UNIVERSIDADE ESTADUAL DE MARINGÁ
DEPARTAMENTO DE INFORMÁTICA
CURSO DE INFORMÁTICA

RAFAEL BIAZUS MANGOLIN

CLASSIFICAÇÃO DE IMAGEM DE COMIDA COM REDES NEURAIS CONVOLUCIONAIS

TRABALHO DE CONCLUSÃO DE CURSO

MARINGÁ
2017

RAFAEL BIAZUS MANGOLIN

CLASSIFICAÇÃO DE IMAGEM DE COMIDA COM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao curso de Informática da Universidade Estadual de Maringá, como requisito parcial para a obtenção do título de Bacharel em Informática.

Orientador: Yandre Maldonado e Gomes da Costa
Universidade Estadual de Maringá

MARINGÁ
2017

AGRADECIMENTOS

A minha família pelo incentivo e apoio incondicional.

Aos meus amigos de graduação, pelas trocas de ideias, criticas e auxílios.

Ao meu orientador Dr. Yandre Maldonado e Gomes da Costa, por me apresentar a área de sistemas inteligentes e pelo auxilio e sugestões no desenvolvimento deste trabalho.

A todos que direta ou indiretamente fizeram parte da minha formação.

RESUMO

MANGOLIN, Rafael. Classificação de imagem de comida com redes neurais convolucionais. 2017. 22 f. Trabalho de Conclusão de Curso – curso de Informática, Universidade Estadual de Maringá. Maringá, 2017.

Esta monografia aborda o problema de classificação de imagens de comida, que está inserido na área de reconhecimento de padrões. Rede neural convolucional (*convolutional neural network*, CNN), foi utilizada para realizar a tarefa de classificação. As CNNs vem melhorando o estado da arte na área de classificação de imagens desde 2012. As CNNs são técnicas de *deep leaning*, que utilizam por meio de redes neurais de muitas camadas para realizar os processos de aprendizado e classificação. As CNNs apresentam um problema recorrente no processo de classificação, o *overfitting*, esse problema ocorre geralmente devido a falta de amostras para a realização do treino da rede neural. Para a solução desse problema neste trabalho foi proposto duas técnicas: o *data augmentation* e a inicialização dos pesos da rede a partir de uma rede treinada. A rede neural proposta neste projeto tem como base de estrutura a CNN *AlexNet*. A base de dados utilizada foi formulada com amostras retiradas das bases *ImageNet* e *Food-101*, contendo 16 classes de imagens. O melhor resultado foi obtido quando aplicada as técnicas propostas.
Palavras-chave: Classificação de imagens de comida. Rede neural convolucional. *Deep learing*.

ABSTRACT

MANGOLIN, Rafael. Food image classification with convolutional neural networks. 2017. 22 f. Trabalho de Conclusão de Curso – curso de Informática, Universidade Estadual de Maringá. Maringá, 2017.

This monograph approaches the problem of food image classification, which is inserted in the area of pattern recognition. Convolutional neural network (CNN), was used to perform the classification task. The CNNs has been improving the state of the art in the area of image classification since 2012. CNNs are deep learing techniques, that use through multi-layers neural networks to accomplish the processes of learning and classification. CNNs present a recurring problem in the classification process, the overfitting, this problem usually occurs due to the lack of samples for performing neural network training. To solve this problem in this work two techniques have been proposed: the data augmentation and the initialization of network weights from a trained network. The proposed neural network in this project is based on CNN AlexNet. The database used was formulated with samples taken from the bases ImageNet and Food-101, containing 16 classes of images. The best result was obtained when applying the proposed techniques.
Keywords: Food image classification. Convolutional neural network. Deep learing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação

LISTA DE SÍMBOLOS

Γ	Letra grega Gama
λ	Comprimento de onda
$\in$	Pertence

SUMÁRIO

1 INTRODUÇÃO

O problema de classificação de imagens utilizando técnicas tradicionais de aprendizado de máquina requer conhecimento e experiência no domínio que será classificado, para extrair características (do inglês *features*) relevantes, que são utilizadas para a obter a classificação. Assim, utilizando técnicas que não dependem de extratores especializados de *features*, como redes neurais convolucionais (CNN, *convolutional neural networks*), se torna mais fácil desenvolver modelos eficazes de aprendizado de máquina para novos conjuntos de dados. Nesse projeto foi utilizada CNN para a classificação de imagens com comida, assim não será preciso encontrar descritores especializados nesse domínio.

Segundo (LECUN; BENGIO; HINTON, 2015), o *deep learning* vem sendo utilizado para resolver problemas computacionalmente complexos que temos no nosso dia-a-dia, e seu uso vem evoluindo o estado-da-arte de muitas áreas. Este projeto foi aplicado na área de reconhecimento de imagem, para classificação de imagens.

Neste projeto utilizamos de *deep learning*, mais especificamente, de rede neural convolucional, para fazer o reconhecimento e classificação de imagens de comida, visando reconhecer o tipo de comida descrito na imagem, a partir de um conjunto de tipos previamente definidos.

Na seção de revisão de literatura desse projeto são descritos conceitos que auxiliam no seu desenvolvimento, sendo dividida em dois tópicos principais: Redes Neurais e *Deep learning*. Na seção de metodologia descreve como foi descrito o projeto, e as etapas utilizadas na sua aplicação, é composta por três tópicos principais: a definição e organização da base de dados; a arquitetura da rede neural proposta; e técnicas para melhorar o poder de classificação da rede neural.

A seção de resultados contém uma avaliação dos resultados obtidos, contendo uma comparação entre os resultados e avaliando a melhora na acurácia e a redução do *overfitting*, conforme as técnicas eram adicionadas ao modelo. Na seção de conclusão é apresentada as contribuições deste trabalho na área e proposto projetos visando a melhora de resultado da rede.

2 REVISÃO DE LITERATURA

Nesta seção, será descrita a revisão de literatura sobre redes neurais, *deep learning* e redes neurais convolucionais.

2.1 Rede neural

Rede neural artificial pode ser definida como sendo um conjunto interconectado de elementos básicos de processamento (GURNEY, 1997). Seu funcionamento é inspirado na capacidade de aprendizado do cérebro animal, que possui uma imensa estrutura com capacidade de definir regras a partir de experiências, que vão ocorrendo durante a vida, gerando ligações físicas (sinapses) mais fortes, aprimorando assim o que foi aprendido (HAYKIN, 2001).

> Uma rede neural é um processador maciçamente paralelo e distribuído, constituído de unidades de processamento simples, que tem a propensão natural para armazenar conhecimento experimental e torna-ló disponível para uso. Ela se assemelha ao cérebro em dois aspectos: o conhecimento é adquirido pela rede a partir de seu ambiente por meio de um processo de aprendizagem; forças de conexão entre neurônios, definidos como pesos sinápticos, são utilizados para armazenar o conhecimento adquirido (HAYKIN, 2001).

Dadas essas características, as redes neurais vem sendo amplamente utilizadas para resolução de problemas que não possuem uma resolução trivial, como a classificação de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), a identificação de câncer de pele (ESTEVA et al., 2017), tomada de decisão no mercado de ações (GAMBOGI, 2013), e entre outras áreas. Essa versatilidade de áreas em que é utilizada ocorre devido a sua generalidade na maneira de encontrar pontos no problema que devem ter mais destaque, características mais relevantes que são identificada pela rede durante seu processo de aprendizagem, sendo reavaliadas pela própria rede em cada instância testada. Como dito por Zhang (2000), as redes neurais possuem a habilidade de se adaptar aos dados para realizar as classificações sem a necessidade de apontar explicitamente o que deve ser observado no modelo.

Segundo Kriesel (2007) uma rede neural pode ser descrita por três elementos:

- Unidades simples de processamento, ou neurônios.
- Elos de conexão entre os neurônios (sinapses).
- A importância entre a conexão de um neurônio com outro, descrita por uma função de peso *w*.

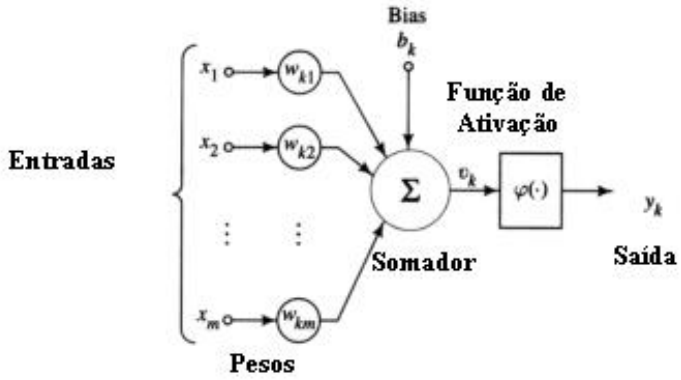
Uma rede neural pode ser descrita matematicamente pela tríplice (N,V,w), onde N é um conjunto de neurônios, V é um conjunto de conexões entre os neurônios definida por $V = \{(i,j) \mid i,j \in N\}$ e w é a função que determina o peso das conexões definida por $w : V \implies \mathbb{R}$, descrita em $w(i,j)$.

2.1.1 Neurônio

HAYKIN (2001) define o neurônio como a unidade de processamento de informação que é primordial para o funcionamento de uma rede neural artificial. É nele que ocorre o processamento das entradas, e o redirecionamento da saída, indicando onde irá influenciar tal processamento.

Na Figura 1 é possível identificar os três elementos fundamentais de um neurônio:

Figura 1 – Modelo de um neurônio, contendo as entradas, funções de peso, função somadora, o *bias*, função de ativação da saída.



Fonte: imagem retirada do *Google*[1]

- O fluxo de entrada dos dados, sendo ele um conjunto de conexões que serão sujeitas a função de peso, para ser feito o uso na função somadora (HAYKIN, 2001). As conexões podem se originar tanto de uma entrada de dados na rede, quanto de neurônios que estão localizados em camadas superiores.
- A função somadora é responsável por realizar o processamento do fluxo de entrada. Um exemplo desse tipo de função é a soma dos pesos (KRIESEL, 2007), a função realiza a multiplicação do peso w_{kj} com a entrada x_j, e depois realiza a soma das m entradas do neurônio representada pela função matemática:

$$u_k = sum_{j=1}^{m} w_{kj} x_j$$

  O resultado dessa etapa é propagado pela rede dados os critérios da função de ativação.
- A função de ativação é responsável por restringir a abrangência do dado gerado pelo processamento do neurônio. Foi identificado por HAYKIN (2001) três tipos básicos de função de ativação sendo elas:
  - **Função de limiar**:

$$f(x) = \begin{cases} 1 & se\ x \geq 0 \\ -1 & se\ x < 0 \end{cases}$$

[1] http://www.gsigma.ufsc.br/popov/aulas/rna/neuronio_artificial/neuronio_artificial.jpg, acessada em 02 de junho de 2017, ás 18:53

É conhecido na literatura como função de *Heaviside*, definindo a saída de maneira binaria.

– **Função linear por partes**:

$$f(x) = \begin{cases} 1 & se\ x \geq 1 \\ x & se\ 0 \leq x < 1 \\ 0 & se\ x < 0 \end{cases}$$

Esse tipo de função pode ser analisada como uma tentativa de simulação de um amplificador não linear, tendo sua área variável e seus pontos de saturação.

– **Função Sigmoide**:

$$f(x) = \frac{1}{1 + exp(-av)}$$

Esse tipo de função de ativação é o mais utilizado na construção de redes neurais artificiais (HAYKIN, 2001). É definida por uma função crescente não linear, quando seu parâmetro de curva se aproxima do infinito, apresenta comportamento semelhante a funções de limiar.

O modelo neural da figura acima também inclui um *bias* (b_k) aplicado externamente. Tem como função aumentar ou diminuir a entrada mínima de dados. Defini um limiar no neurônio e pode ser utilizado para o cálculo da função de ativação.

Dado as iteração, a *bias* e os pesos da entrada podem ser modificados pelo processo de aprendizagem, aprimorando sua resposta conforme a rede é treinada.

2.1.2 Processos de aprendizagem

A habilidade que se destaca de uma rede neural é a aprendizagem, adaptando-se aos dados que estão em seu ambiente, para melhorar o seu desempenho. Essa habilidade vem do processo de aprendizagem da rede neural artificial, que é definido por Demuth et al. (2014) como o procedimento de ajuste das funções de pesos e dos *bias* dos neurônios da rede, acontecendo na etapa de *"treino"* da rede e tem como objetivo preparar a rede para executar uma tarefa.

Podemos dividir o processo de aprendizagem em três categorias principais sendo elas:

- **Aprendizado supervisionado:** método no qual uma parte da base é utilizada para treinar a rede. Assim após cada processamento é verificado o resultado da classificação, e se necessário são feitas correções nos pesos dos neurônios que influenciaram esse resultado, para assim reforçar uma classificação boa ou corrigir uma classificação ruim.
- **Aprendizado por reforço:** método similar ao aprendizado supervisionado, tendo como diferença a forma de avaliação. Como dito por Kaelbling, Littman e Moore (1996), a principal diferença entre aprendizagem supervisionada e a aprendizagem por reforço, é que o aprendizado por reforço não apresenta conjuntos de saídas corretos e errados, e após cada ação é aplicado uma taxa de correção e indicado os estados seguintes, mas não é informado qual escolha teria sido a melhor para o caso.

- **Aprendizado não-supervisionado:** método no qual não existe um avaliador ou dados pré-definido informando a classe da entrada, a própria rede é responsável por agrupar os dados, os ajustes dos pesos e dos *bias* é feito apartir das entradas. A rede basicamente aprende como categorizar a entrada de dados em uma quantidade finita de classes.

2.1.3 *Perceptron*

O *perceptron* é tido como a forma mais simples de rede neural para classificar duas classes que são linearmente separáveis (HAYKIN, 2001). Como essa rede é composta por um único neurônio com pesos de conexões e *bias* ajustáveis está limitado a classificar a entrada apenas em duas classes.

A rede é inicializada com pesos aleatórios, e após a execução de cada entrada sua saída é comparada com o resultado esperado, obtendo assim um sinal de erro, que é utilizado para fazer ajustes nos pesos. Como ocorre nos processos de aprendizado supervisionado.

Uma generalização do *perceptron*, é o perceptron de múltiplas camadas (no inglês *multiple layer perceptron*, MLP). Onde o MLP é configurado em no mínimo três camadas, onde a primeira delas é a camada de entrada, em que ocorrem a entrada dos dados na rede, e a última é a camada de saída onde está contida a classificação da entrada. As camadas intermediárias tem a função de analisar características mais complexas da entrada, dando possibilidade de uma melhor classificação.

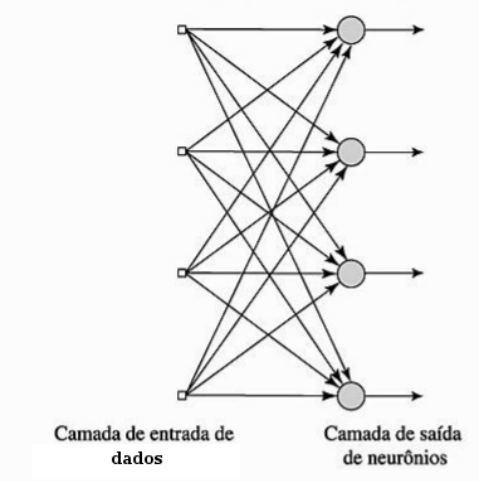
O método de aprendizagem utilizado pelo *perceptron* de múltiplas camadas é conhecido como *error backpropagation* (algorítimo de retropropagação de erro) (HAYKIN, 2001). Para isso, utiliza do método de aprendizado supervisionado de correção por erro. Quando é identificada a necessidade de ajuste nos pesos, ocorre uma retropropagação nos neurônios que influenciaram a classificação, ajustando seus pesos e *bias*.

2.1.4 Tipos de redes

Para problemas mais complexos, redes com apenas um neurônio tendem a não resolvê-los. Geralmente é necessário ter vários deles trabalhando em paralelo (uma camada de neurônios) (DEMUTH et al., 2014). (HAYKIN, 2001) descreve três classes de arquitetura de rede que geralmente são encontradas:

- **Redes alimentadas adiante de uma camada:** essa classe é a forma mais simples de rede em camada, na qual se tem uma camada de dados e uma camada de neurônios (camada de processamento), que também é a camada de saída, como na Figura 2. A camada de entrada de dados não é contada, pois nela não ocorre processamento.

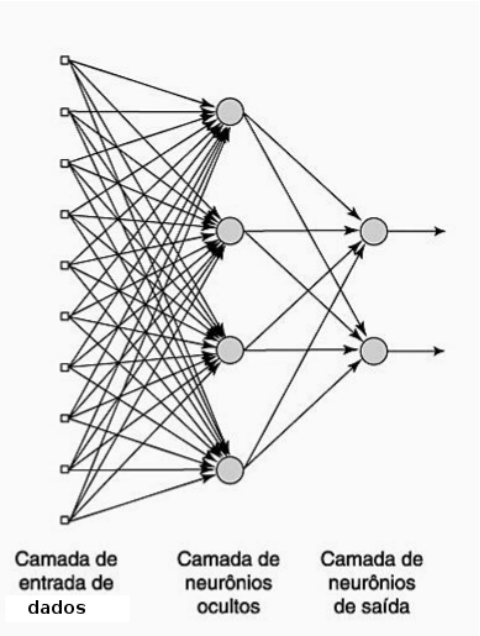
Figura 2 – Exemplo de rede alimentada adiante de uma camada.



Fonte: (HAYKIN, 2001)

- **Redes alimentadas diretamente com múltiplas camadas:** essa classe de rede, é também alimentada adiante, mas possui uma ou mais camadas ocultas. As camadas ocultas estão localizadas entre a camada de entrada de dados e a camada de saída. Ao adicionar camadas ocultas na rede é possível ter acesso a características mais específicas da entrada, melhorando o resultado da rede.
  Como representado na Figura 3, nesse modelo cada camada só fornece dados à camada posterior, e reciprocamente, só recebe dados da camada anterior. Exemplificando, a camada de entrada recebe os dados e formata a saída para a entrada da camada seguinte, a primeira camada oculta processa os dados fornecidos pela camada de entrada e o formata para a camada seguinte. Esse processo continua até chegar na camada de saída, conhecida também como camada final, a saída produzida por essa camada contém a resposta global produzida pela rede para a entrada fornecida na camada inicial.

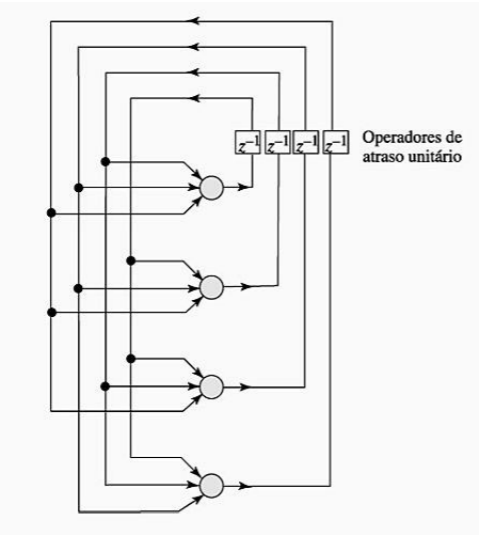
Figura 3 – Exemplo de rede alimentada diretamente de múltiplas camadas.



Fonte: (HAYKIN, 2001)

- **Redes recorrentes:** essa classe de arquitetura se diferencia das anteriores pelo fato de possuir pelo menos uma camada com realimentação, ou seja, a saída da camada serve de entrada para a mesma, exemplo Figura 4. Os operadores de atraso unitário, representados na imagem pelo símbolo z^{-1}, são aplicados nas conexões de realimentação modificando de maneira dinâmica e não linear os valores informados. É dito que a rede possui uma auto-realimentação quando a saída de um neurônio realimenta a sua entrada.

Figura 4 – Exemplo de rede retroalimentada.



Fonte: (HAYKIN, 2001)

2.2 *Deep learning*

Deep learning pode ser definido como uma hierarquia de "*conceitos*"de aprendizagem, em que "*conceitos*"complexos se originam de grupos formados por "*conceitos*"mais simples. Se representar esses "*conceitos*"em um grafo, é possível ver como um "*conceito*"é montado baseado no outro, como se possuíssem muitas camadas (GOODFELLOW; BENGIO; COURVILLE, 2016).

> *Deep learning* permite que modelos computacionais compostos de múltiplas camadas de processamento aprendam representações de dados com múltiplos níveis de abstração. O *deep learning* descobre estruturas complexas em vastos conjuntos de dados com o uso do algoritmo de retropropagação para indicar como a máquina deve mudar seus parâmetros internos que são utilizados para computar a representação resultante da camada anterior em cada camada (LECUN; BENGIO; HINTON, 2015).

Bengio, Courville e Vincent (2013) categoriza o *deep learning* como um método de aprendizagem de representação (do inglês, *representation learning*). Métodos ditos como *representation learning* são capazes receber dados sem tratamento como entrada e a partir de processamentos internos encontrar automaticamente características relevantes para realizar a classificação. Na qual, para o *deep learning*, cada camada oculta de processamento produz uma nova e avalia um nova representação, podendo ser descrito como uma método de aprendizado de multi representações. Dessa maneira a entrada pode ser uma imagem, descrita em um mapa de bits, na qual a primeira camada analisaria informações mais superficiais, como contornos ou formas em certas áreas das imagens. Já na segunda camada seriam identificados padrões avaliando certas disposições de bordas ou formas em partes da imagem, ignorando pequenas variações. E na terceira camada seria identificado padrões que se assemelham a parte de objetos conhecidos, e nas camadas posteriores seriam mais padrões até chegar ao ponto de realizar a classificação dos objetos contidos na imagem (LECUN; BENGIO; HINTON, 2015).

2.2.1 Redes neurais convolucionais

Como descrito por LeCun et al. (1989) redes neurais convolucionais são um tipo especializado de rede neural para processamento de dados que se organizam em grade (ou matriz), tendo como um exemplo de entrada uma imagem, uma matriz de *bits*.

Elas possuem o nome de rede convolucional, pois em algumas de suas camadas ocultas ela possui uma camada de convolução. Outro tipo de camada muito utilizada nessas redes é a camada de *pooling* (GOODFELLOW; BENGIO; COURVILLE, 2016).

Uma camada de uma rede neural convolucional geralmente é composta de três fases: a primeira fase onde é aplicada diversas convoluções em paralelo na mesma imagem gerando um conjunto de ativações lineares; a segunda fase propõem a aplicação de uma função de ativação não linear, sendo a unidade linear de correção (do inglês *rectified linear unit*, ReLU) muito utilizada atualmente (LECUN; BENGIO; HINTON, 2015); e no terceiro e ultimo estágio

é utilizado uma função de *pooling* para modificar o dado que será fornecido para a próxima camada.

2.2.1.1 Operação de convolução

Camadas de convolução são baseadas essencialmente na operação de convolução. Segundo (GOODFELLOW; BENGIO; COURVILLE, 2016), a operação de convolução é descrita por uma operação que ocorre entre duas funções, podendo ser descrita da seguinte maneira:

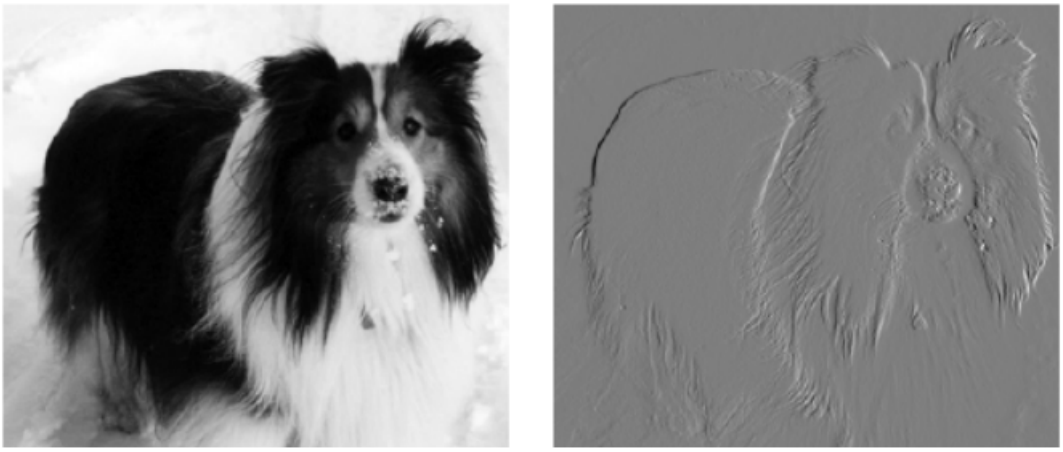
$$s(t) = (x * w)(t)$$

Em redes convolucionais os argumentos da função de convolução são geralmente compostos pela a entrada de dado (x) e o *kernel* utilizado para a modificação (w). Sua saída é um mapa de características.

Assim a entrada de dados normalmente é uma matriz de dados, no caso uma imagem. O *kernel* utilizado também costuma ser uma matriz de parâmetros que podem ser ajustados pelo processo de aprendizagem (retropropagação). E como saída, cria uma matriz da dados com algumas características ressaltadas.

A operação de convolução é uma maneira eficiente de descrever transformações para serem aplicadas em áreas menores mantendo a linearidade, em todo o dado de entrada. Como levantado por Goodfellow, Bengio e Courville (2016), para realizar uma operação de subtração entre os *pixels* de uma imagem, para serem encontradas as bordas contidas na imagem como visto na Figura 5, é necessário uma quantidade muito menor de computação para obter o resultado desejado.

Figura 5 – Exemplo da aplicação de operações para encontrar as bordas verticais de uma imagem. A esquerda a imagem normal em escala cinza e a direita a imagem aplicada a operação subtração dos *pixeis* vizinhos.



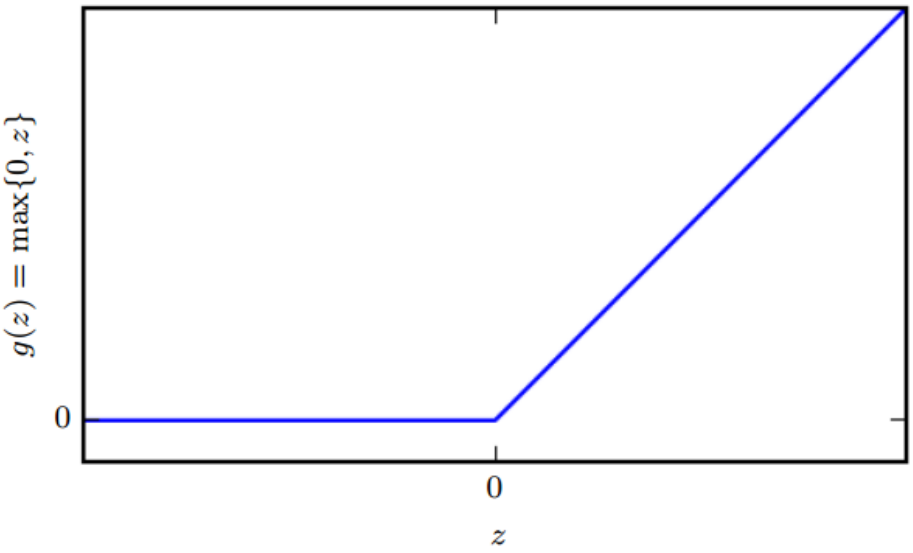
Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

2.2.1.2 ReLu

A função de ativação linear (ReLu), vem sendo recomendada a ser utilizada em redes neurais alimentadas adiante (GLOROT; BORDES; BENGIO, 2011). Como visto na Figura 6, a ReLu se mantém muito próxima de uma função linear. Funções ReLus apresentam características que as possibilitam aplicarem as otimizações utilizadas em funções lineares.

Sua função é descrita por pela equação $g(z) = max\{0,z\}$, onde o z no contexto de CNN é a taxa de correção, calculada pelas saídas e as *bias*.

Figura 6 – Gráfico da função de ativação não linear ReLu.



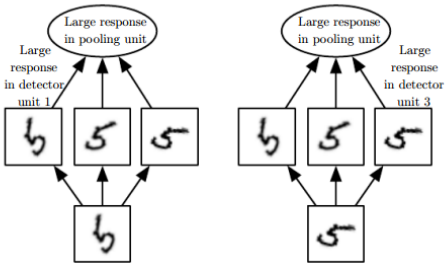
Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

2.2.1.3 Operação de *pooling*

A aplicação da função de *pooling* para modificar o dado de entrada, ajuda a tornar o modelo classificador adaptado a pequenas translações da entrada (GOODFELLOW; BENGIO; COURVILLE, 2016). Dessa forma a aplicação dessa operação permite identificar se o objeto está contido na imagem independente do local que aparece e da inclinação que apresenta. Essa característica se torna muito eficaz para ser aplicada em redes que necessitam dessa variabilidade de padrões de posição para uma mesma classe.

Exemplificando, em uma carta a aplicação da operação de *pooling* permite a rede identificar os números do código postal que estão escrito a mão na carta, mesmo que estes não estejam localizados no mesmo local, ou inclinação de cada respectiva imagem, como exemplo na Figura 7.

Figura 7 – Exemplo de como é feito a ativação de um neurônio na camada de *pooling*.



Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

3 METODOLOGIA

Nessa seção, são descritas as etapas do desenvolvimento dessa monografia, sendo elas a formulação e pré-processamento da base de dados, a configuração da rede neural e sua codificação utilizando o *framework keras* (CHOLLET et al., 2015), e pesquisas e aplicações de melhorias para a rede neural proposta como o *data augmentation* e a inicialização dos pesos a partir de uma rede já treinada.

3.1 Formulação da base

No aprendizado de máquina, a base de dados em que serão feitos os treinos e os testes do método escolhido devem possuir um balanceamento na distribuição de amostras por classes, e as amostras de uma mesma classe devem possuir características que as diferenciem das outras amostras de outras classes.

Nesse trabalho será feito a classificação de imagens de comida, dessa maneira a base montada contém imagens segregadas em classes como *pizza* e *sushi*. Também foram adicionadas classes de imagens que não estão relacionadas com comida, como *plant*(no português, planta) e *domestic animals*(no português, animais domésticos), visando melhorar o classificador, quando utilizado em bases que possuam imagens não associadas a comida.

As imagens utilizadas para a formulação da base de dados desse trabalho foram retiradas das bases de dados *ImageNet*(DENG et al., 2009) e *Food-101*(BOSSARD; GUILLAUMIN; GOOL, 2014). A base *ImageNet* é densamente estruturada e organizada em hierarquia de árvore, facilitando assim encontrar as categorias que estão relacionadas ao tema abordado. Já base *Food-101* é composta de 101 classes de imagens de comida, selecionadas e com um tamanho fixo de 1000 imagens por classe. A precisão de categorização das bases *ImageNet* e *Food-101* são necessárias para a formulação da base de teste deste projeto, tendo em vista que a categorização manual de imagens não seria uma abordagem viável, uma vez que redes neurais convolucionais demandam uma grande quantidade de imagens para um treinamento adequado.

A base formulada possui um total de 16000 imagens separadas em 16 classes, sendo dessas classes 13 relacionadas com comida (*chocolate cake, french fries, hamburger, ice cream, pizza, spaghetti bolognese, sushi,club sandwich, filet mignon, fried rice, hot dog, steak, tacos*) e três não relacionadas com comida (*domestic animal, people, plant*). A diversidade dessas imagens seguem como exemplo na Figura 8.

Figura 8 – Exemplos de imagens encontradas na base de dados.

Fonte: imagens retiradas das bases *ImageNet*(DENG et al., 2009) e *Food-101*(BOSSARD; GUILLAUMIN; GOOL, 2014)

3.1.1 Pré-processamento

Redes neurais convolucionais requerem de uma grande quantidade de imagens, dessa maneira as entradas fornecidas para a rede neural devem seguir um padrão, onde todas as imagens devem possuir as mesmas dimensões. Assim as imagens foram redimensionadas para 227x227 *pixels*, tendo em vista que é um valor que obteve bons resultados em trabalhos semelhantes (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Nas alterações de dimensões das imagens foi realizado um corte nas imagens originais para forçar uma formato quadrado antes de ser aplicado o redimensionamento, preservando os formatos das imagens.

Para a realização dos treinos e testes com a rede neural a base de dados foi separada em dados de treino e dados de teste. Como informado na Tabela 1 70% das imagens (11200 imagens) de cada classe serão utilizadas para o treino da rede neural, e os 30% das imagens restantes (4800 imagens) serão utilizadas para a fase de teste da rede neural.

Tabela 1 – Separação da base de dados em classes, sendo informado a quantidade de amostras separadas para realizar as etapas de treino e teste, também é informado a base de origem do dado.

Classe	Amostras de treino	Amostras de teste	Amostras totais	Base de origem
chocolate cake	700	300	1000	*Food-101*
french fries	700	300	1000	*Food-101*
hamburger	700	300	1000	*Food-101*
ice cream	700	300	1000	*Food-101*
pizza	700	300	1000	*Food-101*
spaghetti bolognese	700	300	1000	*Food-101*
sushi	700	300	1000	*Food-101*
club sandwich	700	300	1000	*Food-101*
filet mignon	700	300	1000	*Food-101*
fried rice	700	300	1000	*Food-101*
hot dog	700	300	1000	*Food-101*
steak	700	300	1000	*Food-101*
tacos	700	300	1000	*Food-101*
domestic animal	700	300	1000	*ImageNet*
people	700	300	1000	*ImageNet*
plant	700	300	1000	*ImageNet*

3.2 Configuração da Rede Neural

A configuração da rede neural convolucional utilizada nesse projeto é fundamentada na rede neural *AlexNet* definida por Krizhevsky, Sutskever e Hinton (2012). Essa rede utiliza de 8 camadas ocultas de processamento, sendo cinco camadas convolucionais e três camadas fortemente conectadas. Tendo em vista que a rede utilizada como exemplo foi estruturada para classificar mil classes, é necessário fazer algumas adaptações para classificar uma quantidade menor de classes, uma dessas alterações seria a modificação da função *softmax* para obter o resultado da classificação na última camada.

A primeira camada de convolução da rede irá separar a entrada, uma imagem de 227x227X3 em 96 núcleos de 11x11x3 (com uma distancia de 4 *pixels* entre os centros das imagens vizinhas), onde cada imagem gerada será processada separadamente. Após essa camada é aplicada uma camada de *max pooling*.

A segunda camada convolucional tem como entrada a saída da primeira camada convolucional normalizada e com *pooling* aplicado, essa entrada é filtrada em 256 núcleos de 5x5x48. Após esse camada também é aplicado uma camada de *max pooling*.

A terceira camada convolucional possui 384 núcleos de 3x3x256 que são conectados a entrada fornecida pela saída da segunda camada convolucional normalizada e aplicado o *pooling*. As conexões entre a terceira, quarta e quinta camada convolucional, não possuem nenhuma operação de *pooling* entre si.

Assim a quarta camada convolucional possui 384 núcleos de 3x3x192 e a quinta

camada possui 256 núcleos de 3x3x192. Após a quinta camada convolucional é aplicado uma camada de *max pooling*.

A duas camadas seguintes são camadas fortemente conectadas com 4096 neurônios cada. E por fim uma camada com 10 neurônios (um neurônio para cada classe) fortemente conectada com a operação de *softmax* para obter a predição da entrada.

Está sendo utilizado o *framework Keras* (CHOLLET et al., 2015) para descrever a rede neural. *Keras* é um *framework* em *python* para execução de redes neurais utilizando *GPU*(*Graphics Processing Unit*). Com o *keras* é possível configurar em alto nível uma rede neural, abstraindo a complexidade da descrição e implementação das rotinas de execução das camadas. Como exemplo de implementação temos a Figura 9, contendo a codificação da primeira e segunda camada da rede neural proposta.

Figura 9 – Trecho de código com implementação utilizando o *framework keras* das duas primeiras camadas convolucionais da rede neural, contendo a definição da entrada e as implementações das camadas de transição entre as camadas de convolução um e dois.

```
def AlexNet():
 # Descrição do formato da entrada
    entradas = Input(shape=(3,227,227))
 # Descrição da primeira camada convolucional contendo a quantidade de amostras que irá
 #  gerar, e a função de ativação que irá utilizar
    camada_conv_1 = Convolution2D(96, 11, 11,subsample=(4,4),activation='relu',
                                  name='camada_conv_1')(entradas)
 # Descrição da segunda camada convolucional, indica as camadas de pooling e normalização que
 #  ocorre antes da sua execução
    camanda_conv_2 = MaxPooling2D((3, 3), strides=(2,2))(camada_conv_1)
    camanda_conv_2 = crosschannelnormalization(name="camada_pool_1")(camanda_conv_2)
    camanda_conv_2 = ZeroPadding2D((2,2))(camanda_conv_2)
    camanda_conv_2 = merge([
        Convolution2D(128,5,5,activation="relu",name='camanda_conv_2_'+str(i+1))(
            segregracao_entradas(ratio_split=2,id_split=1)(camanda_conv_2)
        ) for i in range(2)], mode='concat',concat_axis=1,name="camanda_conv_2")

```

Fonte: Imagem produzida pelo autor (2017).

3.3 Melhorias para a rede neural proposta

Melhorias no desempenho de redes neurais podem ser aplicadas em diversas etapas do processo de aprendizado e classificação. O aumento da base de treino, definir camadas serão para serem treinadas em determinadas épocas, a inicialização dos pesos da rede neural com valores de uma rede treinada com uma quantidade maior de amostras e o aprimoramento dos parâmetros de configuração da rede, são métodos que podem ser utilizados para aprimorar sua performance de classificação.

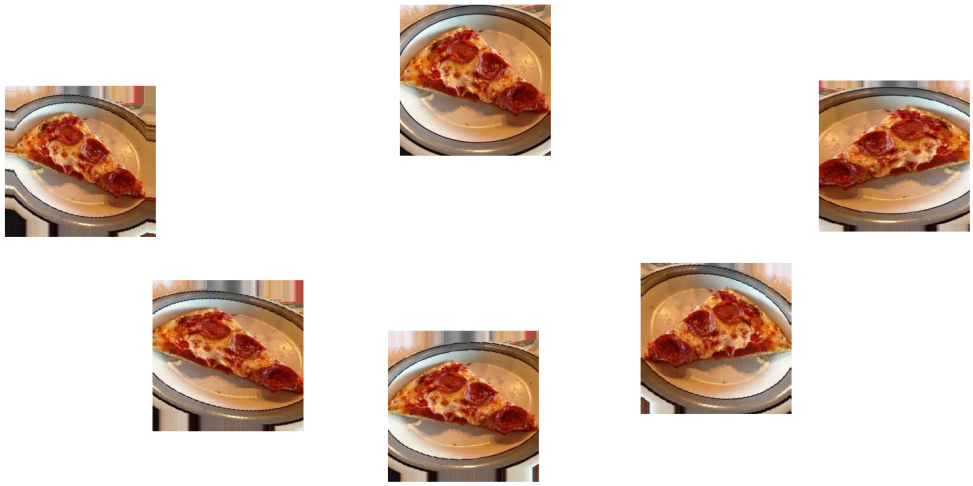
Nesse projeto foram aplicadas técnicas para a melhora na classificação, sendo elas o aumento dos dados na base de treino e a inicialização dos pesos da rede neural com pesos de uma rede já treinada.

3.3.1 *Data augmentation*

Uma técnica que vem sendo muito utilizada no para a redução do *overfitting* na fase de treino de uma rede neural é a *data augmentation* (CUI; GOEL; KINGSBURY, 2015). Os dados da base são ligeiramente modificados, para obter aumento na quantidade de amostras. Essas mudanças nas imagens podem ser inversões nos eixos, pequenas rotações, aproximações em certas partes das imagens ou até a aplicação da imagem em escala cinza. Essa técnica tem o propósito de aumentar a quantidade de amostras em que serão realizados os treinos, buscando evitar um *overfitting* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Nesse projeto foi utilizada a técnica de *data augmentation* aplicando inversão da imagem no eixo y, realizando um zoom de aproximação ou distanciamento de até 20% da imagem e aplicado uma taxa de inclinação de até 0,2 radianos. As imagens são geradas com a combinação das transformações possíveis informadas, criando nove imagens para cada imagem de treino como representado na Figura 10. Essas imagens são geradas durante a execução do treino da rede e são armazenadas em memória.

Figura 10 – Imagem representado a técnica de *data augmentation* aplicada na base de treino. A imagem no centro é a original, e as outras possíveis modificações aplicadas na imagem original, como a inversão do eixo y e o aumento e diminuição do *zoom*.



Fonte: Imagem produzida pelo autor (2017).

3.3.2 Inicialização dos pesos

Uma rede neural treinado do inicio ao fim, tem seus pesos inicializados de maneira aleatória e conforme vai realizando suas predições os pesos são corrigidos para melhorar o poder de classificação. A inicialização dos pesos da rede neural com valores obtidos a partir de uma rede treinada, vem sendo utilizado como maneira de melhorar a classificação. Geralmente os pesos vem de redes que treinaram uma quantidade muito grande de dados, conseguindo de

certa forma transferir o aprendizado obtido para a rede que está inicializando os pesos. Nesse projeto foi realizado a inicialização dos pesos da rede neural, utilizando os pesos obtidos no treinamento da rede desenvolvida por Krizhevsky, Sutskever e Hinton (2012).

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Nesta seção está descrito os resultados obtidos a partir dos testes realizados com a rede neural convolucional. A métrica utilizada para avaliar os modelos foi a acurácia, que determina a porcentagem de acerto do classificador.Os testes foram iniciados a partir da rede neural convolucional sem nenhuma alteração, seguindo para a inclusão das alterações na base e melhorias na rede, com o fim de avaliar os resultados obtidos e determinar o modelo com melhor acurácia.

4.1 Modelo inicial

O primeiro teste realizado foi com a base de dados sem modificação sobre o modelo inicial de rede neural proposta, com uma configuração de execução com 80 épocas e uma taxa de redução de neurônios nas camadas totalmente conectadas de 50%. Como visto na **??** no final da execução da época 80 foi obtido uma acurácia na fase de teste de 53,6% e 94,02% na fase de treino, identificando assim um *overfitting* do classificador sobre as amostras da base de treino. É possível perceber dado a análise do **??** que a acurácia da fase de teste a partir de 42 épocas mantém valores constantes, não apresentando uma ganho expressivo, diferente da acurácia obtida na fase de treino que mantém uma curva crescente.

4.2 Modelos com *data augmentation* e inicialicialização dos pesos

Também foram realizados teste com as estrategias de melhorias da rede descritas anteriormente , com o objetivo de reduzir o *overfitting* na rede neural. Foram realizados três testes: o teste apenas com a técnica de *data augmentation*; o teste com a técnica de inicialização dos pesos a partir dos valores treinados na rede desenvolvida por Krizhevsky, Sutskever e Hinton (2012); e o teste aplicando as duas técnicas.

Como informado na **??** as acurácias obtidas na fase de teste e treino, respectivamente, apenas com *data augmentation* foram de 58,23% e 71,66%, vendo com essa mudança uma redução significativa do *overfitting* na rede, além de uma melhora da acurácia na fase de teste.

O teste realizado com a inicialização dos pesos obteve um resultado mais expressivo se comparado com o teste apenas com *data augmentation* obtendo as acurácias na fase de teste e treino, respectivamente, de 68,08% e 99,63%. Com a aplicação dessa técnica foi obtido uma melhora significativa na acurácia se comparado com o modelo inicial e o modelo com *data augmentation*, mas assim como no modelo inicial esse modelo apresenta *overfitting* na fase de treino, como é possível observar no **??** a partir da época 40 a acurácia de teste permanece estável e já a acurácia da fase de treino continua crescendo até atingir valores próximos a 100%.

Com o intuito de solucionar o problema de *overfitting* e obter melhoria na classificação, foi realizado o teste utilizando as duas técnicas. A acurácia obtida na fase de treino foi de 98,18% e na fase de teste obteve um valor de 71,77%, apresentando o melhor resultado entre os três testes realizados.

4.3 Aprimoramento da taxa de *Dropout*

Foram realizados testes com a taxa aplicada na função de *Dropout* buscando diminuir o *overfitting* na rede. A taxa de *dropout* foi variada de 50% a 90%, com intervalos de 10% entre cada teste. Como pode ser verificado na **??** a taxa de 80% foi a que apresentou melhor resultado com uma acurácia de 74,56% na fase de teste e 93,96% de acurácia na fase de treino.

A partir da **??** é possível concluir que os teste realizado com a taxa de *dropout* 80% apresentou o melhor resultado pois se encontra no limiar entre uma taxa que permite aprender características importantes das classes avaliadas sem ficar super adaptada a base de dados de treino (*overfitting*).

5 CONCLUSÃO

Parte final do texto, na qual se apresentam as conclusões do trabalho acadêmico. É importante fazer uma análise crítica do trabalho, destacando os principais resultados e as contribuições do trabalho para a área de pesquisa.

5.1 TRABALHOS FUTUROS

Também deve indicar, se possível e/ou conveniente, como o trabalho pode ser estendido ou aprimorado.

5.2 CONSIDERAÇÕES FINAIS

Encerramento do trabalho acadêmico.

Referências

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 35, n. 8, p. 1798–1828, 2013. Citado na página 8.

BOSSARD, L.; GUILLAUMIN, M.; GOOL, L. V. Food-101 – mining discriminative components with random forests. In: **European Conference on Computer Vision**. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 12 e 13.

CHOLLET, F. et al. **Keras**. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>. Citado 2 vezes nas páginas 12 e 15.

CUI, X.; GOEL, V.; KINGSBURY, B. Data augmentation for deep neural network acoustic modeling. **IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)**, IEEE Press, v. 23, n. 9, p. 1469–1477, 2015. Citado na página 16.

DEMUTH, H. B. et al. **Neural Network Design**. 2nd. ed. USA: Martin Hagan, 2014. ISBN 0971732116, 9780971732117. Citado 2 vezes nas páginas 4 e 5.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**. [S.l.], 2009. p. 248–255. Citado 2 vezes nas páginas 12 e 13.

ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, Nature Research, v. 542, n. 7639, p. 115–118, 2017. Citado na página 2.

GAMBOGI, J. A. **Aplicação de redes neurais na tomada de decisão no mercado de ações.** Tese (Doutorado) — Universidade de São Paulo, 2013. Citado na página 2.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics**. [S.l.: s.n.], 2011. p. 315–323. Citado na página 10.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 4 vezes nas páginas 8, 9, 10 e 11.

GURNEY, K. **An Introduction to Neural Networks**. Taylor & Francis, 1997. (An Introduction to Neural Networks). Acessado em 23 de maio de 2017. ISBN 9781857285031. Disponível em: <https://books.google.com.br/books?id=HOsvlIRMMP8C>. Citado na página 2.

HAYKIN, S. **Redes Neurais - 2ed.** [S.l.]: BOOKMAN COMPANHIA ED, 2001. ISBN 9788573077186. Citado 6 vezes nas páginas 2, 3, 4, 5, 6 e 7.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of artificial intelligence research**, v. 4, p. 237–285, 1996. Citado na página 4.

KRIESEL, D. **A Brief Introduction to Neural Networks**. [s.n.], 2007. Acessado em 28 de maio de 2017. Disponível em: <http://www.dkriesel.com>. Citado 2 vezes nas páginas 2 e 3.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Acessado em 27 de maio de 2017. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>. Citado 6 vezes nas páginas 2, 13, 14, 16, 17 e 18.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Research, v. 521, n. 7553, p. 436–444, 2015. Citado 2 vezes nas páginas 1 e 8.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, MIT Press, v. 1, n. 4, p. 541–551, 1989. Citado na página 8.

ZHANG, G. P. Neural networks for classification: a survey. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, IEEE, v. 30, n. 4, p. 451–462, 2000. Citado na página 2.