Visão Computacional

Nicole Rakov 96661, Tiago Oliveira 99064

Visão Computacional

Departamento de electrónica e telecomunicações

Nicole Rakov 96661, Tiago Oliveira 99064
nicolemrakov@ua.pt, tiagooliveira95@ua.pt

Resumo

Visão Computacional, uma área associada ao Deep Learning e Machine Learning, que permite que computadores sejam capazes de interpretar imagens através de redes neurais - modelos computacionais que remetem a organização organismos inteligentes.

Essas redes são constituídas por várias camadas de informações que são usadas para filtrar conteúdo e obter resultados. Uma vez que são capazes de interpretar informações mais rapidamente e em maior quantidade, tornam trabalhos mais ágeis e promovem avanços no cotidiano da sociedade, agilidade em pesquisas científicas e em diversos outros campos, desde anúncios, carros autónomos a agilidade no diagnóstico de doenças, como o cancro de pele.

Logo, percebe-se que se trata de uma tecnologia muito útil para a sociedade e está acompanhada de um grande potencial de expansão, tanto para áreas fora do âmbito computacional, mas também para se aperfeiçoar, tornando-se cada vez exacta e precisa em seus resultados.

Conteúdo

Capítulo 1

Introdução

O objectivo deste relatório técnico é expor e debater a respeito do Deep Learning e da Visão Computacional. Tecnologias que se encontram presentes em diversas áreas fora do âmbito computacional e estão associadas a diversos avanços científicos, mas também fazem parte do quotidiano. E que vem demonstrando potencial de crescimento devido a sua capacidade de aprofundar e agilizar a análise da informação, principalmente no que convém a interpretação de imagens e a detecção de padrões.

Neste relatório será exemplificado o uso dessas fermentas tendo como embasamento a análise de artigos científicos de diversas áreas do conhecimento e publicações em geral. Além disso, será observado vantagens e desvantagens do uso dessas tecnologias. Ademais, será exemplificado o uso e funcionamento das redes neurais convolucionais e recorrentes. Redes essas que formam a base para a Visão Computacional, permitindo assim o processamento de imagens de maneira inteligente e eficaz. Tornando um processo que antes restrito a seres vivos, possível a máquinas.

Este documento está dividido em 7 capítulos. Após esta introdução, no Capítulo 2 é apresentada a análise dessas duas tecnologias. Em seguida, no Capítulo 3 é abordada a visão de computacional, detalhando-a de forma a explicar seu funcionamento e falaremos de filtros que são utilizados para detectar detalhes num objecto. Posteriormente, no Capítulo 4 será tratado das Redes Neurais Convolucionais. Já o Capítulo 5 tem foco na Rede neural recorrente. Finalizando com o Capítulo 6 no apresenta as conclusões do trabalho e o Capítulo 7 que informa a contribuição de cada autor.

Capítulo 2

Análise

Deep Learning Deep Learning (DL) é uma técnica de Machine Learning Machine Learning (ML), tecnologia esta que permite que máquinas sejam capazes de aprender por si próprias, basicamente pela identificação de padrões. DL, consiste do uso da arquitectura neural - neural network architectures, nomenclatura dada pela semelhança do seu aspecto a redes neurais de organismos inteligentes - por isso esta especialização do ML também é chamado de deep neural network ou Deep Neural Networks (DNN).

A grande vantagem do DL é que permite a detecção de padrões complexos e exaustivos, com mais facilidade e rapidez que um ser humano seria capaz. Dessa forma, DNN está a atuar em diversas áreas do conhecimento.

Sua presença se dá desde o auxílio em buscas em websites promovendo o filtro de conteúdo pertinente, identificação de produtos visualmente semelhantes para anúncios em redes sociais e em e-commerce, bem como diversos websites, resolução de problemas, bem como uma vasta actuação na ciência, negócios e segurança.

Tendo em vista que o Deep Learning, permite identificar padrões complexos, o mesmo facilitou o a predição do comportamento de fármacos, drogas e moléculas. Sendo assim, DL e DNN são usados para reconhecer propriedades farmacológicas de múltiplas drogas em diferentes organismos e situações distintas [Ali+16]. Entretanto, o uso de DL no campo farmacêutico é muito promissor e vai além de predições de actividades biológicas, o mesmo se vê útil na questão do descobrimento de fármacos, design de moléculas e análise biológica de imagens [Che+18].

A grande vantagem do uso do Deep Learning na medicina é a hierarquização da informação e conexões entre fenómenos, por meio de uma tecnologia, evitando um desgastante trabalho manual. Essa tecnologia se tornou grande aliada da medicina na análise de imagens, detecção, segmentação e registo de informações triviais para um diagnóstico [Ker+18] [Est+17]. Seu uso vem ajudando no diagnóstico de doenças e mutações [LBH15].

Na área financeira o DL vem sendo usado para a predição e classificação de problemas como, instabilidade de preços e levantamento de risco. E vem se

mostrando um grande aliado desse segmento [HPW17].

Assim como o Deep Learning, a Visão Computacional Visão Computacional (VC) também está presente em áreas além da computação e tecnologia e mostra promissora, principalmente no que convém a facilitar avanços científicos e facilitar actividades do quotidiano.

Este segmento da computação se tornou conhecido pela sua actuação em tecnologias como carros autónomos e também na área de segurança, com o reconhecimento facial. Mas, é na área comercial, com a questão de anúncios, que esse segmento de DNN demonstra grande actuação[AM18].

No entanto, sua participação em áreas do conhecimento não é restrita apenas a esses dois segmentos. O uso de DL na agricultura por exemplo, contribuiu para diversos avanços, uma vez que revoluciona a agilidade nos resultados, como a classificação de imagens, antes feitas manualmente, um processo desgastante de demandava tempo e esforço [KP18]. Em seu livro "Computer Vision Technology for Food Quality Evaluation", [**sun2017**], exemplifica umas das utilizações da visão de computador na industria alimentícia. O autor também mesmo afirma que a VC se trata de um ramo em crescente expansão, não só na questão que envolve a análise de alimentos, mas também diversas áreas.

A VC, por exemplo vem sendo usada na ecologia, aumenta a eficiência na questão da analise de imagens, ferramenta importante para esta área cinética [Wei18]. O mesmo beneficio também foi encontrado por [Bar+17], para a automação do processo de identificação de plantas.

Ademais, o uso da Visão Computacional também se dá no âmbito do reconhecimento facial, detecção de objectos em geral bem como no reconhecimento de acções e actividades, como debatido em [Vou+18]. O mesmo também comenta o uso de DL conjunto com VC para o reconhecimento de vídeos, promovendo a detecção e classificação de eventos em geral por meio da análise de segmentos de vídeos. Ferramenta muito útil e também usada por diversas plataformas de streaming de vídeos.

Contudo, apesar das diversas vantagens na qual DL e VC estão associadas, desvantagens testam presentes quando se trata dessas ferramentas. Uma delas é a questão financeira, ambas as tecnologias necessitam de profissionais qualificados, bem como computadores adequados para que o processamento da informação e outras tarefas sejam realizadas com êxito.

Além disso, para obter resultados satisfatórios identificação de padrões, esses recursos computacionais necessitam de uma vasta quantidade de informação inicial – por isso, mostram-se tão eficaz em projectos de grande porte, como pesquisas científicas.

Capítulo 3

Visão Computacional

Uma imagem não passa de uma matriz tridimensional que contem os valores de Red, Green, Blue (RGB) estes valores estão compreendidos entre 0 e 255, para imagens de 8 bits.

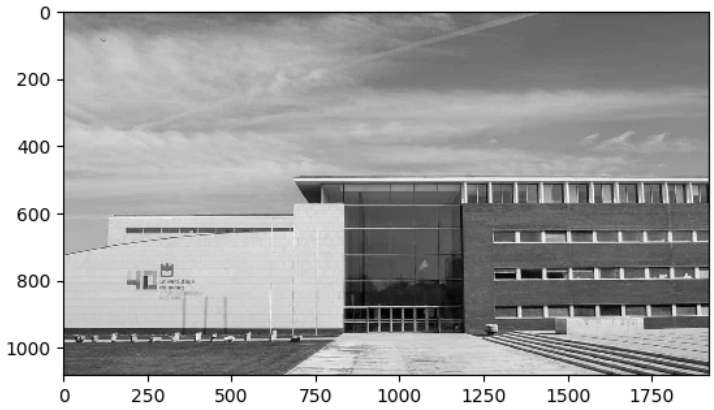
Analisaremos a seguinte imagem da Universidade de Aveiro (UA):

Figura 3.1: Reitoria UA

$https://media-manager.noticiasaominuto.com/1920/naom_58ca60ead971f.jpg$

Podemos observar que a imagem tem coordenadas X e Y, e o ponto de referencia x=0, y=0 faz corresponder a parte superior esquerda da imagem.

Para um humano, é simples olhar para esta foto e saber que se trata de um edifício, e conseguimos ver com facilidade todas as margens entre o céu e o edifício, e entre a relva e o pavimento, contudo para um computador ver estas características todas necessita primeiro de filtrar a imagem para a tornar mais simples de entender, pois a imagem acima não passa de uma matriz de valores para o computador, logo ele não consegue fazer sentido dessa matriz de valores. Usando visão computacional podemos obter as margens de um objecto, como a cor não é importante para detectarmos margens vamos converter a imagem para tons de cinza usando a liberaria OpenCV o que vai facilitar o processamento.



Para obter a imagem acima corremos este código no JupiterNotebook:

```
import numpy as np
import matplotlib.image as pyimg
import matplotlib.pyplot as pyplot
import cv2

%matplotlib qt

imagem = pyimg.imread('C:\\Users\\tiago\\Desktop\\ua.jpg')

escala_cinza = cv2.cvtColor(imagem, cv2.COLOR_RGB2GRAY)

pyplot.imshow(escala_cinza, cmap='gray')
```

Listing 3.1: Imagem RGB para escala de cinzas

Introdução a Kernels.

Um kernel é uma matriz n*n, que sendo aplicado a uma matriz de pixeis, o resultado final será uma imagem com um filtro aplicado.

Em visão computacional existem vários filtros, uns que detectam linhas verticais, linhas horizontais entre outros.

Vejamos um exemplo de uma matriz que detecta arestas verticais.

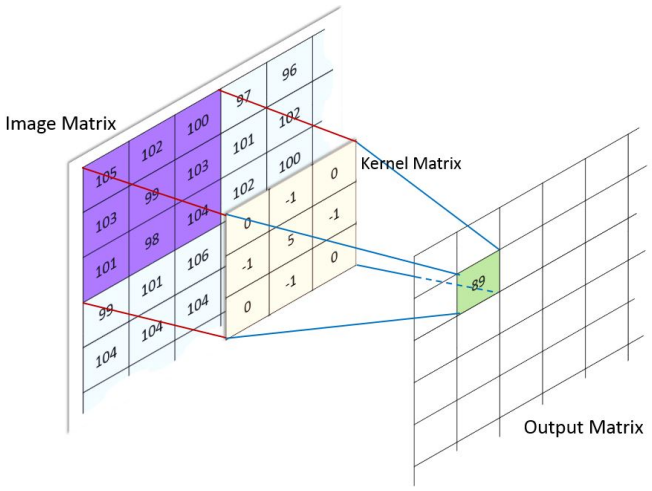
$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Este filtro consegue detectar diferenças em intensidade sobre pixeis vizinhos, reparemos que a parte esquerda assume valores negativos e a direita positivos, e que a soma de todos os elementos da matriz é igual a 0, ao colocar esta matriz sobre outra matriz de pixeis, e multiplicarmos os elementos da matriz K, pela matriz de pixeis, se a matriz de pixeis for consistente, a soma da matriz resultante será 0 ou perto de 0.

Contudo se os valores a esquerda forem inferiores aos da direita, a matriz resultante terá valores mais elevados a direita e negativos a esquerda que não se cancelam um ao outro, assim na soma dos elementos desta matriz se obtivermos um valor elevado, vamos poder dizer com certeza que na imagem existe uma grande variação da esquerda para a direita, o que significa que estamos perante uma aresta de um objecto.

As diferenças são detectadas aplicando o filtro e depois subtraindo os pixeis, se o valor não for 0, sabemos que existe uma diferença, positiva ou negativa, que fara com que o pixel fique mais claro ou mais escuro.

Esta matriz ira passar por todos os pixeis da imagem.



A matriz é colocada sobre o pixel, e multiplicamos os valores dos pixeis pelo valor correspondente do nosso kernel.

Por exemplo em baixo temos uma matriz I que foi retirada de uma certa imagem

$$I = \begin{bmatrix} 50 & 100 & 200 \\ 50 & 100 & 220 \\ 50 & 100 & 210 \end{bmatrix}$$

Conseguimos ver que a esquerda os valores são baixos, e na direita altos, ou seja os pixeis passam de escuro para claro muito rapidamente, podemos afirmar então que estamos perante uma aresta vertical, mas como podemos ver isso usando o Kernel acima mencionado ?

Ao multiplicar cada elemento da matriz K por I obtemos a seguinte matriz

$$R = \begin{bmatrix} -50 & 0 & 200 \\ -100 & 0 & 440 \\ -50 & 0 & 210 \end{bmatrix}$$

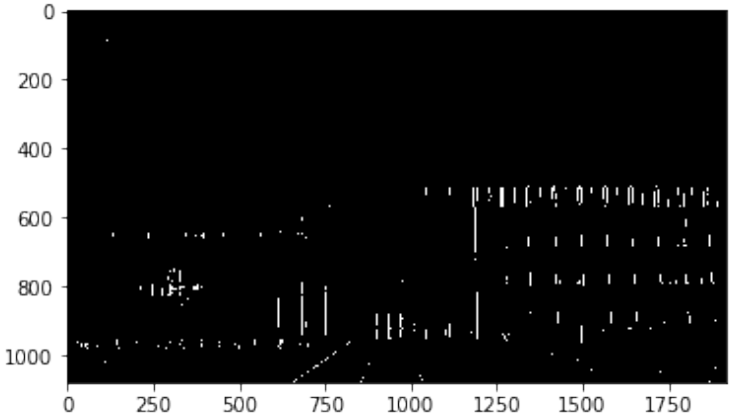
Agora sumamos todos os elementos da matriz que dá 650, isto significa que existe uma grande diferença da esquerda para a direita, então o valor do pixel central passa a ser 255.

Executamos este código, que ira passar a matriz k por todos os pixeis da imagem:

```
imagem = pyimg.imread('C:\\Users\\tiago\\Desktop\\ua.jpg')

escala_cinza = cv2.cvtColor(imagem, cv2.COLOR_RGB2GRAY)

kernal = np.array([
    [-1,0,1],
    [-2,0,2],
    [-1,0,1]
])
#Usamos um threshold, isto faz com que os pixeis fiquem
    compreendidos entre 0 e 1 que faz com que a imagem fique mais
    nitida
n, imagemFiltrada = cv2.threshold(cv2.filter2D(escala_cinza,-1,
    kernal_vertical),100,255, cv2.THRESH_BINARY)

pyplot.imshow(imagemFiltrada, cmap='gray')
```

Listing 3.2: Aplicação de um filtro vertical

Fazendo isto para todos os pixeis da imagem, obtemos:

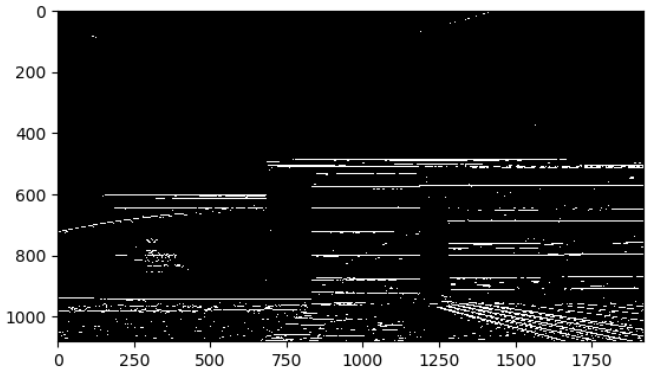


Nesta imagem podemos claramente ver todas as arestas verticais.

Um exemplo de um Kernel que consegue detectar arestas horizontais é o seguinte:

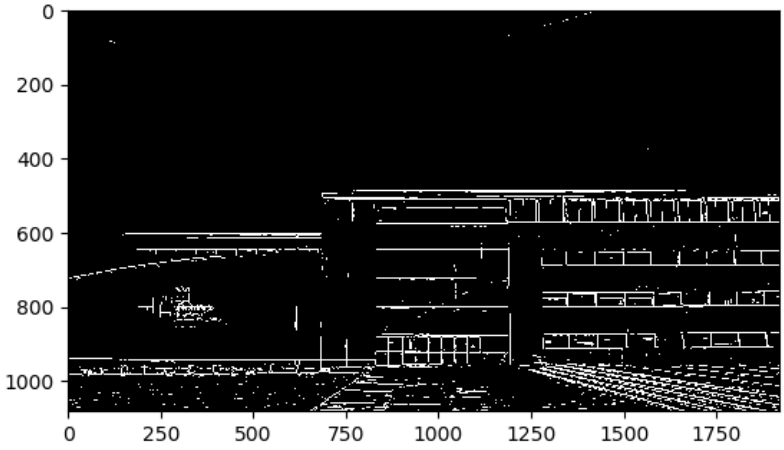
$$K = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Agora aplicando este kernel usando o mesmo processo obtivemos esta imagem:



Usando este filtro, podemos ver que obtivemos uma imagem em que as arestas horizontais estão claramente visíveis.

Neste momento temos 2 imagens, uma com as arestas verticais e outra com arestas verticais, como uma imagem se trata de uma matriz, para juntar as duas, basta somarmos uma á outra




```
pyplot.imshow(imagem_horizontal + imagem_vertical, cmap='gray')
```

Listing 3.3: Aplicação de um filtro vertical e horizontal

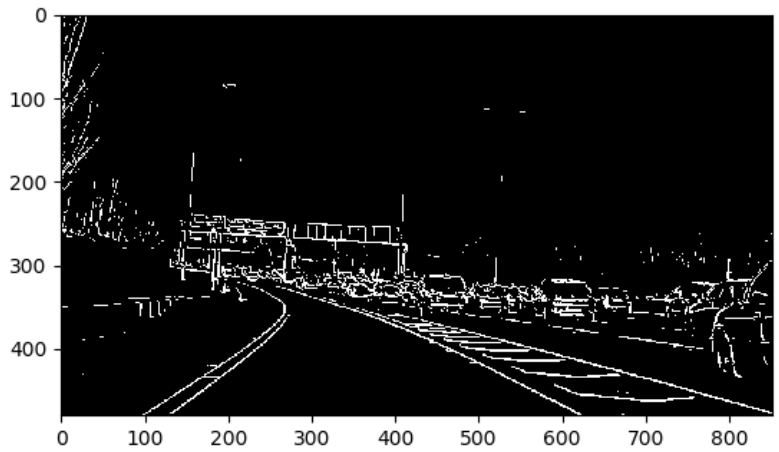
Juntando as duas imagens obtemos uma imagem completa, com arestas horizontais e verticais bem visíveis.

Podemos muito rapidamente aperceber nos de como esta ferramenta pode ser útil para detectar os limites de uma estrada.

Nós, humanos, ao olhar para esta imagem conseguimos muito facilmente ver os veículos circulam na via, saber quais são os limites da estrada, contudo, um computador teria bastante dificuldade a entender esta imagem, lembramos que o computador apenas 'vê' uma matriz de números.

Desta forma temos de simplificar esta imagem, usando o mesmo processo, passamos a imagem para escala de cinzas, e usamos 2 kernels para detectar as arestas horizontais e verticais.

Aplicando um filtro horizontal e vertical obtivemos esta imagem:



Esta imagem tens valores de pixeis compreendidos entre 0 e 1, em que 0 é preto e 1 é branco, ficando assim, mais fácil para um computador entender a sua localização na estrada.

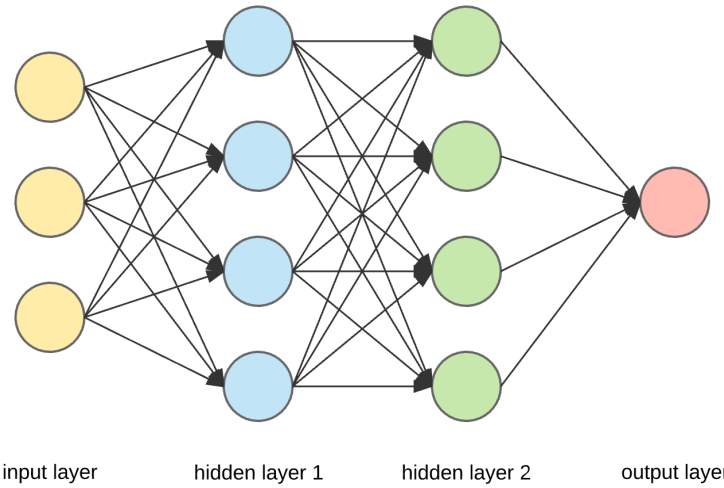
Podíamos até, passar o filtro por cada matriz RGB da imagem, assim podíamos até detectar formas com diferentes cores, como por exemplo, diferenciar linhas brancas de amarelas numa estrada.

Passando esta foto por uma Convolutional Neural Networks (CNN), tendo uma base de dados suficientemente grande, podemos treinar uma rede neuronal a detectar os limites da estrada bem como carros e sinais.

Capítulo 4

Redes Neurais Convolucionais

Redes neurais convulsionais, ou CNN são constituídas por varias camadas, e por módulos aos quais chamamos nodes.



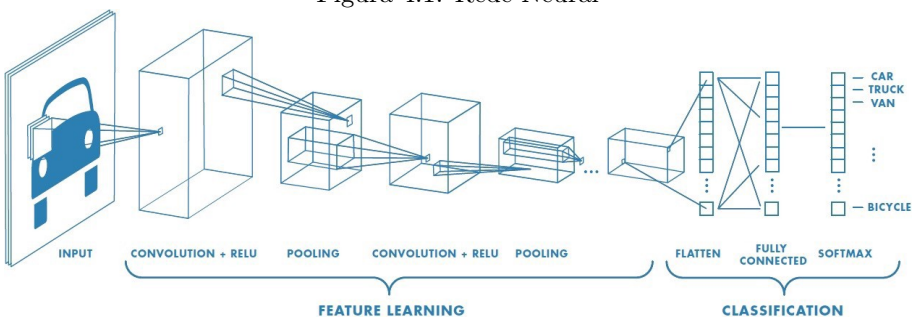
Os frameworks mais conhecidos são o TenserFlow e PyTorch, a rede neural recebe como input os pixeis de uma imagem, n*1, em que n é o numero de pixeis determinada imagem.

As imagens são redimensionadas para um valor que pode ser definido, assim a camada inicial terá sempre o mesmo numero de neurónios independentemente do tamanho da imagem de entrada.

A imagem passara depois a outras camadas escondidas, o numero de camadas, é também definido manualmente, este pode ter um impacto quer positivo quer negativo, tendo este valor de ser afinado para obter os melhores resultados.

A imagem vai passar por várias camadas, estas camadas e nodes têm como objectivo filtrar, simplificar, e remover aspectos que não são importantes.

Figura 4.1: Rede Neural



$https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148$

Na imagem acima a imagem passa por uma primeira camada, que vai aplicar filtros, tais como os mencionados no capitulo 3, estes passam então por uma função de activação, neste caso uma Rectified Linear Unit (ReLU), contudo existem outras mais especificas para cada caso.

Esta converte todos os valores negativos para 0, tornando o precessão mais eficiente e rápido.

Uma Pooling Layer, pode ser máxima, ou média e somatória, numa camada pooling máxima, seleccionamos uma secção da imagem, por exemplo 2x2 pixeis, e obtemos o pixel com o valor máximo, e associamos a nova imagem, desta forma estamos a reduzir o tamanho da imagem para metade, pois por cada 2x2 pixeis na imagem original estara associado apenas 1 pixel na imagem resultante.

No modo de média, o processo é igual, mas em vez de seleccionarmos o pixel com o valor mais alto, sumamos todos os elementos da matriz e dividimos pelo numero de elementos, que nos da a média, este valor é então utilizado para a nova imagem.

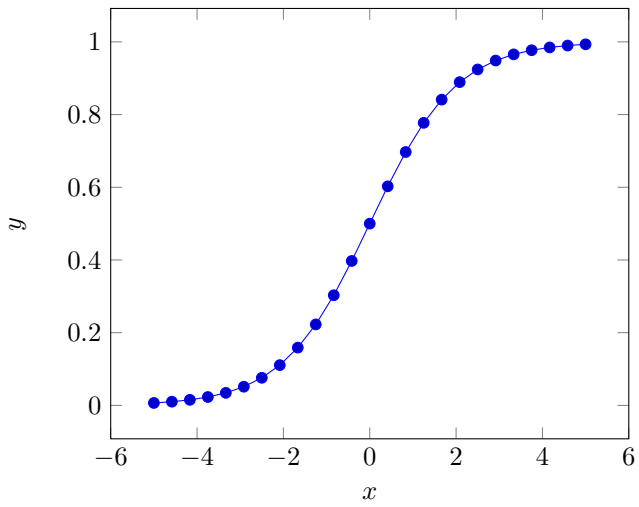
No modo sumatorio, iremos somar todos os valores de pixeis da matriz, e este valor será o valor do pixel na imagem.

As camadas finais, são as camadas de classificação, existiram nodes que ficaram activos quando for detectado um farol de um carro, uma roda, uma carroçaria entre outros elementos que constituem um carro.

No segundo nível vamos obter uma matriz n*1, em que a cada n faz corresponder a uma classificação, se a rede estiver bem treinada, e se na camada anterior tivermos os nodes que fazem corresponder a elementos de um carro com valores elevados, a classificação que representa o carro na camada final tera um valor elevado.

Este valor é representado de 0 a 1, sendo este calculado na camada softmax, neste caso é usada a função sigmóide.

$$\omega(x) = \frac{1}{1 + e^{-x}}$$



Esta função aproxima números elevados a 1, e números baixos a 0.

Este processo, do inicio ao fim, é chamado de feed-forward, os nodes de uma CNN têm pesos e um número de bias, este numero é usado para aumentar a diversidade da rede neural, estes pesos e biases têm de ser afinados de forma, a que no fim, obtenhamos uma classificação correcta, este processo de treino tem o nome de back-propagation.

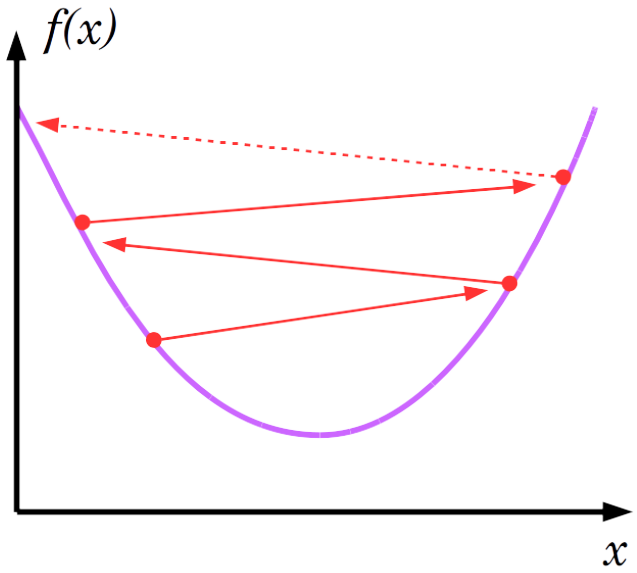
Este processo é bastante complexo e demorado podendo demorar vários dias a treinar.

Fornecemos a CNN uma base de dados conhecida, e informamos quais as classificações de cada imagem, a CNN ira então realizar um feed-forward, e calculara o erro obtido, depois a CNN usando gradiente descendente ira tentar calcular o ponto da função com o menor valor usando gradiente descendente.

Este gradiente descendente pode ser afinado a partir da taxa de aprendizagem que podemos deferir manualmente, esta taxa é extremamente importante, se o valor for baixo, iremos demorar muito tempo a encontrar o valor mínimo.

Se for muito elevado pode causar um efeito de pêndulo, podendo até chegar a um erro superior ao inicial, ao detectar isto, teríamos de ajustar o valor de n, para um valor inferior.

Figura 4.2: Exemplo de um n elevado



$https://www.neural-networks.io/fr/single-layer/gradient-descent.php?fbclid=IwAR14yLerF3UFedq1qKOPDYtVYPCvM9cV359inHI_E6Ed6yNw7Mc16XgEFgw$

Imagine que o ponto vermelho é uma bola que é atirada por uma colina, a cada passo fica com um valor de y superior ao anterior, com este n, iríamos ficar presos nesta zona e seria muito difícil ou demorado encontrar o ponto mínimo desta função.

Estes cálculos tem de ser feitos para cada node, e para cada camada, podemos ver que iremos ter que fazer muitos cálculos, para isso é usamos o Graphic Processing Unit (GPU), pela sua capacidade de processamento paralelo, também é possível usar o Central Processing Unit (CPU) para estes cálculos, contudo é menos eficiente e demoraria muito mais tempo.

De forma a tornar o treino mais rápido, a rede é treinada usando lotes, o tamanho destes lotes podem ser defendidos manualmente, e são bastante dependentes da memoria disponível no computador, um valor elevado pode provocar um erro por falta de recursos.

Um epoch é quando a rede passa por todos os elementos, apenas uma vez.

O tamanho de lote, e números de epoch são defendidos manualmente, um valor elevado significa que vamos passar mais vezes a mesma informação pela rede, isto aumenta a sua precisão, contudo quando a rede atinge a sua precisão máxima, fazendo mais epochs não fará com que a rede melhore, ficando assim estagnada naquele intervalo, assim temos de escolher um numero equilibrado que permita o bom treino e que não torne o tempo de treino desnecessariamente elevado.

Num projecto disponível no GitHub e realizado por Tiago Oliveira no âmbito de um trabalho realizado para um curso online, que tem a função de classificação de imagens foi usado uma colecção de informação com 13GB, usando um GPU GeForce 980Ti este modelo demorou cerca de 12h a completar.

Foi usado um lote de 256, e 3 epochs, isto significa que a rede percorreu toda a base de dados 3 vezes, e que por cada inteneração processou 256 imagens.

Figura 4.3: Classificação de imagem

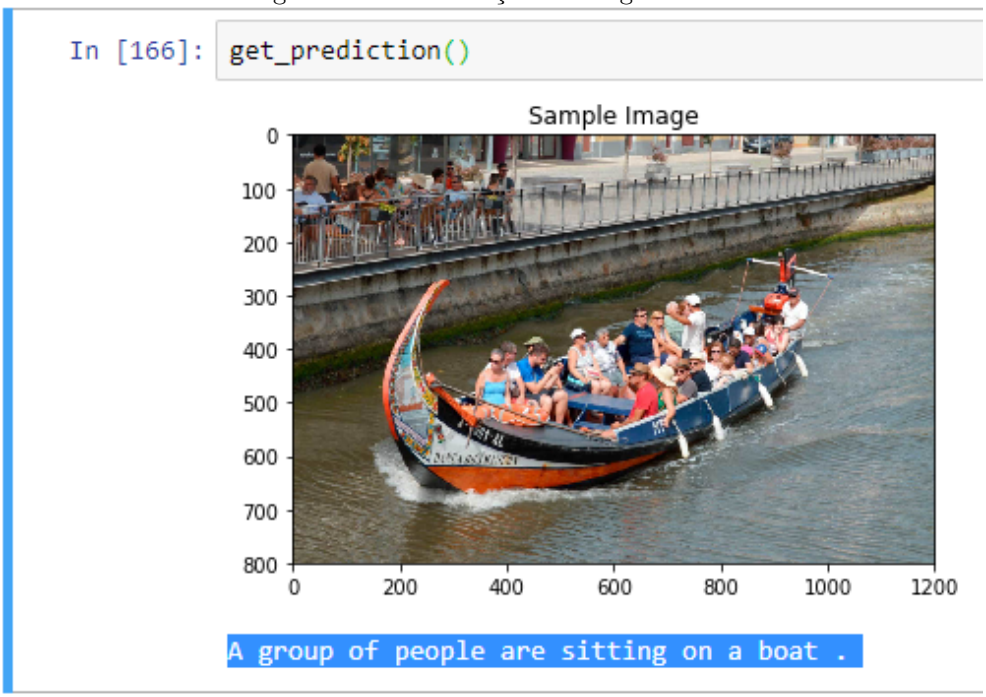


Foto do barco: *https : //www.civitatis.com/en/aveiro/moliceiro − boat − trip/?fbclid = IwAR2vvWp30m24cNiCKQjrXOjQZ − 3bZmsehdW − MftSUA642WlFmcj1b3uycKE*
Projecto GitHub: *https : //github.com/tiagooliveira95/Computer − Vision − Nanodegree − Projects/tree/master/P2 − ImageCaptioning*

Podemos ver que a CNN consegui-o classificar correctamente a imagem acima, dando nos a legenda que a imagem se trata de um grupo de pessoas sentadas num barco.

Esta ferramenta é bastante útil, podemos treinar uma rede para detectar vários tipos de cancro, para detectar edifícios antigos fornecendo nos um link com toda a informação, para detectar texto numa imagem e traduzir para diferentes línguas em tempo real e até podíamos usar um drone para tirar fotos a um campo agriculta para detectar zonas que com pouca hidratação.

As aplicações desta área são infinitas.

Capítulo 5

Rede Neural Recorrente

Uma rede neural recorrente, ou Recurrent Neural Networks (RNN) é idêntica com uma CNN funcionam lado a lado, mas uma RNN toma em atenção informação sequencial.

A Google usa uma RNN para prever a nossa pesquisa, a medida que vamos introduzindo caracteres, a rede neural vai tentar prever qual será o carácter seguinte mais provável tendo em conta os caracteres inseridos pelo utilizador, como a Google passou anos a treinar este modelo, hoje em dia, por vezes basta introduzirmos uma simples letra para a Google detectar as pesquisas mais prováveis.

Uma CNN apenas usa as suas entradas para determinar a saída, uma RNN usa as entradas e informações passadas para determinar a saída.

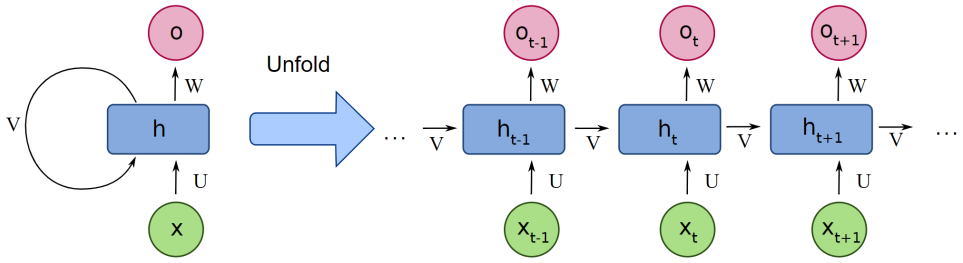
Ao classificarmos uma imagem de uma pessoa, para sabermos se ela esta parada ou a andar necessitaríamos de mais que uma simples imagem, ao fornecer vários frames a uma RNN esta ira conseguir ver que a posição das pernas muda de frame para frame, dependendo do movimento detectado a RNN vai conseguir determinar se a pessoa esta parada, a andar ou a correr.

As RNN têm bastante utilidade, desde prever qual o filme ou vídeo recomendar, padrões de tráfego automóvel e até assistentes virtuais como o Assistente da Google, Seri da Apple, e Alexa da Amazon.

Combinando as duas tecnologias, podemos usar uma CNN para extrair os aspectos mais importantes de uma imagem e usar RNN nos locais onde necessitamos de um contexto.

Uma RNN é geralmente representada assim:

Figura 5.1: Rede RNN



*https : //medium.com/deeplearningbrasilia/deep − learning − recurrent − neural − networks − f*9482*a*24*d*010

Podendo esta ter vários níveis empilhados como peças de legos.

Os valores de x e o representam os valores de entrada e de saída respectivamente.

Os valores W, U e V são pesos, estes são os pesos que são calibrados durante durante o processo de treino designado por back-propagation.

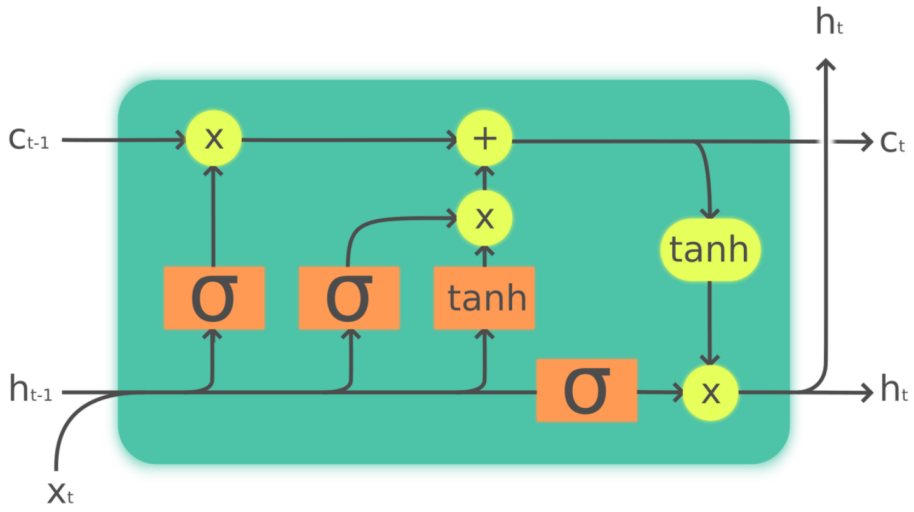
O bloco h, representa uma função, esta função pode ser escolhida manualmente desde Time delay neural network (TDNN) a ELMAN.

Estas foram as primeiras tentativas de adicionar memoria, contudo em meados de 1990 foi descoberto que estas funções todas sofriam de um problema, não conseguiam se lembrar de eventos que ocorreram a muito tempo, pois a cada itineração o passo dessa memoria fica cada vez mais pequeno, tornando o menos e menos relevante.

Foi então em 1997 as Long Short-Term Memory (LSTM) foram introduzidas por Sepp Horchreirter e Jürgen Schmidhuber para combater este problema, que introduz secções de memoria fixa que podem ser utilizados após um longo período de tempo.

Exemplo de uma célula LSTM

Figura 5.2: Célula LSTM



https : //medium.com/deeplearningbrasilia/deep − learning − recurrent − neural − networks − f9482a24d010

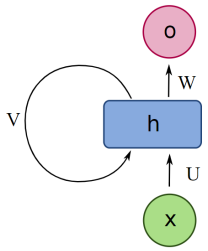
Em primeiro lugar calcula se a função sigmóide, seguida da função tangente hiperbólica e da multiplicação e soma.

A função sigmóide tem valores de 0 a 1 e funciona como uma porta, estas 3 portas decidem o que entra na célula, o que fica, e o que sai da célula, estas portas são treinadas a partir dos pesos de entrada que são defendidos quando treinamos o modelo.

As saídas são então passadas para outras células formando uma rede de neurónios ou nodes.

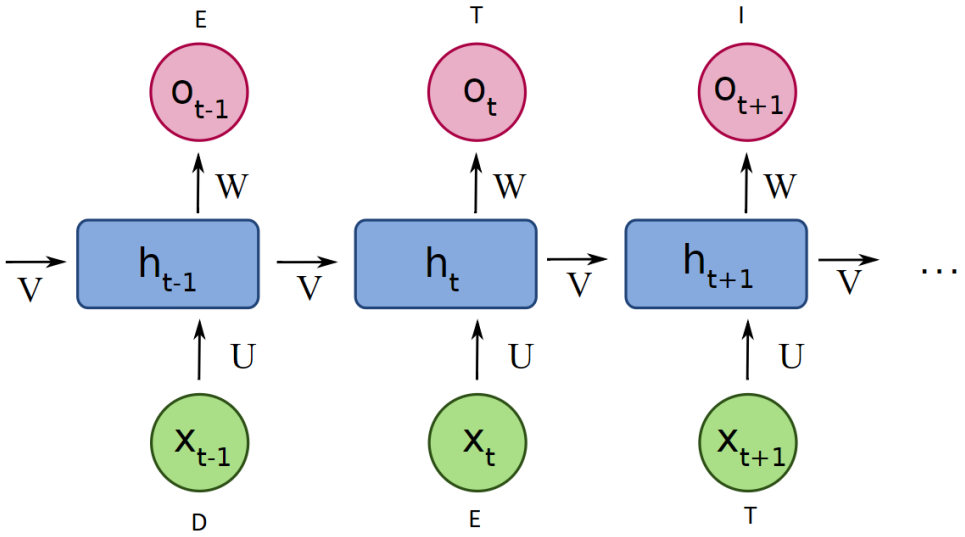
Vamos dar um exemplo de como a Google consegue prever o que estamos a escrever.

Olhemos para este exemplo:



Ao introduzir a letra D, queremos obter a letra I na saída, então treinamos este modelo a partir de uma basta colecção de pesquisas.

Aqui podemos ver um exemplo de uma rede, nas entradas inserimos os caracteres nas saídas obtemos uma lista de distribuição que contem os caracteres mais prováveis.



Neste caso foi inserido um D, e obtivemos um I de saída, este I pode ser enviado para cima e voltaria a passar por um node ou neurónio, em que obtinhamos a letra T, fazendo isto 4 vezes obteríamos a palavra DETI.

A mediada que vamos caminhando para a direita, vão sendo inseridos mais caracteres, quantos mais forem introduzidos a probabilidade da pesquisa intencional ser mesmo DETI ira crescer exponencialmente.

Isto acaba por ser um exemplo simplificado, pois a partir da letra D podemos derivar muitas outras palavras que começam por D, mas a mediada que o utilizador insere caracteres será possível deleitar a serie de frases mais prováveis.

Uma RNN não serve apenas para prever a próxima letra, é também utilizada em carros autónomos para prever o que os outros condutores vão fazer a seguir, isto ajuda a evitar acidentes, e usado também para ajudar pessoas com problemas de saúde, como por exemplo traduzir linguagem gestual para oral, e vice versa, assim uma pessoa que não saiba linguagem gestual pode facilmente comunicar com uma pessoa com problemas auditivos sem necessitar de aprender linguagem gestual, pois existem RNN capasses de fazer a tradução em tempo real, para que ambos consigam comunicar.

Capítulo 6

Conclusões

Portanto, a partir da análise de artigos científicos e outras fontes, é possível concluir que as tecnologias de Deep Leaning e Visão Computacional são ferramentas úteis que provem a performance na análise e classificação de dados. Promovendo assim, avanços científicos mais rápidos e eficazes.

Apesar de ser possível treinar modelos em qualquer CPU ou GPU, para esta tecnologia ser usada a nível profissional, seria necessário equipamentos apropriados, pois os GPU não estão desenhados para estes tipos de cálculos e demoram muito tempo a processar os dados, deste modo, equipas profissionais para treinar modelos de forma eficiente necessitam de tecnologias caras podendo estas ultrapassar os 10mil EUR por cada GPU para alem de uma uma grande quantidade de dados para que uma análise pertinente seja feita.

Logo, conclui-se que DL e VC são tecnologias adequadas para finalidades que possuam os recursos financeiros e informacionais significativos. Mas, espera se que com sua crescente expansão estas tecnologias se tornem cada vez mais acessíveis.

Capítulo 7

Contribuição dos autores

Este relatório foi desenvolvido por dois autores, Nicole Rakov (96661) e Tiago Oliveira (99064). Composto por 7 secções principais, a contribuição dos autores por cada secção é dada por:

	Nicole Rakov	Tiago Oliveira
Resumo	100%	0%
Introdução	100%	0%
Análise	100%	0%
Visão Computacional	0%	100%
Redes neurais convolucionais	0%	100%
Rede neural recorrente	0%	100%
Conclusão	70%	30%

Também pertinente mencionar que a correcção e revisão ortográfica foi feita por Tiago Oliveira, referencias bibliográfica por Nicole Rakov. Os acrónimo e a revisão do código por ambos os autores.

Acrónimos

UA Universidade de Aveiro

CNN Convolutional Neural Networks

RNN Recurrent Neural Networks

RGB Red, Green, Blue

ReLU Rectified Linear Unit

GPU Graphic Processing Unit

CPU Central Processing Unit

DL Deep Learning

ML Machine Learning

VC Visão Computacional

LSTM Long Short-Term Memory

TDNN Time delay neural network

DNN Deep Neural Networks

Bibliografia

[Ali+16] Alexander Aliper et al. "Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data." eng. Em: *Molecular pharmaceutics* 13.7 (jul. de 2016), pp. 2524–2530. ISSN: 1543-8392 (Electronic). DOI: 10.1021/acs.molpharmaceut.6b00248.

[AM18] N. Akhtar e A. Mian. "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey". Em: *IEEE Access* 6 (2018), pp. 14410–14430. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2807385.

[Bar+17] Pierre Barré et al. "LeafNet: A computer vision system for automatic plant species identification". Em: *Ecological Informatics* 40 (2017), pp. 50–56. ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2017.05.005. URL: http://www.sciencedirect.com/science/article/pii/S1574954116302515.

[Che+18] Hongming Chen et al. "The rise of deep learning in drug discovery". Em: *Drug Discovery Today* 23.6 (2018), pp. 1241–1250. ISSN: 1359-6446. DOI: https://doi.org/10.1016/j.drudis.2018.01.039. URL: http://www.sciencedirect.com/science/article/pii/S1359644617303598.

[Est+17] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". Em: *Nature* 542.7639 (2017), pp. 115–118. ISSN: 1476-4687. DOI: 10.1038/nature21056. URL: https://doi.org/10.1038/nature21056.

[HPW17] J. B. Heaton, N. G. Polson e J. H. Witte. "Deep learning for finance: deep portfolios". Em: *Applied Stochastic Models in Business and Industry* 33.1 (2017), pp. 3–12. DOI: 10.1002/asmb.2209. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asmb.2209. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.2209.

[Ker+18] J. Ker et al. "Deep Learning Applications in Medical Image Analysis". Em: *IEEE Access* 6 (2018), pp. 9375–9389. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2788044.

[KP18] Andreas Kamilaris e Francesc X. Prenafeta-Boldú. "Deep learning in agriculture: A survey". Em: *Computers and Electronics in Agriculture* 147 (2018), pp. 70–90. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2018.02.016. URL: http://www.sciencedirect.com/science/article/pii/S0168169917308803.

[LBH15] Yann LeCun, Yoshua Bengio e Geoffrey Hinton. "Deep learning". Em: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[Vou+18] Athanasios Voulodimos et al. "Deep Learning for Computer Vision: A Brief Review". Em: *Computational Intelligence and Neuroscience* 2018 (2018), p. 13. DOI: 10.1155/2018/7068349.

[Wei18] Ben G. Weinstein. "A computer vision for animal ecology". Em: *Journal of Animal Ecology* 87.3 (2018), pp. 533–545. DOI: 10.1111/1365-2656.12780. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/1365-2656.12780. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/1365-2656.12780.