguagem de programação de alto nível. Com APIs que suportam muitas linguagens de programação, o Tensorflow é uma biblioteca de software que além de trazer agilidade para os desenvolvedores, está se apresentando bem maleável as novas situações e novos problemas.

FASTFORMAT

Conclui-se que as redes neurais convolucionais profundas são bem eficientes no aprendizado de padrões em imagens e vídeos. Há uma exigência de poder de processamento para realizar os treinamentos, pois, somente a utilização de processadores não é vantajosa. Potém, com o aproveitamento do poder de processamento em paralelo que as placas gráficas (GPUs) oferecem, os treinamento tornam-se mais rápidos. Não há um ganho tão assertivo, removendo a necessidade de maior tempo gasto nos treinamentos, assim como o ganho de desempenho no carregamento de imagens durante todo o processo.

Em virtude dos fatos mencionados, é possível que qualquer programador consiga desenvolver, treinar, testar e fazer o uso de sistemas que utilizem redes neurais convolucionais profundas em qualquer *hardware*, entretanto, quanto maior for o poder de processamento em paralelo, melhor será a performance do sistema.

A visão humana é um processo ativo de amostras sequencialmente que chegam na matriz óptica de uma maneira inteligente, com uma pequena e alta resolução. Espera-

se que o futuro do progresso na computação visual venha de sistemas que são treinados de ponta a ponta e combinem redes convolucionais com rede neurais recorrentes que usam aprendizagem de reforço para decidir onde procurar. Os sistemas que combinam aprendizado profundo e aprendizado de reforço são novos, mas já superam os sistemas de visão passiva nas tarefas de classificação e produzem resultados impressionantes ao aprender a jogar vários jogos.

Em última análise, o progresso importante na inteligência artificial surgirá através de sistemas que combinam aprendizagem de representação com raciocínio complexo. Embora o aprendizado profundo e o raciocínio simples tenham sido usados para o reconhecimento de fala e escrita por muito tempo, novos paradigmas são necessários para substituir a manipulação baseada em regras de expressões simbólicas por operações em vetores grandes.



Você precisar comprar esse documento para remover a marca d'água.
Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.
10-page documents are free.

Referências

- ARRUDA, F. *A história dos processadores*. 2011. Disponível em: <https://www.tecmundo.com.br/historia/2157-a-historia-dos-processadores.htm>. Acesso em: 04/12/2017.
- DENG, L.; YU, D. Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*, v. 7, n. 3-4, p. 197 – 387, 2014. Disponível em: <http://dx.doi.org/10.1561/2000000039>.
- EREMENKO, K. PART 2. CONVOLUTIONAL NEURAL NETWORKS (CNN). Disponível em: <https://www.superdatascience.com/deep-learning/>. Acesso em: 01/08/2017.
- FERREIRA, A. dos S. Redes Neurais Convolucionais Profundas na Detecção de Plantas Daninhas em Lavoura de Soja. 03 2017.
- GIBNEY, E. Google AI algorithm masters ancient game of Go. *Nature Publishing Group*, 01 2016. Disponível em: <https://www.nature.com/news/google-ai-algorithm-masters-ancient-game-of-go-1.19234#/b1>.
- KRIZHEVSKY, A.; NAIR, V.; HINTON, G. CIFAR 10. Disponível em: <https://www.cs.toronto.edu/~kriz/cifar.html>. Acesso em: 25/10/2017.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*, v. 60, n. 6, p. 84 – 90, 06 2017. Disponível em: <http://doi.acm.org/10.1145/3065386>. Acesso em: 01/12/2017.
- LECUN, Y. THE MNIST DATABASE of handwritten digits. Disponível em: <http://yann.lecun.com/exdb/mnist/>. Acesso em: 01/10/2017.
- LECUN, Y. et al. Handwritten Digit Recognition with a Back-Propagation Network. In: *Advances in Neural Information Processing Systems*. [S.I.]: Morgan Kaufmann, 1990. p. 396 – 404.
- MOORE, G. E. Cramming more components onto integrated circuits. *Electronics*, v. 38, n. 8, 1965. 04/09/2017 cisar comprar esse documento para remover a marca d'água.
- PYTHON Tutorial. Disponível em: https://www.tutorialspoint.com/python/python_functions.htm. Acesso em: 01/12/2017.
- RAINAS, R.; MADHAVAN, A.; NG, A. Y. Large-scale Deep Unsupervised Learning using Graphics Processors. 2009.
- RANZATO, M. et al. Efficient Learning of Sparse Representations with an Energy-Based Model. *Advances in Neural Information Processing Systems 19*, p. 1134 – 1144, 2007.
- ROSENBLATT, F. The Perceptron: A Perceiving and Recognizing Automaton Project Para. *Cornell Aeronautical Laboratory*, 1957. Report 85-460-1. Disponível em: https://books.google.com.br/books?id=P_XGPgAACAAJ. Acesso em: 12/11/2017.
- SQUIRE, L. et al. *FUNDAMENTAL NEUROSCIENCE*: Third Edition. 3. ed. [S.I.]: Elsevier Inc., 2008a. ISBN 978-0-12-374019-9.

SQUIRE, L. et al. *FUNDAMENTAL NEUROSCIENCE*. 3. ed. [S.I.]: Elsevier Inc., 2008b. ISBN 978-0-12-374019-9.

TENSOFLOW. Disponível em: <https://www.tensorflow.org/>. Acesso em: 04/12/2017.

TURING, A. M. Computing Machinery and Intelligence. *Mind*, Oxford University Press on behalf of the Mind Association, v. 59, n. 236, 10 1950. Disponível em: <http://www.jstor.org/stable/2251299>. Acesso em: 01/12/2017.

WERBOS, P. J. Backpropagation Through Time: What It. Does and How to Do It. *Proceedings of the IEEE*, v. 78, n. 10, p. 1550 – 1560, 10 1990.

WIDROW, B.; LEHR, M. A. 30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, v. 78, n. 9, p. 1415 – 1442, Setembro 1990. Disponível em: <http://ieeexplore.ieee.org/document/58323/>. Acesso em: 12/11/2017.

WIDROW, B.; LEHR, M. A. Adaptive neural networks and their applications. *Int. J. Intell. Syst.*, v. 8, n. 4, p. 453 – 507, 1993. Disponível em: <http://dx.doi.org/10.1002/int.4550080403>.



Você precisar comprar esse documento para remover a marca d'água.
Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.
10-page documents are free.



Você precisar comprar esse documento para remover a marca d'água.
Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.
10-page documents are free.

APÊNDICE A – Instalação do Python

Instalando o Python

A distribuição Python está disponível para uma grande variedade de plataformas. Você precisa baixar apenas o código binário aplicável para sua plataforma e instalar o Python.

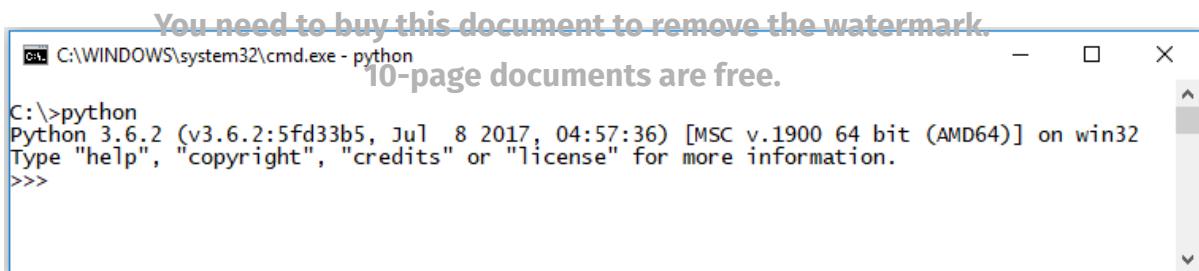
Se o código binário para sua plataforma não estiver disponível, você precisa de um compilador C para compilar o código-fonte manualmente. Compilar o código-fonte oferece mais flexibilidade em termos de escolha dos recursos que você precisa em sua instalação.

Aqui estão as etapas para instalar a máquina Python no Windows.

- Abra um navegador da Web e vá para <https://www.python.org/downloads/>.
- Siga o link para o arquivo python-XYZ.msi do instalador do Windows, onde XYZ é a versão que você precisa instalar.
- Para usar este instalador python-XYZ.msi, o sistema Windows deve suportar o *Microsoft Installer 2.0*. Salve o arquivo do instalador em sua máquina local e depois execute-o para descobrir se o seu aparelho suporta MSI.
- Execute o arquivo baixado. Isso traz o assistente de instalação do Python, que é realmente fácil de usar. Apenas aceite as configurações padrões, aguarde até que a instalação seja concluída e você terminou.

Para se certificar que o Python foi instalado corretamente, abra o *prompt* de comando e execute a seguinte linha de comando “python” que irá aparecer a versão do Python instalada: **Documentos de 10 páginas são gratuitos.**

Figura 28 – Verificando instalação/versão do Python



```
C:\>python
You need to buy this document to remove the watermark.
C:\>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

APÊNDICE B – Instalação do Tensorflow

O Tensorflow possui suporte para CPU e para GPU também, então primeiramente, é necessário a escolha de qual plataforma será instalado.

B.1 Instalação da versão Tensorflow-CPU

Para a instalação da versão Tensorflow-CPU, siga os passos abaixo:

- Certifique-se que há uma versão do Python instalada e a mesma seja compatível com o Tensorflow. Caso necessário realize a instalação ou atualização do Python para uma versão compatível.

OBS.: Neste trabalho foi utilizado a versão Python 3.6.2.

- Abra o *prompt* de comando do Windows e execute o seguinte comando:

C:\> pip install –upgrade tensorflow

B.2 Instalação da versão Tensorflow-GPU

Requisitos para executar TensorFlow com suporte de GPU

Se você pretende instalar o TensorFlow com suporte de GPU, o seguinte software NVIDIA deve estar instalado em seu sistema (OBS.: As versões citadas abaixo são exclusivas para o Tensorflow 1.3.0.):

Você precisa comprar esse documento para remover a marca d'água.

CUDA® TOOLKIT 8.0. Documentos de 10 páginas são gratuitos.

- 1) Acesse: <https://developer.nvidia.com/cuda-downloads> e faça o download da versão que aceita as configurações padrões no processo de instalação.
- 2) Configurando o caminho (PATH):

- > Clique no menu Iniciar e procure por “variáveis de ambiente”
- > Clique no botão “Variáveis de Ambiente”
- > Em “Variáveis de Sistemas” selecione “PATH” e clique em editar
- > Adicione os seguintes caminhos:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\bin

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\libnvvp

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\extras\CUPTI\libx64

- **Os drivers NVIDIA associados ao CUDA Toolkit 8.0.**

- 1) Certifique-se que os *drivers* da placa gráfica estão atualizados, caso necessário, atualize-os.
- 2) Para atualizar os *drivers*, acesse: <http://www.nvidia.com/Download/index.aspx>
- 3) Selecione o modelo da sua GPU, faça o *download* e instale a versão mais atualizada do *driver*.

- **CuDNN v6.0.**

- 1) Acesse <https://developer.nvidia.com/rdp/cudnn-download> e crie uma conta (se necessário).
- 2) Selecione CUDNN 6.0 para CUDA tool kit 8.
- 3) Extraia o conteúdo do arquivo zip e coloca-os em algum lugar (eu escolhi meu drive C)
- 4) Adicione um caminho para a pasta bin (Da mesma maneira feita anterior). Por exemplo:

FASTFORMAT

C:\cuda\bin

Você precisa de uma GPU (GTX 650 ou mais nova) para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

- Abra o *prompt* de comando do Windows e execute o seguinte comando:

You need to buy this document to remove the watermark.

C:\> pip install --upgrade tensorflow-gpu

B.3 Testando a instalação

- Abra o *prompt* do Windows e execute:

C:\> python

- Após a inicialização do Python, execute os seguintes comandos:

```
>> import tensorflow as tf  
>> ola = tf.constant('Olá, TensorFlow!')  
>> sess = tf.Session()  
>> print(sess.run(ola))
```

- Após feito isso, você irá visualizar na tela:

Olá, TensorFlow!



Você precisar comprar esse documento para remover a marca d'água.
Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.
10-page documents are free.

APÊNDICE C – MNIST Tutorial

Figura 29 – Código MNIST — parte 1

```

1  from __future__ import absolute_import
2  from __future__ import division
3  from __future__ import print_function
4
5  import argparse
6  import sys
7
8  from tensorflow.examples.tutorials.mnist import input_data
9  import tensorflow as tf
10 FLAGS = None
11
12
13 def main(_):
14     # Importa as imagens
15     mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)
16
17     # Criando as variáveis
18     # X -> são as entradas. Cria-se um vetor 784 pois cada imagem possui 28 x 28 pixels
19     x = tf.placeholder(tf.float32, [None, 784])
20
21     # W -> são os pesos (Weights) que serão multiplicados por ele para produzir um vetor
22     # de evidências para as 10 classes de diferença.
23     W = tf.Variable(tf.zeros([784, 10]))
24
25     # b -> são os bias que possui um vetor com 10 valores que serão adicionados para gerar as saídas
26     b = tf.Variable(tf.zeros([10]))
27
28     # y -> são as saídas, onde é multiplicado as entradas 'X' * os pesos 'W' e somado as bias 'b'
29     y = tf.matmul(x, W) + b
30
31     # Alocando um espaço na memória que será definido a perda.
32     y_ = tf.placeholder(tf.float32, [None, 10])
33
34     # Definindo a variável de cross_entropy que utiliza a função softmax, a qual nos permite
35     # interpretar as saídas como probabilidades
36     cross_entropy = tf.reduce_mean(
37         tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))
38
39     # Definindo os passos do treinamento com o método de Gradiente Descendente
40     # train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
41
42     # Iniciando a sessão
43     sess = tf.InteractiveSession()
44
45     # Primeiro, temos que criar uma operação para inicializar as variáveis que criamos
46     tf.global_variables_initializer().run()
47

```

Você precisa comprar esse documento para removê-lo.

Este documento de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.

10-page documents are free.

Figura 30 – Código MNIST — parte 2

```

48     # A partir desta linha, o real treinamento será iniciado
49     # O treinamento inicia em 0 e será repetido por durante 1000 vezes
50     for _ in range(1000):
51         # A cada passo será utilizada 100 imagens randomicamente para teste
52         batch_xs, batch_ys = mnist.train.next_batch(100)
53         sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
54
55     # Avaliando os resultados obtidos
56     # tf.argmax é uma função extremamente útil que lhe dá os índices, ou argumentos,
57     # da entrada na qual as saídas da função são os maiores
58     correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
59
60     # Finalmente, encontramos sua acurácia em nossos dados de teste.
61     accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
62
63     # Imprimimos o resultado da acurácia
64     print(sess.run(accuracy, feed_dict={x: mnist.test.images,
65                                         y_: mnist.test.labels}))
66
67 if __name__ == '__main__':
68     parser = argparse.ArgumentParser()
69     parser.add_argument('--data_dir', type=str, default='/tmp/tensorflow/mnist/input_data',
70                         help='Directory for storing input data')
71     FLAGS, unparsed = parser.parse_known_args()
72     tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)

```

https://github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/mnist/mnist_softmax.py

Para esse treinamento espera-se um acurácia de 92%.

FASTFORMAT

Você precisar comprar esse documento para remover a marca d'água.

Documentos de 10 páginas são gratuitos.

You need to buy this document to remove the watermark.

10-page documents are free.