Universidade Federal de Pernambuco

Centro de Informática - Cin

Graduação em Engenharia da Computação

Análise da relevância das saídas de *ConvNets* através de aprendizagem não-supervisionada

Heitor Rapela Medeiros

TRABALHO DE GRADUAÇÃO

Recife, Junho de 2017

Universidade Federal de Pernambuco

Centro de Informática - Cin

Graduação em Engenharia da Computação

Análise da relevância das saídas de *ConvNets* através de aprendizagem não-supervisionada

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Aluno: Heitor Rapela Medeiros **Orientador:** Hansenclever de França Bassani

"Comece fazendo o que é necessário, depois o que é possível, e de repente você estará fazendo o impossível." - **São Francisco de Assis**

Agradecimentos

Gostaria de agradecer a Deus por ter me dado saúde e força para superar as dificuldades. Agradeço a todos os professores por me proporcioanr o conhecimento, e em especial, ao professor Hansenclever pela oportunidade. Também gostaria de agradecer a minha família, por serem tudo que se pode esperar de uma família. Em especial, a minha mãe Angela por ser um exemplo de mãe, batalhadora, esforçada, professora e pesquisadora, além de atuar na sua área. Ao meu pai, por nos proporcionar uma boa qualidade de vida, além de muito amor, é um exemplo de pai. A meu irmão por me dar orgulho nos estudos, e nos jogos de vídeo-game. A minha irmã por ser chata, mas mesmo assim eu gosto muito dela, e sei que o trabalho que ela faz realmente vai mudar o mundo ajudando as crianças que precisam de sua ajuda. A minha madrinha e minhas avós por terem me criado dá melhor maneira possível. A minha namorada Lorenna por me ajuda a ter calma em momentos difíceis. E por fim, em memória do meu avô, que foi um exemplo de homem para mim, e me ensinou a ser um pouco de engenheiro, consertando vários equipamentos danificados.

Resumo

Atualmente, se tem máquinas mais potentes e unidade de processamento gráfico (GPU's) dedicadas para fazer processamento paralelo e realizar tarefas que antigamente, não eram possíveis, além de uma quantidade grande de dados que está organizada para ajudar no processo de aprendizagem. Com estes avanços, permitiu-se que métodos de aprendizagem profunda, como as redes convolucionais pudessem explorar funções complexas, para aprender representações de certas caracterísiscas. Porém, ainda não se sabe como as redes aprendem as características que conseguem descrever bem suas entradas. Por isto, este trabalho tem como objetivo fazer uma análise das relevâncias das saídas de uma rede neural convolucional através de aprendizagem não-supervisionada e técnicas de visualização, e assim, tentar entender o seu funcionamento. No desenvolvimento foram utilizadas técnicas como redes de deconvolução, para gerar as visualizações das características e a rede não-supervisionada LARFDSSOM, para agrupar as categorias e fornecer as relevâncias das características. Além de fornecer um material sólido para pesquisas futuras, o trabalho apresenta uma possível abordagem para o entendimento das redes através de aprendizagem não-supervionada e ferramentas de visualização de características.

Palavras-chaves: Redes Neurais Convolucionais; Visualização de Características; Aprendizagem Profunda; Mapas Auto-Organizáveis;

Abstract

Nowadays, there are more powerful machines and GPUs dedicated to process tasks that 10 or 20 years ago were not possible, and a large amount of data have been organized to aid in the learning process. With these advances, deep learning methods, such as convolutional networks, were allowed to explore complex functions and learn representations of certain features. However, it is not yet known how networks learn the fetuares that can describe their inputs well. This work aimed to analyze relevances of the outputs from a convolutional neural network through unsupervised learning and visualization techniques, and try to understand it. In the development, techniques such as deconvolution networks were used to generate the features visualizations and the non-supervised network, LARFDSSOM, to group the categories and provide the features relevance. In addition, this work, provides a solid material for future research, the paper presents a possible approach to understanding networks through unsupervised learning and feature visualization tools.

Keywords: Convolutional Networks; Deconvolution Networks; Deep Learning; Features Visualization; Self Organizing Maps.

Sumário

1 Introdução **11**

1.1 Motivação 11

1.2 Objetivos 12

1.3 Estrutura do Documento 12

2 Revisão de Literatura **13**

2.1 Conceitos Básicos 13

2.1.1 Aprendizagem de Máquina 13

2.1.2 Aprendizagem Supervisionada 15

2.1.2.1 *Perceptron* 15

2.1.2.2 *Multilayer Perceptron* (MLP) 15

2.1.3 Aprendizagem Não-Supervisionada 17

2.1.3.1 *Self-organizing Map* (SOM) 17

2.2 Aprendizagem Profunda - *Deep Learning* 19

2.2.1 Representação Visual 20

2.2.2 Rede Neural Convolucional (RNC) - *ConvNets* 22

2.2.2.1 Camada Convolucional 23

2.2.2.2 Camada *Rectified Linear Unit* (ReLU) 25

2.2.2.3 Camada de *Pooling* 26

2.2.2.4 Camada Totalmente Conectada 27

2.2.3 GoogLeNet - Rede Convolutiva Profunda (RCP) 28

2.2.4 Rede de Neural de Deconvolução (RND) - DeConvNet 30

2.2.4.1 *Unpooling - max pooling reverso* 32

2.2.4.2 Deconvolução 32

2.2.5 *Transfer Learning* 33

2.3 Mapa Auto-Organizável Seletivo de Dimensões com Campo Receptivo Localmente Adaptativo (LARFDSSOM) 34

2.3.1 DSSOM 34

2.3.2 LARFDSSOM 35

3 Metodologia **38**

3.1 Base de dados - *Dataset Easy* 38

3.2 Implementação 38

3.2.1 Etapa LARFDSSOM 39

3.2.2 Etapa de Visualização 40

3.3 Métricas 41

3.3.1 *Clustering Error* - CE 41

3.3.2 Matriz de Confusão 42

3.3.3 Aplicação da Metodologia 42

4 Resultados do Trabalho **43**

4.1 Busca de Parâmetros e CE 43

4.2 Matriz de Confusão 43

4.3 Estudo de Caso sobre as Relevâncias Aprendidas pelo LARFDSSOM 47

5 Conclusão **53**

Lista de Tabelas

1 *Dataset Easy* - 20 categorias de objetos selecionadas por Tuytelaars *et al.* [1] . . 39

2 Intervalos de mínimo e máximo - Busca de parâmetros no LARFDSSOM 43

3 *Clustering Error* e Parâmetros do LARFDSSOM - 10 melhores resultados encontrados 44

4 Matriz de Confusão - LARFDSSOM (*Cluster* x Rótulo) 44

5 Mapeamento Categoria do *Dataset Easy* e Rótulo 45

6 Relevâncias do LARFDSSOM por *cluster* x Neurônios de maiores ativações para categoria zebra 50

7 Relevâncias do LARFDSSOM por *cluster* x Neurônios de maiores ativações para categoria *french horn* 51

Lista de Figuras

1 Modelo de Aprendizagem Supervisionado. O professor na figura é responsável por rotular os exemplos de entrada - Fonte: Braga *et al.* (2007)[2] 14

2 Modelo de Aprendizagem Não-Supervisionado - Fonte: Braga *et al.* (2007) [2] . 14

3 Modelo de Neurônio Artificial: Peceptron Simples - Fonte: Autor 15

4 Ilustração de um tipo de MLP treinada com o algoritmo *Backpropagation* - Fonte: Autor 16

5 Ilustração de uma rede SOM, onde o vetor na parte de baixo é o vetor de entrada **x**, as ligações são os pesos **w** e a parte de cima é a camada de saída - Fonte: Bassani, H (2012) [3] 18

6 Representação do treinamento de um Mapa Auto-Organizável. O conjunto de padrões de entrada, em azul, e o padrão escolhido na iteração atual, ponto branco. O grid preto são os vetores de pesos e suas conexões, e o vetor do vencedor (em amarelo). Ele e seus vizinhos são então movidos para mais próximo do padrão de entrada. Fonte: Wikipédia (disponível sobre licença Creative Commons) 18

7 Hubel e Weisel - Fonte: Lehar, Steven [4]. 21

8 Visualização de características de uma ConvNet - Baixo, Médio e Alto Nível, Fonte: Zeiler e Fergus [5]. 22

9 Rede Convolucional de LeCun *et al* (1998). 23

10 Convolução com filtro 5x5 onde a entrada é mapeada na saída (mapa de ativação), Fonte: Michael A Nielsen (2015) [6] 24

11 Comportamento da função ReLU - Fonte: Autor 25

12 Pooling de máximo: reduz o tamanho dos dados - Fonte: Karpathy, A (2016) [7]. 26

13 Camadas de uma CNN em sequência (Convolução, ReLU, Totalmente Conectada) - Fonte: Deshpande, A [8]. 27

14 Módulo Inception com redução de dimensão - Fonte: Szegedy *et al.* [9] 29

15 Arquitetura da rede GoogLeNet - Fonte: Autor 29

16 Reconstrução de uma imagem usando DeconvNet - Fonte: Karpathy, A (2016) [7]. 30

17 Lado esquerdo: DeConvNet e Lado Direito: ConvNet - Fonte: Zeiler e Fergus (2013) [5] 31

18 Visualização da técnica de *unpooling* - Fonte: Karpathy, A (2016) [7]. 32
19 Visualização dos processos de *Pooling*, *Unpooling*, *Convolução* e *Deconvolução* - Fonte: Karpathy, A (2016) [7]. 33
20 Arquitetura da rede GoogLeNet retirando a camada do classificador e inserindo o LARFDSSOM - Fonte: Autor 40
21 *Deep Visualization Toolbox* adaptado - Fonte: Autor 41
22 Subdivisão possível da categoria *rotary phone* Fonte: Autor 46
23 A imagem da esquerda contém a semelhança no formato cúbico do dado e do anel. A imagem do meio apresenta uma junção sobreposta das duas imagens anteriores. A imagem mais a direita apresenta outra semelhança no formato - Fonte: Autor 46
24 Imagens da categoria *dice* que são difíceis de agrupar, porque há outros objetos nas imagens - Fonte: Autor 46
25 Imagens da categoria *diamond ring* que são difíceis de agrupar, porque há outros objetos na imagem, ou a iluminação da imagem está ruim - Fonte: Autor 47
26 Imagens de treinamento na ferramenta *Deep Visualization Tool*: o neurônio 54 (verde) selecionado na camada *inception*3a, a imagem de entrada (azul), os valores da resposta do neurônio a entrada (vermelho) e a imagem gerada pela deconvolução (amarelo) - Fonte: Autor 47
27 Imagens de algumas categorias para o neurônio 813 camada pool5/7x7_s1. A entrada, em azul, o valor dos pesos (tons de cinza variando de 0 até 1), em amarelo, e as imagens geradas pela deconvolução, em vermelho - Fonte: Autor . 48
28 Maiores ativações neurônio 813 camada pool5/7x7_s1, em azul, e as imagens geradas pela deconvolução, em vermelho - Fonte: Autor 49

1 Introdução

Neste capítulo, serão apresentados: a motivação para o desenvolvimento do trabalho (Seção 1.1), os objetivos propostos (Seção 1.2) e a estruturação do documento (Seção 1.3).

1.1 Motivação

Permitir que os computadores modelem o mundo de maneira satisfatória é algo que está sendo mais possível devido ao avanço tecnológico. Atualmente, se tem máquinas mais potentes e unidade de processamento gráfico (GPU's) para fazer processamento paralelo e realizar tarefas que antigamente, não eram possíveis, além de uma quantidade grande de dados que está organizada para ajudar no processo de aprendizagem. Com estes avanços, permitiu-se que métodos de aprendizagem pudessem explorar funções não-lineares e com níveis crescentes de complexidade, para aprender representações de caracterísiscas de forma hierárquica [10].

A maioria das tarefas realizadas por computadores partem do processo de dividir atividades mais complicadas em atividades menores, por exemplo em algoritmos de ordenação. Partindo deste princípio, pode-se fazer um comparativo com as redes com várias camadas, onde as camadas iniciais aprendem certas características, e nas camadas finais, uma representação próxima da interpretação desejada, geralmente de alto nível de abstração. Atualmente, não se tem uma explicação de como as camadas conseguem representar bem as caracteríticas da entrada, porém existem ferramentas que buscam criar uma aproximação do que cada camada da rede está aprendendo, com o objetivo de fornecer uma representação que facilite a interpretação do que está sendo aprendido.

Se através de um mecanismo automático for possível descrever com precisão partes de objetos, poderiam ser feitos agrupamentos e processos de composições destas partes, para se chegar em uma representação formada por partes do objeto e que seja próxima da representação completa dele. Imaginando um robô autônomo responsável pelo resgate de pessoas, se este robô é capaz de identificar com precisão grupos que contenham olhos e grupos que contenham dentes afiados, ele poderia criar uma representação de um grupo, automaticamente, sendo a junção destes dois grupos, para representar um possível predador e, com isto, resgatar a pessoa (que não tem dentes afiados) de um tigre (tem olhos e dentes afiados). Ou seja, descobrir os valores de características que são relevantes com precisão e que descrevem partes de objetos, resultaria em um importante avanço para área.

O trabalho, tem como foco entender o funcionamento das camadas das redes neurais convolucionais, para entender a relação existente entre os métodos de aprendizagem supervisionada e não-supervisionada.

1.2 Objetivos

Esse trabalho de graduação tem como objetivo geral fazer uma análise das relevâncias das saídas de uma Rede Neural Convolucional através de aprendizagem não-supervisionada e técnicas de visualização. São objetivos específicos deste trabalho:

1. Utilizar técnicas de visualização de características das camadas intermediárias, para entender o funcionamento da rede.

2. Extrair as características de uma rede neural convolucional, para o trabalho proposto foi escolhido a rede GoogLeNet, através da ferramenta de aprendizagem profunda, Caffe. Utilizar as características extraídas na rede não-supervisionada LARFDSSOM.

3. Realizar uma análise comparativa entre a técnica de visualização, redes de deconvolução, e as relevâncias aprendidas pelo LARFDSSOM.

1.3 Estrutura do Documento

Este trabalho está organizado da seguinte maneira: no Capítulo 2 serão apresentados os principais conceitos básicos, além de Redes Convolucionais, Redes de Deconvolução e a Rede LARFDSSOM. No Capítulo 3, será apresentada a Metodologia Científica, onde é mencionadas as ferramentas e as métricas utilizadas. No Capítulo 4, os resultados obtidos na etapa de testes. Por último, no Capítulo 5 será apresentado a conclusão deste trabalho.

2 Revisão de Literatura

Neste capítulo, é aprensentado o referêncial teórico para o entendimento do trabalho realizado. O capítulo está dividido em três partes:

1. Conceitos Básicos: Na Seção 2.1.1, contém definições de aprendizagem de máquina, além dos conceitos de aprendizagem supervisionada e não-supervisionada. Em seguida, a Seção 2.1.2, parte supervisionada, fala sobre os conceitos de *Perceptron* e *Multilayer Perceptron* (MLP). Na última Seção 2.1.3, parte não-supervisionada, é apresentado o *Self-organizing Map* (SOM).

2. *Deep Learning*: Na Seção 2.2.1, fala sobre a inspiração de uma ConvNet. Em seguida, na Seção 2.2.2, tem-se um conceito de Rede Neural Convolucional. Na Seção 2.2.3, é apresentada a GoogLeNet. Na Seção 2.2.4, é apresentada a Rede Neural de Deconvolução. Para finalizar, há uma uma Seção 2.2.5, referente a *Transfer Learning*.

3. LARFDSSOM: Na Seção 2.3.1, é apresentado o DSSOM. Em seguida, na Seção 2.3.2, o capítulo é finalizado com a definição e explicação do LARFDSSOM.

2.1 Conceitos Básicos

2.1.1 Aprendizagem de Máquina

Aprendizagem de máquina é "um campo de estudo da computação que dá ao computador a habilidade para aprender sem ser explicitamente programado" [11]. Já Tom Mitchell define aprendizagem de máquina como: "Estudo de algoritmos que melhoram sua performance P, em uma tarefa T com uma experiência E". Ou seja, ele define uma tarefa de aprendizado através de (P,T,E) [12]. Duas subáreas da aprendizagem de máquina são consideradas neste trabalho:

- Aprendizagem Supervisionada: Tem como objetivo inferir uma função de um conjunto de dados de treino rotulados [13]. A aprendizagem supervisionada, recebe este nome, porque a entrada e saída desejadas para a rede são fornecidas por um supervisor (professor) externo. O objetivo é ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos [2]. A Figura 1 mostra a retroalimentação vinda pelo professor, que através dos rótulos, ajuda no treinamento da Rede Neural Artificial (RNA).

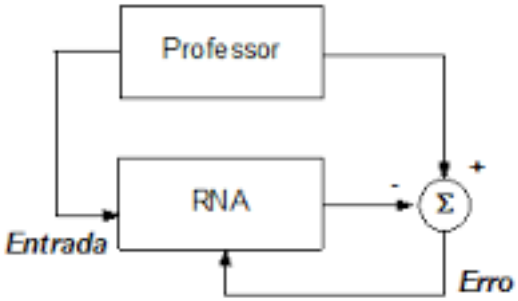

Figura 1: Modelo de Aprendizagem Supervisionado. O professor na figura é responsável por rotular os exemplos de entrada - Fonte: Braga *et al.* (2007)[2]

- Aprendizagem Não-supervisionada: Tem como objetivo inferir uma função para descrever uma estrutura a partir de dados não-rotulados. No aprendizado não-supervisionado, como o próprio nome sugere, não há um professor ou supervisor para acompanhar o processo de aprendizado. A partir do momento em que a rede estabelece uma harmonia com as regularidades estatísticas da entrada de dados, desenvolve-se nela uma habilidade de formar representações internas para codificar características da entrada e criar novas classes ou grupos automaticamente [2]. O método está ilustrado na Figura 2.

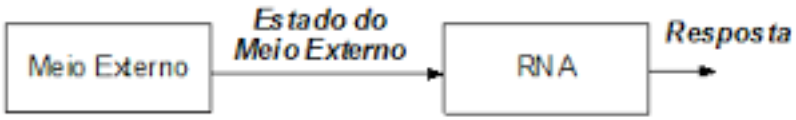

Figura 2: Modelo de Aprendizagem Não-Supervisionado - Fonte: Braga *et al.* (2007) [2]

2.1.2 Aprendizagem Supervisionada

Nesta Seção são apresentados duas técnicas de aprendizagem supervisionada: o *Perceptron* e o *Multilayer perceptron* (MLP).

2.1.2.1 *Perceptron*

O primeiro modelo artificial de um neurônio biológico foi proposto por Warren McCulloch e Walter Pitts em 1943 [14]. O trabalho realizado por McCulloch e Pitts descreve um modelo artificial de neurônio e apresenta suas capacidades computacionais. Em 1949, Donald Hebb publicou um trabalho que relacionava as redes neurais com a tarefa de aprendizagem [15]. A teoria proposta por Hebb, pode explicar alguns tipos de aprendizagem associativos no qual a ativação simultânea de células leva a um crescimento pronunciado na força sináptica. Em 1958, Rosenblatt [16] propôs o modelo do perceptron simples, que através de sinapses ajustáveis, as redes neurais artificiais poderiam ser treinadas para classificar padrões. O perceptron simples, mostrado na Figura 3, recebe os valores (x_i) que vão ser utilizados como entrada pelo neurônio, com seus pesos (w_i), que passam por uma função de ativação, e são somados gerando a saída y.

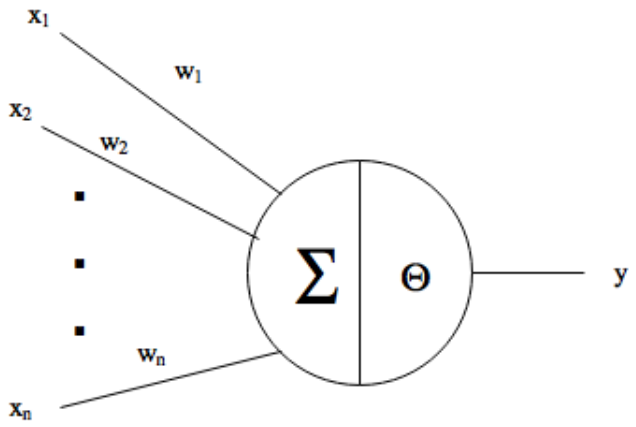

Figura 3: Modelo de Neurônio Artificial: Peceptron Simples - Fonte: Autor

2.1.2.2 *Multilayer Perceptron* (MLP)

Uma Rede Neural Artificial (RNA) é um modelo computacional inspirado no sistema nervoso central dos animais. O modelo de uma RNA, como por exemplo o MLP (Figura 4), consiste

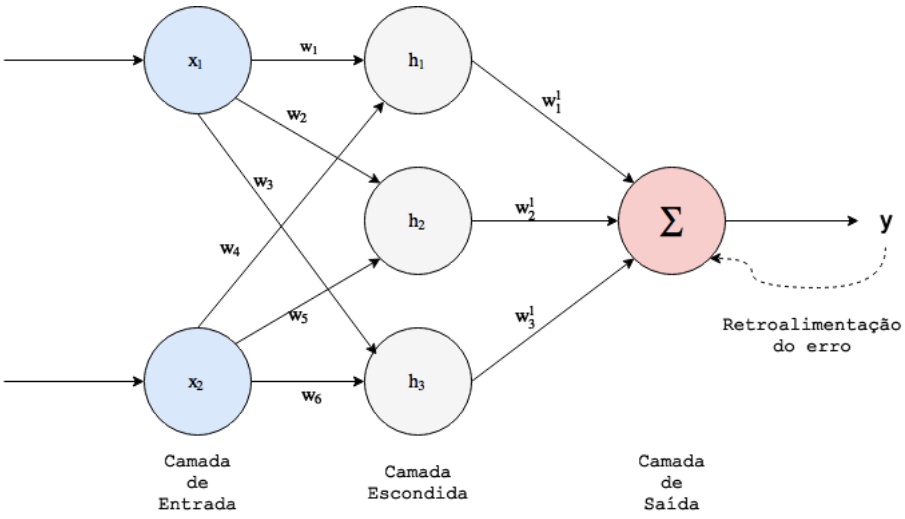

Figura 4: Ilustração de um tipo de MLP treinada com o algoritmo *Backpropagation* - Fonte: Autor

de múltiplas unidades de processamento simples chamadas de neurônios, cada uma produz uma sequência de ativações. Neurônios de entrada são ativados percebendo o ambiente, outros neurônios são ativados pelos pesos de suas conexões com os neurônios ativados anteriormente. Dependendo do problema e como os neurônios estão conectados, o comportamento de saída pode precisar de vários estágios computacionais para serem ativados. Na fase de treinamento das redes neurais, apresenta-se em suas entradas um conjunto de dados de treinamento e espera-se que a rede seja capaz de extrair características importantes. Já na fase de testes, espera-se que a rede tenha generalizado os dados e consiga acertar para entradas diferentes dos padrões treinados previamente.

O treinamento das redes neurais artificiais pode contar com um algoritmo chamado *backpropagation*, que se baseia na retropropagação dos erros para realizar os ajustes de pesos das camadas intermediárias [17]. Com a utilização do *backpropagation* a rede consegue ajustar seus parâmetros para gerar uma saída com erro menor. Na figura 4, é apresentado um modelo de RNA com a técnica *backpropagation*, que recebe os valores (x_i) que vão ser utilizados como entrada pelo neurônio, com seus pesos (w_i), passam por uma camada escondida com os neurônios h_i e seus respectivos pesos h_i^1, que passam por uma função de ativação, e são somados gerando a saída y. A saída obtida (y) que comparada com a saída desejada (t) gera um erro que é propagado até a primeira camada. Segue uma breve descrição de como calcular o erro para ajustar os pesos da rede na fase de treinamento:

a) Passo de avanço da rede (*Forward Pass*) utiliza a função de ativação ϕ, que recebe como entrada o resultado do somatório e converte na saída, representada na Equação 1 para

treinar a rede, onde y é a saída atual do sistema, (w_i é o peso da ligação do neurônio, x_i é o valor atual da entrada do neurônio, n é o número de nós da rede:

$$y = \phi \left(\sum_{i=1}^{n} w_i x_i \right) \tag{1}$$

b) O erro E gerado pela saída é calculado através do erro entre a saída atual y e a saída desejada t, como mostra a Equação 2:

$$E = \frac{1}{2} (t - y)^2 \tag{2}$$

c) Passo de retropropagação (*Backward Pass*) propaga o erro sobre a rede, no qual o valor para atualização de cada peso é computado através da Equação 3 (η é a taxa de aprendizagem) aplicada na Equação 4.

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{3}$$

$$w_i = w_i + \Delta w_i \tag{4}$$

2.1.3 Aprendizagem Não-Supervisionada

Nesta Seção é apresentada a técnica de aprendizagem não-supervisionada: *Self-organizing Map* (SOM).

2.1.3.1 *Self-organizing Map* (SOM)

A rede *Self Organizing Maps* (SOM), proposta por Kohonen [18] (Figura 5), é uma RNA treinada com aprendizagem não-supervisionada. Esta rede realiza um mapeamento de um espaço contínuo de entrada para um espaço discreto de saída, onde as propriedades topológicas da entrada são preservadas. Quando a rede SOM se adapta a entradas de altas dimensões, ela tende a se estender e enrolar para cobrir o espaço de entrada, ilustrado na Figura 6. Além disso, as redes SOM tem a capacidade de organizar dimensionalmente dados complexos em grupos (*clusters*), de acordo com suas relações. Este método necessita apenas dos conjuntos de dados de entrada, mostrando-se ideal para problemas onde os padrões são desconhecidos

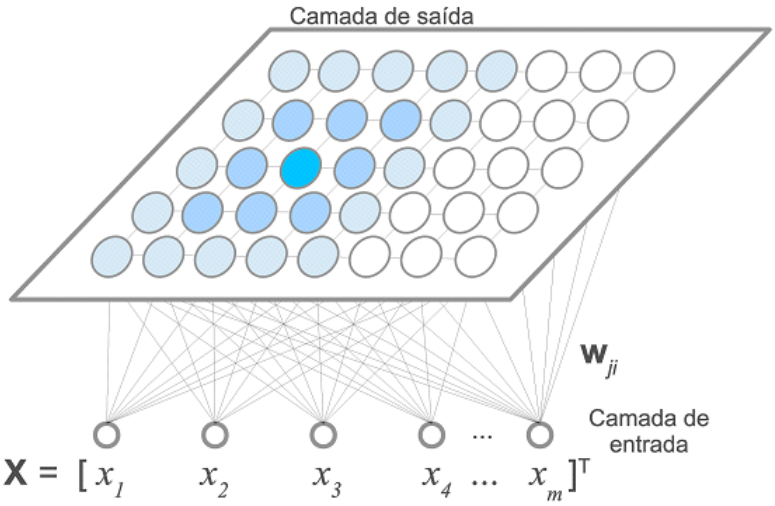

Figura 5: Ilustração de uma rede SOM, onde o vetor na parte de baixo é o vetor de entrada **x**, as ligações são os pesos **w** e a parte de cima é a camada de saída - Fonte: Bassani, H (2012) [3]

ou indeterminados. Este tipo de rede é utilizado para uma variedade de aplicações, tais como em reconhecimento de voz, análise de imagem, processos industriais de controle, organização automática de documentos numa biblioteca, visualização de registros financeiros [19].

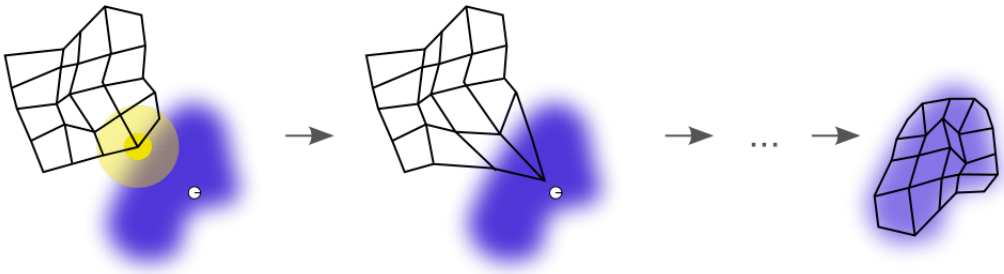

Figura 6: Representação do treinamento de um Mapa Auto-Organizável. O conjunto de padrões de entrada, em azul, e o padrão escolhido na iteração atual, ponto branco. O grid preto são os vetores de pesos e suas conexões, e o vetor do vencedor (em amarelo). Ele e seus vizinhos são então movidos para mais próximo do padrão de entrada. Fonte: Wikipédia (disponível sobre licença Creative Commons)

Na rede SOM, os neurônios são organizados em uma estrutura regular, sendo as mais utilizadas a grade bidimensional hexagonal e a retangular, como na Figura 6. Cada neurônio é associado a um vetor de pesos com a mesma dimensão dos dados de entrada, por exemplo, se o os dados de entrada estão organizados em vetores de 10 componentes, cada neurônio da rede SOM terá um vetor de pesos de 10 componentes. O algoritmo SOM é um algoritmo de aprendizagem competitiva, onde os padrões são mapeados para o neurônio cujo vetor de pesos esteja mais próximo utilizando a distância euclidiana. Os vetores de pesos podem ser vistos como sendo protótipos dos padrões de entrada, com isto, mais protótipos são deslocados para

regiões que possuem mais padrões. A rede SOM pode ser dividida em duas etapas: treinamento e mapeamento.

Na etapa de treinamento cada neurônio i da rede recebe um vetor de pesos w_i. Os vetores de pesos podem ser gerados aleatoriamente ou seguindo os padrões amostrados do conjunto de entrada. Em seguida, por inúmeras vezes: escolhe-se um padrão $\mathbf{x}$, de maneira aleatória do conjunto de entradas, e coloca como entrada da rede SOM. Para cada neurônio: calcula-se a distância euclidiana entre os pesos e o padrão de entrada $\mathbf{x}$, e com isto, o vetor de pesos do neurônio vencedor i, o qual possui a menor distância euclidiana, é atualizado. O resultado desta atualização é mover o vetor de pesos do neurônio vencedor para mais perto do padrão de entrada. O passo de atualização do vetor de pesos, é descrito pela Equação 5, onde e é a taxa de aprendizagem da rede (responsável pelo quanto o padrão vai se mover):

$$w_i = w_i + e(x - w_i) \tag{5}$$

Os vetores de pesos dos neurônios vizinhos de i, na rede, também são atualizados da mesma forma, mas em menor grau, seguindo a Equação 6, onde $N(i, j)$ é uma função responsável por dizer o quão próximo na rede SOM, estão os neurônios i e j. Geralmente, é utilizada uma função gaussiana centrada no neurônio vencedor para a função N.

$$w_j = w_j + N(i, j)e(x - w_i) \tag{6}$$

Durante várias iterações na fase de treinamento, os vetores de pesos da rede convergem para modelar a distribuição dos dados de entrada. Depois, na fase de mapeamento, cada padrão é atribuído ao neurônio com o vetor de pesos mais próximo.

2.2 Aprendizagem Profunda - *Deep Learning*

Os autores Deng et al. [10] sugerem várias definições sobre o que é aprendizagem profunda, segue uma lista com algumas definições:

1. Classe de técnicas de aprendizagem de máquina para explorar camadas de informações não-lineares processando e extraindo características supervisionadas e não-supervisionadas, além de realizar transformações para analisar e classificar padrões.

2. Subcampo de aprendizagem de máquina que é utilizado algoritmos para representação de

níveis com o proposito de modelar relações complexas diante dos dados. Onde características de alto nível são definidas em termos de características de baixo nível.

3. Subconjunto de métodos de aprendizagem de máquina baseados em representações do aprendizado. Uma observação (ex: uma imagem), pode ser representada de diversas maneiras (ex: conjunto de *pixels*), mas algumas representações são mais fáceis para aprender tarefas específicas.

4. Conjunto de algoritmos em aprendizagem de máquina que visam aprender múltiplos níveis, correspondendo a diferentes níveis de abstração. Os níveis em modelos aprendidos correspondem a níveis diferentes de conceitos, onde os conceitos de mais alta ordem são definidos através de conceitos de mais baixa ordem, e os conceito dos níveis de baixa ordem podem ajudar a definir vários dos conceitos de alta ordem.

Pode-se notar que todas as definições tem a ideia comum de que algoritmos de aprendizagem profunda são responsáveis por aprender características hierárquicas de níveis mais altos formados pela composição de características de níveis mais baixos, e isto, permite um sistema complexo aprender funções mapeando a entrada para a saída diretamente dos dados. A arquitetura de uma rede profunda pode ser vista como a composição de vários estágios, surgindo a pergunta: Que tipo de representação pode ser encontrada na saída de cada estágio? [20] (Seção 2.2.1)

Atualmente, existem vários algoritmos para tentar responder está pergunta. Esses algoritmos podem ser vistos como aprendizado para transformar uma representação (a saída de um estágio anterior) em outra representação, para entender o processamento intermediário dos dados. Neste trabalho, as representações das camadas intermediárias são geradas através de Redes Neurais de Deconvolução (Seção 2.2.4) [5] [21].

2.2.1 Representação Visual

Por anos o homem propõe modelos que se inspiram no cérebro, porém ainda existem várias dúvidas: Como o cérebro organiza certas informações? E como ele consegue generalizá-las?

Cada nível de abstração encontrado no cérebro consiste de excitações neurais (ativações) de um subconjunto das características presentes, que não são mutualmente excludentes. Por conta disso, essas características, formam o que é chamado distribuição de representação: a informação encontra-se distribuida em vários neurônios [20]. Então, para certas características

ou composição delas (partes de objetos), deve-se analisar o conjunto de excitações dos neurônios à entrada.

Através de estudos em células do cortex visual dos gatos por Hubel e Wiesel [22] foram encontradas células simples e complexas. De acordo com estes experimentos, notou-se que algumas células são ativadas em resposta à certas propriedades mais simples de entrada nos sensores visuais, como por exemplo, orientação de bordas. Já outras células mostram maior invariância espacial do que às células anteriores, e isto, é uma das fontes de inspiração das arquiteturas de redes neurais profundas [23]. Na Figura 7, pode-se ver o modelo proposto por Hubel e Wiesel. Na natureza existem estruturas capazes de detectar características simples e características complexas, mas será que isto é feito de forma hierárquica, ou seja, as células simples precedem e geram informações para as células complexas?

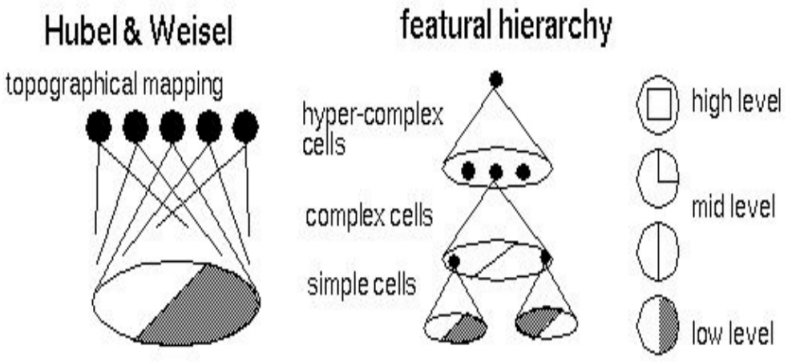

Figura 7: Hubel e Weisel - Fonte: Lehar, Steven [4].

O trabalho pioneiro em atenção visual, se chama *feature integration theory* [24] e foi proposto na psicologia, onde o objetivo era explicar a percepção e cognição do homem. A teoria defende a ideia de que a percepção de características vem antes da percepção de objetos. As características são registradas no campo visual, o cérebro processa tais características básicas, e então ajuda a atenção visual a formar o conceito do que seria um objeto. Características visuais diferentes como: cor, orientação, frequência espacial, brilho e direção, são registrados em mapas de características. A junção dos mapas de características formam um mapa para representar um possível objeto [25]. Ou seja, um modelo aceitável de como o cortex visual humano funciona, parte do aprendizado de características com níveis crescentes de complexidade até se chegar em uma abstração maior, como partes do objeto ou até mesmo o objeto completo.

Em visão computacional, pode-se extrair características de baixo nível como bordas (a partir de filtros), e depois utilizar de transformações nas imagens para detectar padrões mais

frequentes. Uma forma comum é utilizar o *pixel* e transformar em representações mais abstratas como presença de bordas, detecção de formas específicas, identificação de categorias. No final, deve-se agrupar todas estas informações para poder entender o cenário [20].

Como visto, existem células que percebem informações mais simples a partir da entrada bruta, e em seguida, a percepção de objetos ou partes deles por células que percebem informações mais complexas (seguindo uma hierarquia como proposto pela *feature integration theory*). Esta é a ideia por trás da aprendizagem profunda, que representa em uma hierarquia de camadas, independentemente do tipo de entrada, seja ela: som, imagem ou outro tipo de sinal. A Figura 8, mostra o trabalho desenvolvido por Zeiler e Fergus [5] para representar características de baixo, médio e alto nível.

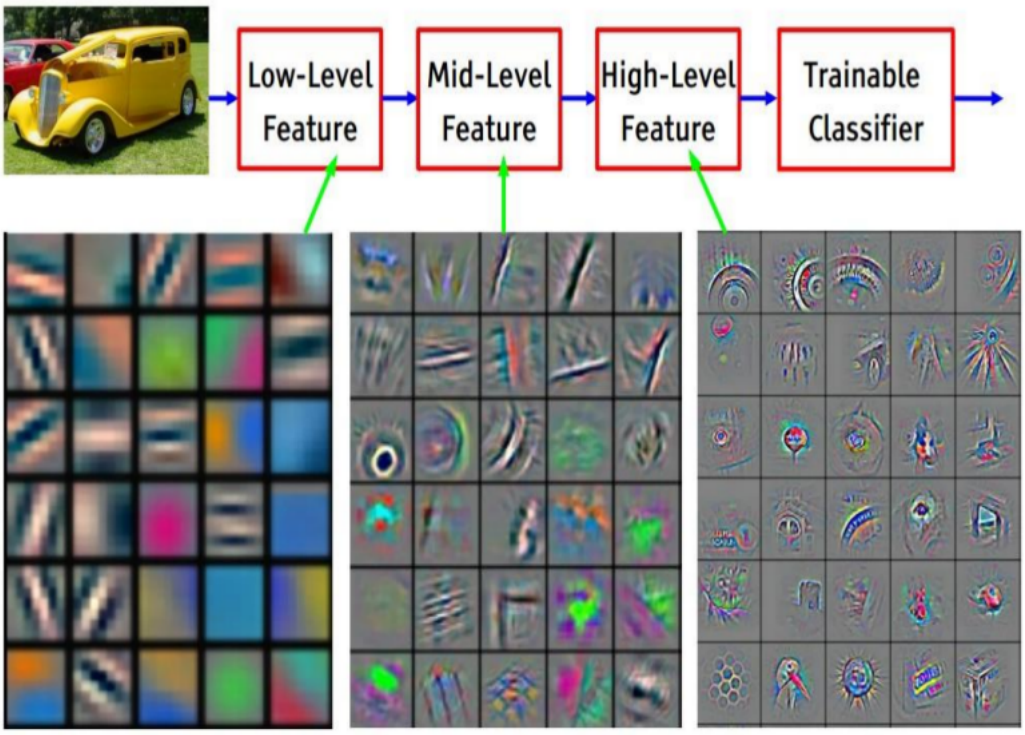

Figura 8: Visualização de características de uma ConvNet - Baixo, Médio e Alto Nível, Fonte: Zeiler e Fergus [5].

2.2.2 Rede Neural Convolucional (RNC) - *ConvNets*

As RNC's são inspiradas na estrutura do sistema visual. Atualmente, sistemas de reconhecimento de padrões baseados em redes convolucionais vem tendo bons resultados em termos de performance [26]. As redes convolucionais são classificadas como RNA's, pois cada neurônio recebe uma entrada e realiza uma operação de produto, opcionalmente podendo ser seguida

de outras operações como o aumento da não-linearidade. O fluxo normal de uma RNC tem o ínicio recebendo os *pixels* da imagem e termina com um vetor de probabilidades referente a entrada pertencer a classe.

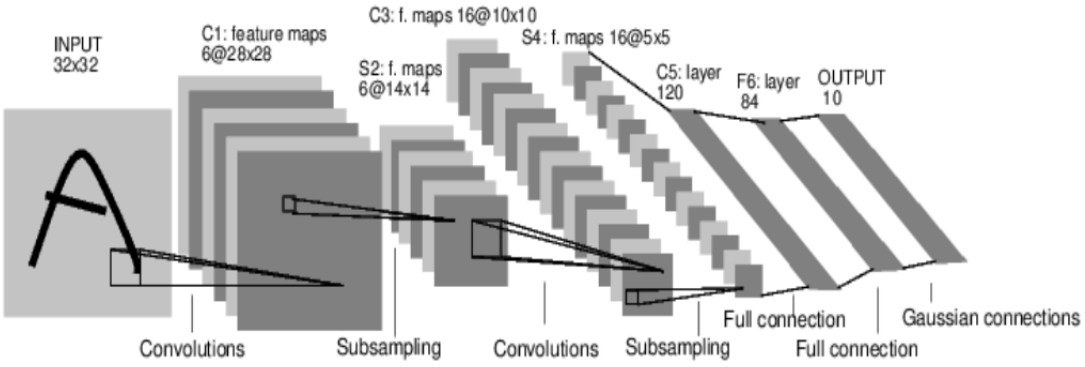

Figura 9: Rede Convolucional de LeCun *et al* (1998).

A RNC de LeCun *et al.* [27] (Figura 9) é organizada em camadas de dois tipos principais: camadas convolucionais e camadas de amostragem. Cada camada tem uma estrutura topográfica (cada neurônio é associado à posição do *pixel* da imagem), possuem um campo receptivo (região da entrada que influência na resposta do neurônio) e um conjunto de pesos de entrada [20].

Uma ConvNet é uma sequência de camadas, onde cada camada transforma um conjunto de ativações em outro conjunto de ativações através de funções. Os tipos principais de camadas de uma arquitetura ConvNet são: Camada Convolucional (Seção 2.2.2.1), Camada ReLU (Seção 2.2.2.2), Camada de Pooling (Seção 2.2.2.3) e Camada Totalmente Conectada (Seção 2.2.2.4).

2.2.2.1 Camada Convolucional

A primeira camada de uma RNC sempre é uma camada Convolucional. Para entender está camada precisamos entender o que é um filtro convolucional (às vezes chamado de neurônio ou kernel) e campo receptivo. O filtro convolucional é composto por uma matriz de pesos (NxN) e processa uma sub-região da imagem de mesmo tamanho (por exemplo 3x3, 5x5), o filtro convolucional é aplicado em todas as sub-regiões NxN da imagem. O campo receptivo é a região na qual o filtro está naquele momento. A Figura 10 mostra um exemplo de convolução com filtro 5x5, mapeando a entrada na saída.

Quando o filtro está passando por todas as subáreas (todos os campos receptivos) da imagem, o campo receptivo serve como um conjunto de valores chamados de pesos ou parâmetros.

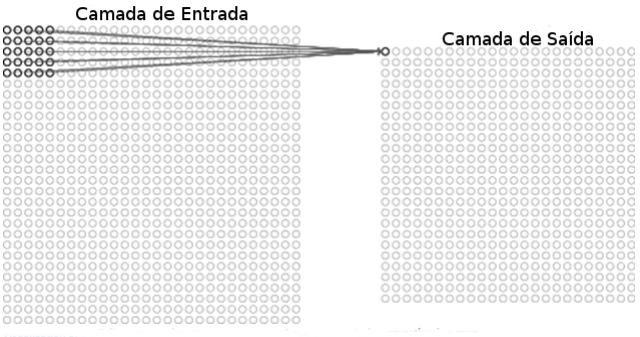

Figura 10: Convolução com filtro 5x5 onde a entrada é mapeada na saída (mapa de ativação), Fonte: Michael A Nielsen (2015) [6]

Enquanto o filtro está passando (*sliding*) ou convoluindo sobre a imagem de entrada, ele está multiplicando os valores do filtro pelos valores originais da imagem (elemento por elemento, começando o filtro pela borda acima e a esquerda). Estas multiplicações são somadas, obtendo apenas um número, este número é o representante daquela posição do filtro, e este processo é repetido por toda a imagem, movendo o filtro para a direita e é descrito pela Equação 7 de convolução discretizada para duas dimensões, onde F é o filtro, I é a imagem, J é o resultado da convolução, x e y são as posições do *pixel* atual, M e N são as dimensões do filtro. Quando todo este processo terminar, tem-se um array de valores que é chamado de mapa de ativação (*activation map*) ou mapa de características (*feature map*).

$$J(x,y) = F * I(x,y)) = \sum_{j=-M}^{M} \sum_{i=-N}^{N} F(i,j)I(x-i,y-j) \tag{7}$$

O tamanho da saída de uma camada convolucional, é controlado por três hiperparâmetros: profundidade, *stride* e *zero-padding*.

1. A profundidade, corresponde ao número de filtros que deseja-se usar, onde cada filtro é responsável por aprender algo sobre a entrada (cor, orientação, textura).

2. O *stride*, especifíca o quanto o filtro vai se mover a cada iteração. Por exemplo, um *stride* de 1, move de 1 em 1 *pixel*, enquanto o *stride* de 2, move de 2 em 2 *pixels*. Quanto maior o *stride*, menor será o tamanho espacial (x, y) da saída.

3. O *zero-padding*, é responsável por preencher a entrada com zeros ao redor da borda. Este hiperparâmetro, é responsável por preservar o tamanho espacial da entrada, na qual, a saída terá a mesma altura e largura da entrada.

Fazendo uma analogia, cada parte da saída pode ser interpretada como um conjunto de neurônios que olham somente para uma pequena região da entrada e compartilham seus parâmetros com todos os neurônios da esquerda e da direita. Há uma conectividade local, quando se está trabalhando com altas dimensionalidade como imagens, é praticamente impossível conectar todos os neurônios com os dados passados. Por isso, conecta-se cada neurônio com uma da região de entrada. O espaço de conectividade recebe o nome de campo receptivo do neurônio e define o tamanho do filtro.

Cada um destes filtros podem ser analisados como identificadores de características. Como por exemplo: linhas retas, padrões de cores, curvas. Por exemplo, uma imagem que possui curvas, quando passadas por um filtro detector de curvas, teria altos valores nas áreas em que as curvas estão presentes.

2.2.2.2 Camada *Rectified Linear Unit* (ReLU)

Depois de cada camada de convolução, é conveniente aplicar uma camada não-linear (ou camada de ativação) logo após. O propósito desta camada é introduzir não-linearidade no sistema que acabou de computar operações lineares durante as camadas de convolução (Operações de multiplicação e soma). Na Figura 11, podemos analisar o comportamento da função ReLU.

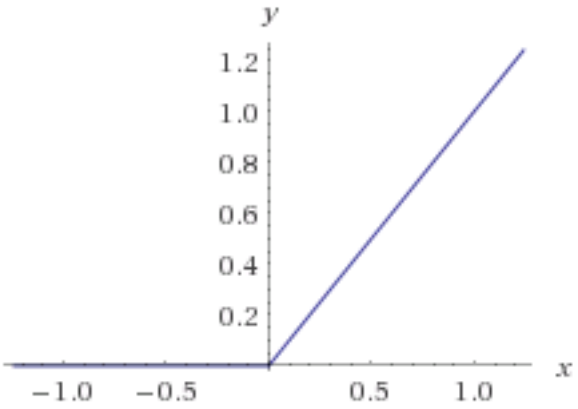

Figura 11: Comportamento da função ReLU - Fonte: Autor

No passado, funções não lineares como tanh e sigmoide eram usadas, mas Krizhevsky *et al.* [26] descobriram que as camadas de ReLU funcionam melhor porque são computacionalmente mais eficientes e não degradam significativamente a acurácia. Segundo Krizhevsky, a ReLU acelera consideravelmente a convergência do Gradiente descendente estocástico (SGD) comparado às funções sigmoide e tanh. Isso acontece devido a sua região linear, que não satura.

A ReLU é consegue inativar neurônios quando a soma ponderada de suas entradas é negativa, por isto, ela não propaga o erro durante a fase de *backpropagation*.

A camada de ReLU aplica a função de ativação exibida na Equação 8 sobre os valores de entrada. Esta camada, muda os valores negativos de ativação para 0, e também aumenta as propriedades de não-linearidade do modelo e toda a rede sem afetar o campo receptivo da camada de convolução.

$$f(x) = max(0, x) \tag{8}$$

2.2.2.3 Camada de *Pooling*

A camada de *pooling* é responsável por fazer uma operação de amostragem espacial, onde um filtro é passado pela imagem, e escolhida o valor, por exemplo, de máximo (*max pooling*). A camada de *pooling* é colocada entre as camadas de convolução, e a camada reduz (*downsampling*) o tamanho espacial da representação, reduzindo a quantidade de parâmetros e computações que são feitas na rede, controlando o *overfitting* (ajuste demasiado aos dados de entrada). Nesta categoria, geralmente tem várias opções de camadas, com *max pooling* sendo o mais popular. A intuição por trás desta camada é que se certa característica estiver no dado original, ele estará com o valor alto, então ele é importante. Como você pode imaginar, a camada reduz drasticamente a dimensão espacial (a largura e altura, mas não a profundidade) do volume de entrada. A Figura 12 ilustra o funcionamento da camada de *pooling* de máximo, onde a entrada é amostrada diminuindo o seu tamanho de 224x244 para 112x122, e visualizando os valores na matriz, a operação que é feita é a escolha dos maiores valores, considerando blocos de 2x2.

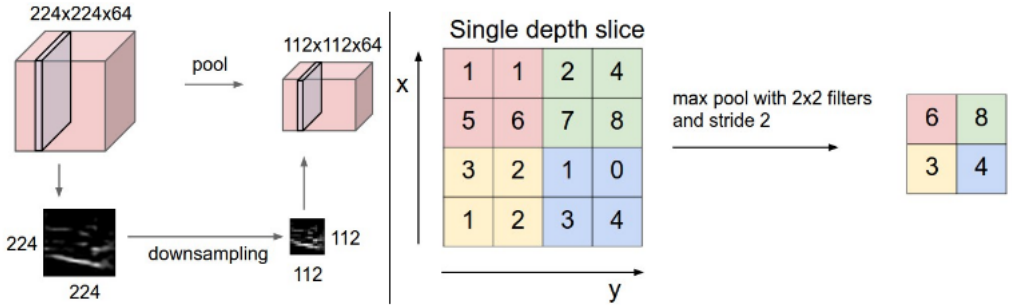

Figura 12: Pooling de máximo: reduz o tamanho dos dados - Fonte: Karpathy, A (2016) [7].

2.2.2.4 Camada Totalmente Conectada

A camada totalmente conectada calcula os *scores* (probabilidades) das classes, sua saída é o tamanho igual a quantidade de classes presentes no sistema, sendo atribuidos valores de 0 até 1, que indicam a probabilidade da entrada pertencer a certa classe. A Figura 13 ilustra uma RNC com suas camadas conectadas em sequência e no final há a presença da camada totalmente conectada.

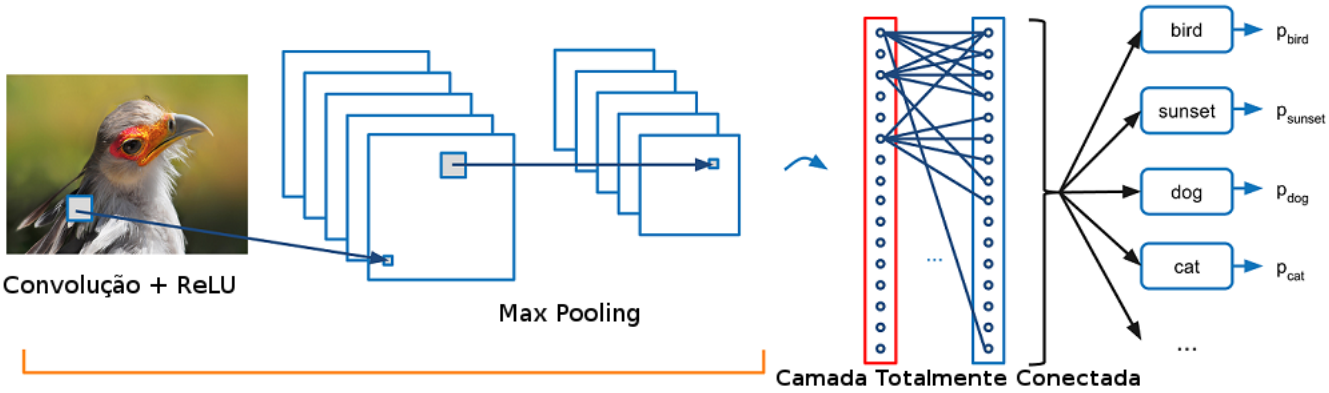

Figura 13: Camadas de uma CNN em sequência (Convolução, ReLU, Totalmente Conectada) - Fonte: Deshpande, A [8].

Deste jeito a camada totalmente conectada recebe como entrada a saída da camada anterior, que como descrito representa o mapa de ativação de características de mais alto nível (que no exemplo da Figura 13, poderia indicar a presença de uma pata, ou 4 pernas) e determina quais características são mais correlacionadas com uma classe particular. Por exemplo, espera-se que ela apresente que uma imagem é um pássaro, espera-se que ela apresente valores altos no mapa de ativação que representa características como patas, bicos, etc. A camada totalmente conectada, considera com maior peso as características de alto nível que são mais fortemente correlacionadas com uma certa classe, então o produto entre os pesos e a camada anterior computado, a saída obtida tende a aproximar a probabilidade esperada para diferentes classes [8]. Uma função *softmax* (Equação 9) é utilizada para transformar o valor em probabilidades que somam 100%.

$$\sigma(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad para\ j = 1, ..., k. \tag{9}$$

2.2.3 GoogLeNet - Rede Convolutiva Profunda (RCP)

Uma rede convolutiva profunda (RCP), é uma RNC com muitas camadas intermediárias. A rede convolutiva profunda GoogLeNet conseguiu os melhores resultados em tarefas de classificação e de imagem da competição *ImageNet Large-Scale Visual Recognition Challenge* 2014 (ILSVRC2014) [9]. A rede convolucional GoogLeNet [9], foi escolhida para os estudos propostos, porque contém o modelo já treinado na ferramenta Caffe, e por ter o módulo *Inception* que por ser uma arquitetura que vem funcionando bem, trás uma motivação para saber como realmente funciona o seu aprendizado. A principal diferença entre sua arquitetura e a das demais Redes Convolucionais (como por exemplo: AlexNet [26], ou a rede de LeCun *et al.* [27]) é a utilização de várias convoluções em paralelo, com filtros de diferentes tamanhos. Em outras redes, apenas uma convolução era realizada por etapa, com filtros de tamanho único. O classificador propriamente dito das RCP's (e também da GoogLeNet em particular) é um simples *softmax*, Equação 9.

A ideia de simplicidade das redes anteriores, é deixada um pouco de lado quando o Google apresenta o módulo *Inception*. Esta foi a primeira arquitetura que empilhou camadas convolucionais e de pooling no topo de cada em uma estrutura sequencial. O custo computacional de se fazer isto aumentou bastante e também aumenta a chance de *overfitting*. No caso das Redes Convolutivas Profundas, o termo profundo é referente ao empilhamento de várias camadas intermediárias, tornando a rede extensa. Já para dimensionalidade de um conjunto de dados, o termo profundo é referente à uma dimensão onde é empilhado as respostas dos filtros convoluidos, por exemplo: se a resposta de dois filtros 3x3, 4x4, forem empilhadas em uma única saída, sua profundidade teria dimensão dois.

Em cada camada de uma RNC tradicional, deve-se fazer uma escolha entre uma operação de *pooling* ou uma operação de convolução e também o tamanho do filtro. O que o módulo *Inception* permite é fazer todas essas operações em paralelo. Desta maneira, a rede gera uma saída com a dimensão muito grande, porque cada filtro gera uma saída. Porém, Szegedy *et al.* propuseram operações de convolução 1x1 antes das camadas 3x3 e 5x5. A convolução 1x1 (ou também rede dentro de uma rede) provê um método de diminuição de dimensionalidade. Isto também pode receber o nome de *pooling of features*, porque estamos diminuindo a profundidade da saída, de forma similar a uma redução da largura e da altura, porém para a profundidade. O *Inception* também tem operação de *pooling* para ajudar reduzir altura e largura, e combater *overfitting*. No topo de tudo isso, temos depois de cada camada de convolução, ReLUs que

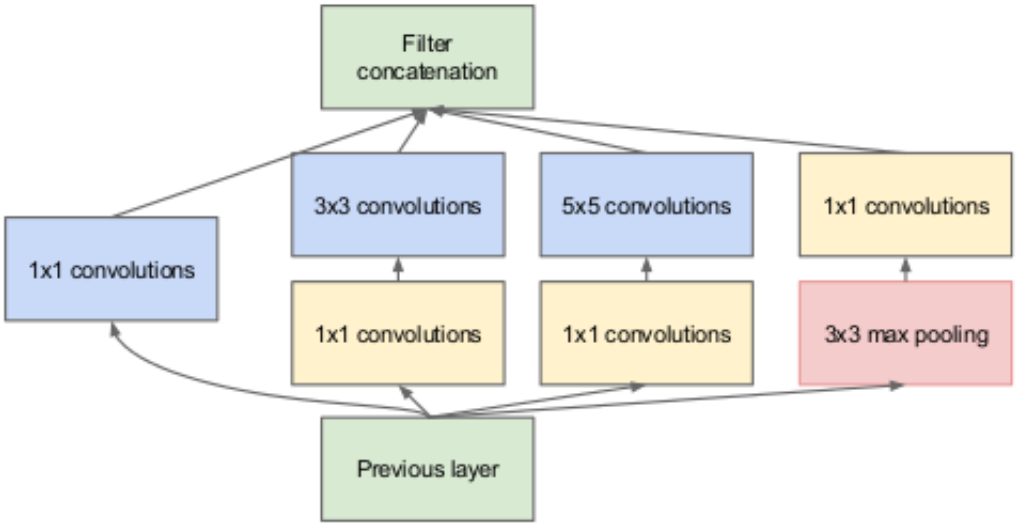

Figura 14: Módulo Inception com redução de dimensão - Fonte: Szegedy *et al.* [9]

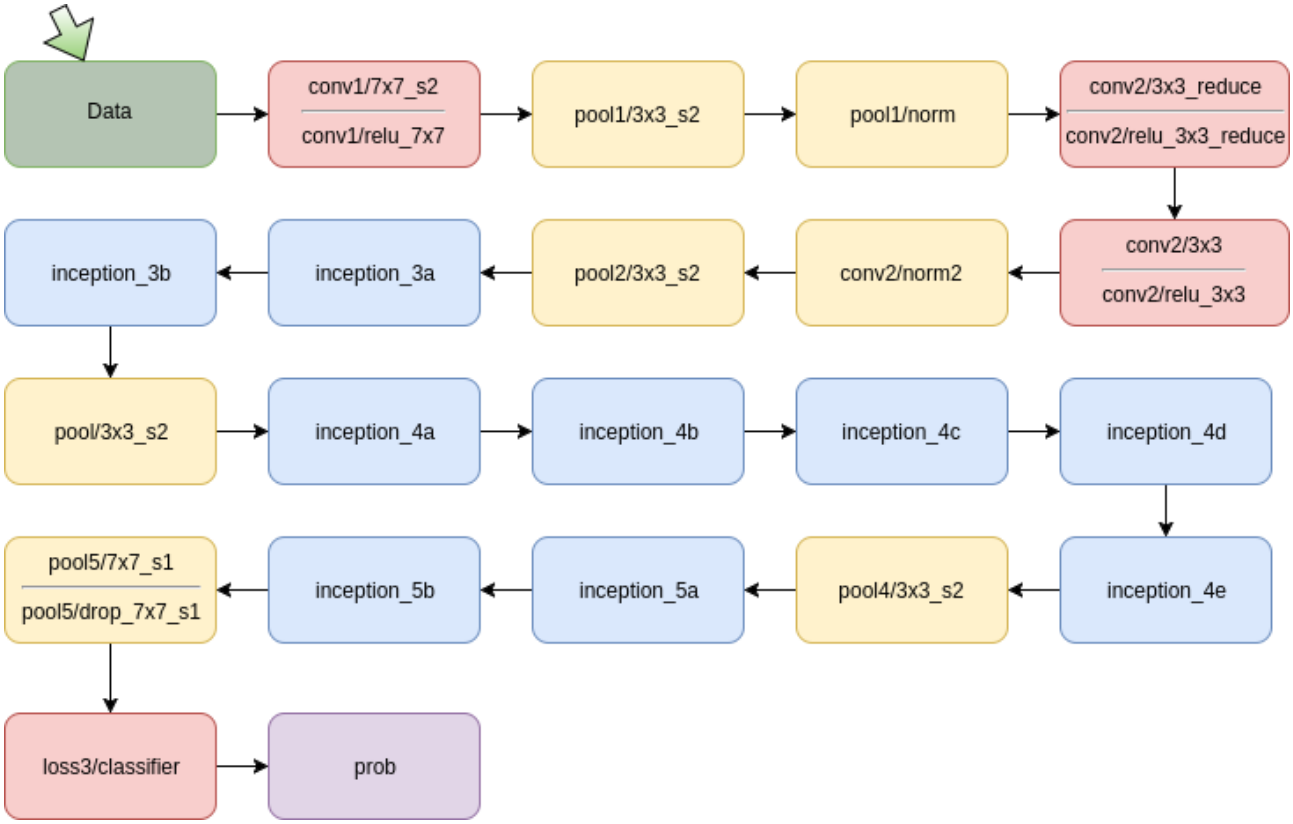

Figura 15: Arquitetura da rede GoogLeNet - Fonte: Autor

introduzem a não-linearidade na rede.

2.2.4 Rede de Neural de Deconvolução (RND) - DeConvNet

Mesmo uma RNC demonstrando uma performance notável em tarefas de reconhecimento de objetos, não se tinha ideia do porque elas conseguiam se sair tão bem nestas atividades. Até que Zeiler e Fergus [5], deram este passo com o as Redes Neurais de Deconvolução, e os pesquisadores começarem a entender o funcionamento das camadas intermediárias das RNCs e propor melhorias. A rede de deconvolucão, é uma ferramenta que permite a construção da representação de imagens hierárquicas. Estas representações podem ser usadas tanto para tarefas de baixo nível como remoção de ruídos, como também prover características para reconhecimento de objetos. Adotando a mesma técnica de uma rede de convolução, a rede de deconvolução é capaz de gerar uma representação estável de cada nível, preservando a localidade. Usando os mesmos parâmetros para aprender cada camada, a RND pode automaticamente extrair características ricas que correspondem a conceitos de níveis intermediários como junção de bordas, linhas paralelas, curvas e elementos geométricos básicos, como retângulos. Na Figura 16, podemos ver uma imagem que foi reconstruida através da técnica de deconvolução.

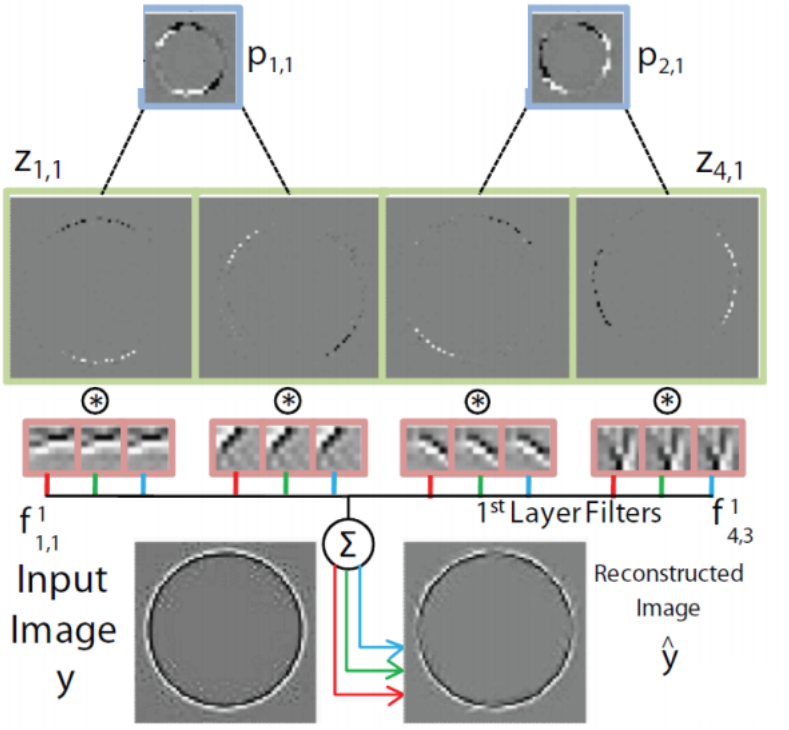

Figura 16: Reconstrução de uma imagem usando DeconvNet - Fonte: Karpathy, A (2016) [7].

A ideia básica por trás de como funciona a DeConvNet é que cada camada de uma RNC treinada, é conectada a DeConvNet. A imagem de entrada alimenta a RNC e suas ativações são calculadas em cada nível. Este é o passo de avanço. Agora, suponha que se queira olhar

para as características na quarta camada de convolução. Então deve-se guardar as ativações deste mapa de características, mas ajustar todos os outros conjuntos de ativação na camada para 0, e depois passar isto para o mapa de características como entrada para a DeConvNet. Esta DeConvNet tem os mesmos filtros da RNC original. A entrada passa por uma série de etapas de *max pooling reverso* (*unpool* - Seção 2.2.4.1), retificação e operações de filtros para cada camada precedente, até que a entrada seja alcançada. A razão por trás deste processo é que queremos analisar qual tipo de estrutura é ativada dado um mapa de características [21]. Isto permite que a reconstrução da ativação na camada abaixo, mostre a ativação da camada escolhida. O uso de RND tem se mostrado com sucesso para atividades de visualização das camadas intermediárias de uma RNC [10]. Na Figura 17, é aprensentado um fluxograma de uma DeConvNet associada a uma RNC, onde no lado esquerdo de cima para baixo, está a DeConvNet e suas operações e no lado direito está a RNC e suas operações.

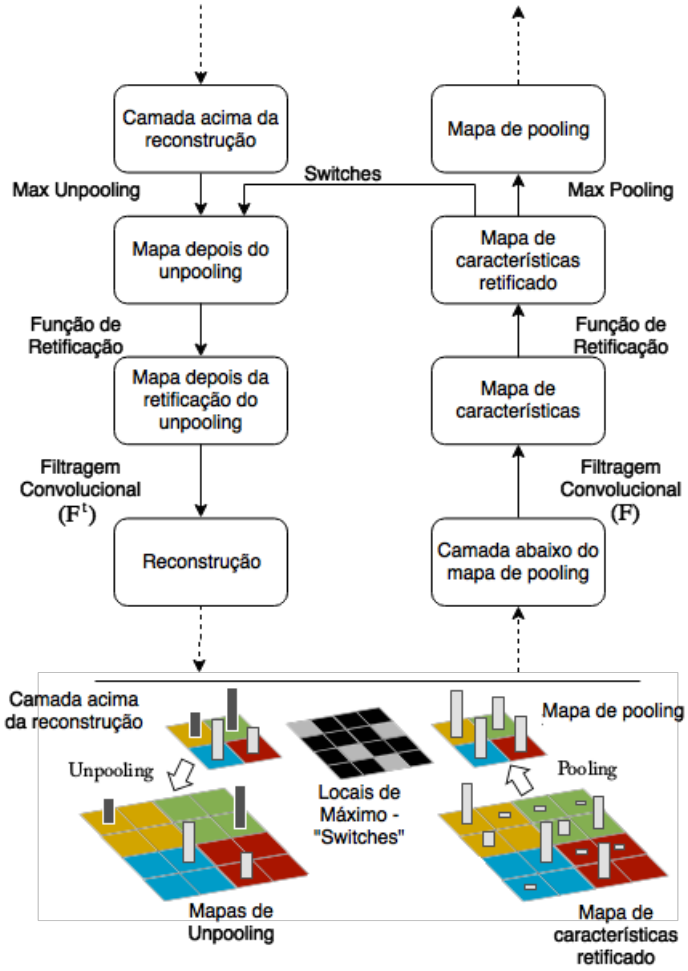

Figura 17: Lado esquerdo: DeConvNet e Lado Direito: ConvNet - Fonte: Zeiler e Fergus (2013) [5]

2.2.4.1 *Unpooling - max pooling reverso*

A camada de *unpooling* em redes deconvolucionais, é a operação inversa do *pooling*, que reconstrói as ativações em seus tamanhos originais. Durante a etapa de *unpooling*, uma operação não-invertível do *max pooling* na RNC, a rede de deconvolução reconstrói uma versão aproximada das características da RNC da camada anterior. A etapa é resolvida por uma aproximação inversa, onde os locais de máximo em cada região de *pooling* são guardados e ajustados como váriaveis de "switches". Estes switches são usados para saber o local da reconstrução da camada de acima no local apropriado, preservando a estrutura do estímulo. Na Figura 18, tem-se a ilustração da arquitetura dessa técnica [28].

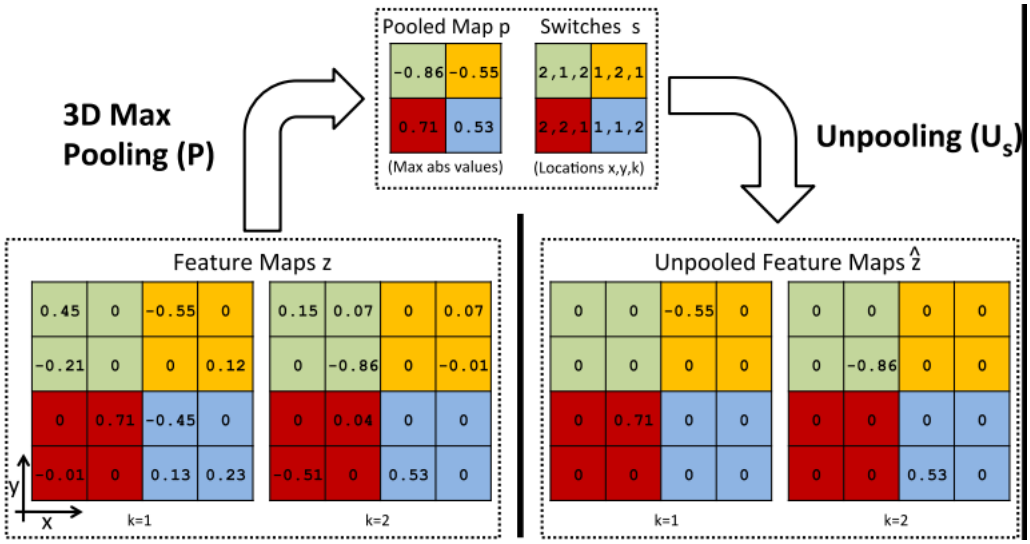

Figura 18: Visualização da técnica de *unpooling* - Fonte: Karpathy, A (2016) [7].

2.2.4.2 Deconvolução

A saída de uma camada de *unpooling* é um mapa de ativações esparso. O trabalho da camada de deconvolução é tornar as ativações mais densas através de operações de convolução com múltiplos filtros que foram aprendidos. Entretanto, contrário ao que as camadas de convolução fazem, que é conectar várias ativações de entrada com um filtro de janela para uma única ativação, a camada de deconvolução associa uma unica ativação a várias na saída. A saída de uma deconvolução é um mapa de ativações maior e mais denso do que se tinha na entrada. Os filtros aprendidos para realizar uma deconvolução, nada mais são do que os filtros aprendidos na convolução, porém transpostos. Na Figura 19, podemos ver o processo de *unpooling* e o de *deconvolução* [28].

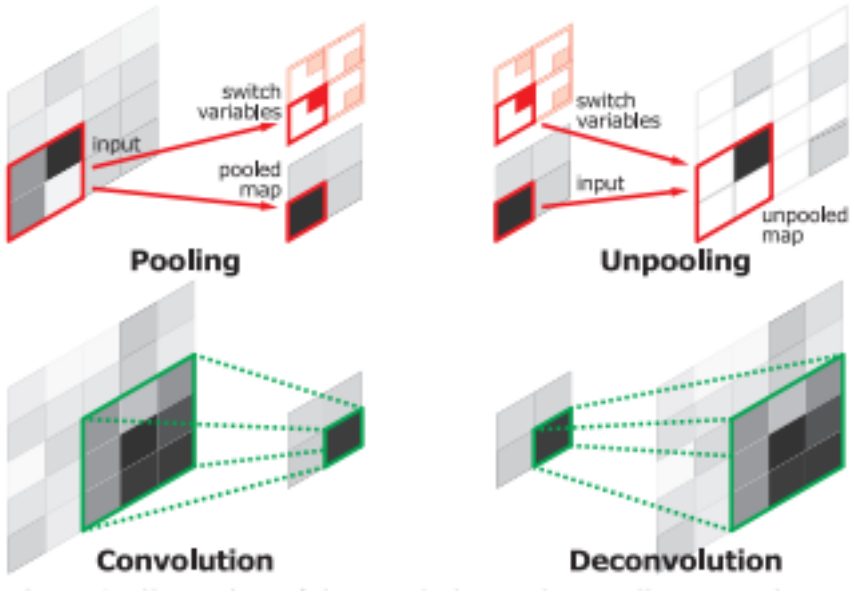

Figura 19: Visualização dos processos de *Pooling*, *Unpooling*, *Convolução* e *Deconvolução* - Fonte: Karpathy, A (2016) [7].

2.2.5 *Transfer Learning*

Dados são importantes para treinar uma rede, e a ideia de *Transfer Learning* (transferência de aprendizagem) tem ajudado a demanda destes dados. Transferência de aprendizagem é o processo de pegar um modelo de rede pré-treinado (os pesos e parâmetros da rede já foram treinados para um tamanho muito grande de dados por alguém) e "calibrar" o modelo para sua base de dados. A ideia é que um modelo pré-treinado vai atuar como um extrator de características. Em seguida, remove-se a última camada da rede e troca pelo classificador (dependendo do que se tratar o problema). Então tem-se os valores de pesos do gradiente e da otimização já inicializados, com isto, basta treinar a rede para a base específica [29] [30] [31].

Quando se pensa nas camadas de baixo nível da rede, sabe-se que vão atuar como detector de características como bordas e curvas. A menos que nosso problema seja muito específico, a rede que deseja-se treinar também vai precisar de um detector de bordas e curvas. Ao invés de ter que treinar a rede inteira sobre valores aleatórios iniciais dos pesos, pode-se utilizar valores de pesos pré-treinados, e focar nas camadas mais importantes, que são as de mais alto nível para treinar. Se a base de dados é muito diferente de algo como a ImageNet [32], então deve-se treinar mais as camadas e não utilizar todos os pesos já pré-treinados das camadas de mais baixo nível.

2.3 Mapa Auto-Organizável Seletivo de Dimensões com Campo Receptivo Localmente Adaptativo (LARFDSSOM)

As redes SOM, conforme descrita na Seção 2.1.3.1, possuem algumas limitações que foram tratadas por trabalhos mais recentes. Dentre elas:

1. Não funcionam para agrupamento em subespaços, quando é necessário que cada agrupamento seja capaz de levar em consideração mais fortemente um subconjunto diferente das dimensões da entrada e um padrão pode ser atribuído a mais de um grupo.

2. Definir o número de nodos e a topologia requer conhecimento a priori a respeito do conjunto de dados, para que se possa feito um agrupamento adequado.

Foram propostos o DSSOM [3] (Seção 2.3.1) para o problema número 1, já o problema 2 foi tratado por outros métodos na literatura, e estes métodos inspiraram a expansão do DSSOM, que se chama LARFDSSOM (Seção 2.3.2)[33]. O Mapa Auto-Organizável Seletivo de Dimensões (DSSOM) foi proposto para o agrupamento de subespaços, o qual permite a cada neurônio atribuir importâncias diferentes a cada dimensão para agrupar os padrões. Esses diferentes níveis de importância são aprendidos ao longo do treinamento, assim como os vetores de pesos. Apesar de ser efetivo para agrupamento em subespaços, seu treinamento é difícil de parametrizar e sua topologia fixa torna difícil adaptar o resultado ao conjunto de dados. Por isto, foi proposto uma melhoria que é Mapa Auto-Organizável Seletivo de Dimensões com Campo Receptivo Localmente Adaptativo (LARFDSSOM), uma extensão de topologia dinâmica do DSSOM na qual neurônios são inseridos e removidos durante o treinamento, e as conexões entre eles são definidas em função da semelhança entre os seus campos receptivos. Em relação ao DSSOM, o LARFDSSOM possui custo computacional reduzido, parametrização simplificada e melhor qualidade do agrupamento. Por isto, o trabalho proposto utiliza o LARFDSSOM.

2.3.1 DSSOM

Na determinação do neurônio vencedor no SOM, cada dimensão do espaço de entrada tem a mesma importância. Mas no DSSOM, cada neurônio pode atribuir uma relevância diferente para cada dimensão no cálculo da distância. Especificamente, dado um padrão X, a distância ponderada entre o vetor de pesos de i e o padrão X é dada pela Equação 10, onde ω_{ij} é a

relevância dada pelo neurônio i à dimensão j e m é o número de dimensões da entrada de dados:

$$D_w(X, i) = \sqrt{\sum_{j=1}^{m} \omega_{ij}^2 (X_i - W_{ij})^2} \tag{10}$$

A ativação do neurônio é calculada pela Equação 11, onde ε é um valor muito pequeno para evitar divisão por zero e Sum_i é a soma das componentes w_i:

$$ac(X, i) = \frac{Sum_i}{D_w(X, i) + Sum_i + \varepsilon} \tag{11}$$

Durante o treinamento da rede, para cada neurônio i é mantida uma média móvel δ_{ij} da distância na dimensão j entre o vetor de pesos W_i e os padrões agrupados por i. Então, w_i é calculado em função de δ_i, onde cada valor w_{ij} é inversamente proporcional à variância na dimensão j dos padrões agrupados pelo neurônio i. Durante as fases de treinamento ou agrupamento final, cada padrão escolhido é apresentado à rede várias vezes, e um vencedor diferente é escolhido toda vez. A atualização dos vetores de peso é semelhante à do SOM, e a ativação dos neurônios a partir do segundo vencedor é calculada de forma diferente. Para mais detalhes consultar [3].

2.3.2 LARFDSSOM

Nas redes SOM já mencionadas, a quantidade de neurônios e a estrutura formada por suas conexões são fixas. Já no LARFDSSOM [3], os neurônios são inseridos e removidos durante o treinamento, e as conexões entre eles são atualizadas periodicamente. Cada neurônio i, assim como no DSSOM (Seção 2.3.1), está associado a três vetores de mesma dimensão do espaço de entrada, além de um contador de vitórias nas competições ($wins_i$) que é usado para escolher quais neurônios serão removidos:

1. Vetor de pesos W_i
2. Vetor de relevâncias w_i
3. Vetor de distâncias δ_i (usado para calcular o vetor de relevâncias)

O treinamento do LARFDSSOM tem possui fases: organização e convergência. A rede começa com apenas um neurônio, e um padrão da entrada é escolhido aleatoriamente para ser

seu vetor de pesos. O vetor de distâncias é inicializado com zeros e o de relevâncias com uns. Na fase de organização há $tmax * n$ competições, onde n é o número de padrões de entrada e $tmax$ um parâmetro de entrada do algoritmo. Em cada competição, assim como nas redes SOM anteriores, um padrão da entrada X é escolhido aleatoriamente e apresentado à rede. A ativação de cada neurônio i é dada pela Equação 12:

$$ac(X, i) = \frac{1}{1 + D_w(X, i)/(\|w_i\|^2 + \varepsilon)} \tag{12}$$

onde ε é um valor muito pequeno para evitar divisão por zero e $(\|w_i\|^2 + \varepsilon)$ é a norma do vetor de relevâncias. A distância calculada no LARFDSSOM é ponderada e segue a Equação 13:

$$D_w(X, i) = \sqrt{\sum_{j=1}^{m} w_{ij}(X_i - W_{ij})^2} \tag{13}$$

Quando se tem um vencedor i, ele tem seus vetores de pesos e o dos seus vizinhos atualizados, do mesmo modo que uma rede SOM normal, com a Equação 5 (Seção 2.1.3.1). Onde, e é a taxa de aprendizagem e satisfaz a Equação 14:

$$e = \begin{cases} e_b, \text{ se for um neurónio vencedor} \\ e_n, \text{ caso contrário} \end{cases} \tag{14}$$

Sendo en é sempre maior que eb. Depois, os vetores de distância são atualizados segundo a Equação 15, onde β é um parâmetro que regula a velocidade de mudança da média móvel:

$$\delta_i = \delta_i + e\beta(|X - W_i|) - \delta_i) \tag{15}$$

Por fim, os vetores de relevância são atualizados, através da Equação 16:

$$w_{ij} = \begin{cases} \frac{1}{1+exp(\frac{\delta_{ij}-\delta_{imean}}{s(\delta_{imax}-\delta_{imin})})} & \text{se } \delta_{imin} \neq \delta_{imax} \\ 1 & \text{caso contrario} \end{cases} \tag{16}$$

Onde, δ_{imin} é a distância mínima encontrada no vetor de distâncias δ_i, δ_{imax} é distância máxima, δ_{imean} é a média das distâncias. Já o s é um parâmetro para controlar a suavidade do vetor de relevâncias. É importante observar que, em [33], a fórmula acima está com um erro:

δ_{ij} é subtraído de δ_{imean}, ao invés do contrário.

No LARFDSSOM há uma diferença em relação aos outros já citados: se a ativação do vencedor da competição não atingir um valor mínimo dado pelo parâmetro at, ele e seus vizinhos não são atualizados. Ao invés disso, um novo neurônio é criado, com seu vetor de pesos igual ao padrão e seu vetor de distâncias é preenchido com zeros e o de relevâncias com uns. Ele é então conectado aos neurônios existentes e inserido na rede. Mas se o número de neurônios estiver no máximo, definido pelo parâmetro $nmax$, nenhum neurônio é criado.

Cada neurônio i mantém seu contador $wins_i$ de quantas competições ele ganhou. A cada $maxcomp$ competições, todos os neurônios com menos de $lp * maxcomp$ vitórias são removidos da rede, onde lp e $maxcomp$ são parâmetros do algoritmo para controlar o tamanho mínimo dos grupos e a periodicidade dos resets, respectivamente. Observa-se que o parâmetro $maxcomp$ é multiplicado pelo número de padrões da entrada antes de ser utilizado no algoritmo. Depois das remoções, as conexões dos neurônios restantes são atualizadas e seus contadores são resetados para zero. Quando um neurônio é criado, ele recebe $lp * wins$ vitórias, onde $wins$ é o número de competições desde o último reset.

Quando as conexões de um neurônio são criadas ou atualizadas, ele é conectado a neurônios que dêem importâncias similares para as dimensões de entrada. Especificamente, dois neurônios i e j são conectados se for satisfeita a Equação 17, sendo m o número de dimensões de entrada e c parâmetro para regular a conectividade:

$$\|w_i - w_j\| < c\sqrt{m} \tag{17}$$

Depois da fase de organização vem a de convergência, que serve para dar um ajuste final aos neurônios restantes. O parâmetro $nmax$ é atualizado com n, a quantidade atual de neurônios, e mais $maxcomp$ competições são realizadas, até uma última fase de remoção. Finalmente, para definir quais padrões pertencem a quais grupos, para cada padrão é calculada a ativação dos neurônios, e ele é atribuído a todos cuja ativação atinja o limiar at. Se o algoritmo estiver configurado para realizar o agrupamento projetivo, onde cada padrão pode pertencer apenas a um grupo, então cada padrão é associado apenas ao neurônio de maior ativação.

3 Metodologia

Neste capítulo, será descrita a metodologia utilizada para o desenvolvimento do trabalho, começando pela escolha da base de dados (Seção 3.1). Em seguida, é descrito como foi feita a implementação (Seção 3.2), que descreve as ferramentas e métricas utilizadas. Por fim, encerra-se o capítulo com a conclusão sobre a utilização das ferramentas para obter os resultados propostos (Seção 3.3.3).

3.1 Base de dados - *Dataset Easy*

Para avaliar a qualidade dos métodos de forma imparcial, costuma-se usar bases de dados (conjunto de dados) que são utilizados na área e já estão consolidados (amplamente divulgados e testados). Com as bases dados disponíveis, é possível realizar comparações de técnicas, além de facilitar os testes, pois evita o retrabalho de preparar os dados e também rotulá-los, se for necessário (por exemplo: aprendizagem supervisionada). Tuytelaars *et al.* [1] realizou um estudo sobre técnicas de agrupamento em diversos bancos de dados, neste estudo, foi proposto o *Dataset Easy*, que obteve bons resultados. Por isto, no trabalho, foi escolhido o *Dataset Easy* que é um subconjunto do famoso base de dados Caltech 256 [34]. O Caltech 256, é composto por 256 categorias de objetos, com várias imagens de cada categoria. Para reduzir a sobreposição de um determinado conjunto em outro, o *Dataset Easy*, contém apenas 20 categorias que foram escolhidas manualmente com base na distribuição das características visuais. As categorias são relacionadas à objetos com apenas um rótulo, segundo Tuytelaars *et al.*, evitando imagens com mais de uma possível categoria. Um dos problemas apresentados pelo *Dataset Easy*, é o fato de ser desbalanceado em termos da quantidade de imagens por categoria. Fica evidente quando é analisado a categoria *faces easy* 101, que possuem 435 imagens de faces, enquanto categorias como a *American flag*, possuem apenas 97 imagens. Para evitar este problema, foram escolhidas as 50 primeiras imagens de cada categoria do *Dataset Easy*. A Tabela 1 contém o nome das categorias presentes no *Dataset Easy*, totalizando 1000 imagens, sendo 50 imagens de cada uma das 20 categorias.

3.2 Implementação

Para realização dos experimentos foi escolhido a ferramenta de *deep learning* Caffe [35], que foi desenvolvida pela Berkeley AI *Research*(BAIR) e é de código aberto. A escolha do Caffe se

Tabela 1: *Dataset Easy* - 20 categorias de objetos selecionadas por Tuytelaars *et al.* [1]

American flag	diamond ring	dice	fern	fire extinguisher
fireworks	French horn	ketch 101	killer whale	leopards 101
mandolin	motorbikes 101	pci card	rotary phone	roulette wheel
tombstone	Pisa tower	zebra	airplanes 101	faces easy 101

deu por ser uma ferramenta rápida, modular e bem documentada com vários tutoriais e material de apoio. O Caffe tem modelos de Redes Neurias Convolucionais já treinados, facilitando os testes. O trabalho foi desenvolvido utilizando a linguagem de programação Python 2.7 (pyCaffe), e foi escolhido o modelo de Rede Convolucional GoogLeNet [9] (2.2.3) para os estudos propostos, porque a GoogLeNet contém o modelo já treinado na ferramenta Caffe. Em seguida, foi necessário preparar o banco de dados como mencionado na Seção 3.1.

A implementação foi sub-dividida em duas etapas, que posteriormente serão conectadas: Etapa de agrupamento, que utiliza o LARFDSSOM (Seção 3.2.1) e Etapa de Visualização que utiliza DeConvNet (Seção 3.2.2).

3.2.1 Etapa LARFDSSOM

O modelo disponível no Caffe do GoogLeNet foi treinado pelos autores com as imagens da ImageNet [32] que é uma base de dados com 1024 categorias, e por isto, utilizou-se da técnica de *Transfer Learning* (Seção 2.2.5) para extrair as características necessárias da penúltima camada, ou seja, removeu-se o classificador existente, e com isto, a rede pré-treinada é vista como um poderoso extrator de características, que serão fornecidas para agrupamento pelo LARFDSSOM (Seção 2.3.2). Escolheu-se a penúltima camada da GoogLenet por conter apenas 1024 características, o que facilitou os testes, existem camadas com mais de 100000 características, porém demandaria muito poder e tempo de processamento. A escolha de uma rede pré-treinada, se deu por facilitar os testes e por conta do tempo de desenvolvimento do trabalho, pois, seria necessária uma etapa de ajuste da rede. Na Figura 20, é apresentada a arquitetura utilizada para a etada LARFDSSOM.

Em seguida, foram realizadas análises para se obter um bom agrupamento das categorias. Na Seção 4, são relatados os testes realizados. Os resultados da busca de parâmetros, etapa necessária para calibrar os parâmetros do LARFDSSOM, e os resultados do *Clustering Error* estão presentes na Seção 4.1. Já na Seção 3.3.2 é descrita a matriz de confusão, e os respectivos resultados (Seção 4.2). Com estas métricas, foi possível realizar os testes e fazer a análise

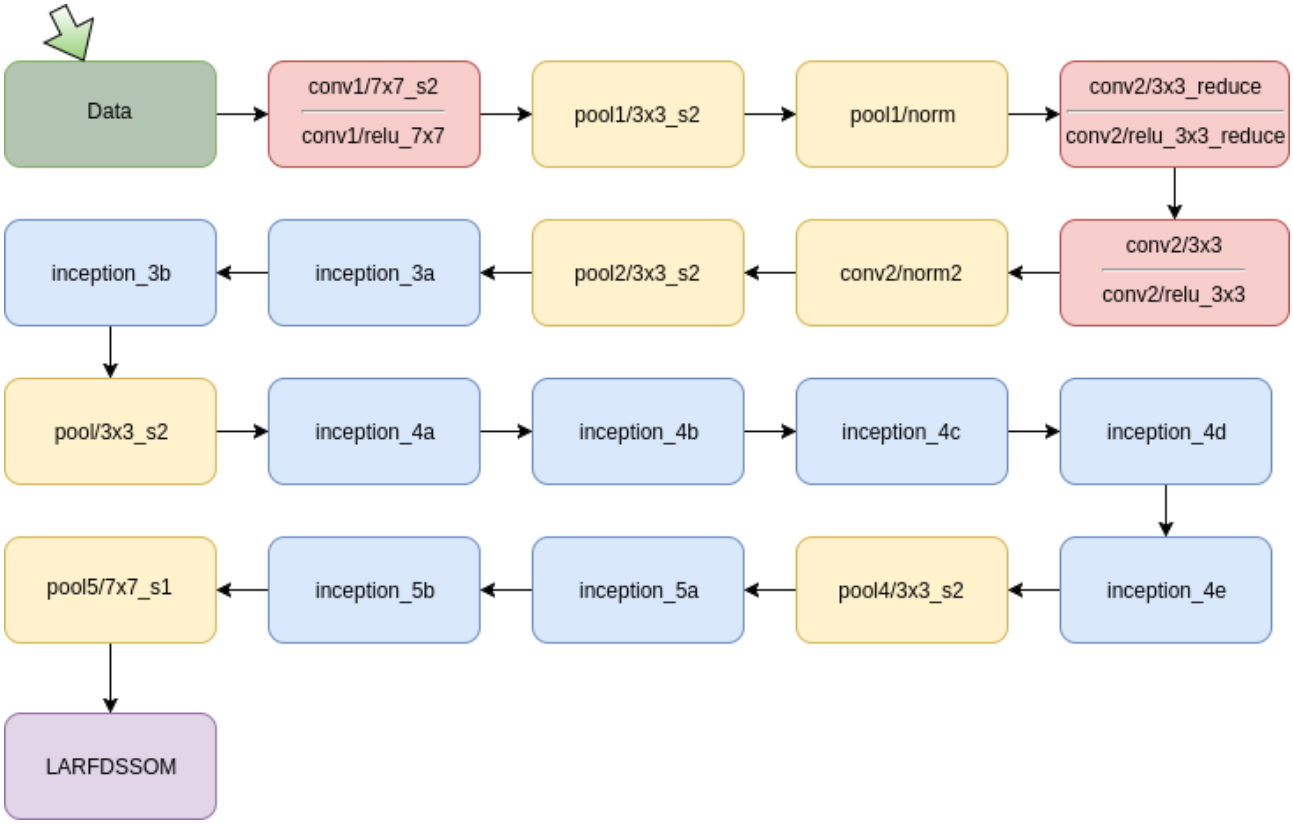

Figura 20: Arquitetura da rede GoogLeNet retirando a camada do classificador e inserindo o LARFDSSOM - Fonte: Autor

das relevâncias dos *clusters*, assim como, obter as maiores relevâncias para comparar com os neurônios dados pela etapa de visualização.

3.2.2 Etapa de Visualização

Para auxiliar nesta etapa, foi utilizado a ferramenta escrita em Python e de código aberto *Deep Visualization Toolbox* [36], que contém a implementação de Redes Neurais de Deconvolução (Seção 2.2.4). A ferramenta possibilita visualizações como: imagens geradas pelo passo de avanço para visualizar as ativações, imagens geradas pelo processo de deconvolução, imagens que geraram ativação máxima no neurônio e imagens de deconvolução da ativação máxima. Porém, a ferramenta foi escrita apenas para a rede Caffenet-yos, tendo sido necessário fazer adaptações no código para o correto funcionamento na rede GoogLeNet. A Figura 21, ilustra a adaptação desta monografia. As visualizações das imagens geradas por nodo é um processo lento e tem que ser feito camada por camada da rede. Adaptar as maiores imagens de deconvolução de cada nodo, para visualização dentro da ferramenta, seria um processo complicado, atualmente isto é feito externamente à ferramenta original. O ferramental permitiu que com-

parações fossem feitas a cerca do que a rede está realmente aprendendo, e permitiu realizar a análise das relevâncias, junto com a etapa do LARFDSSOM.

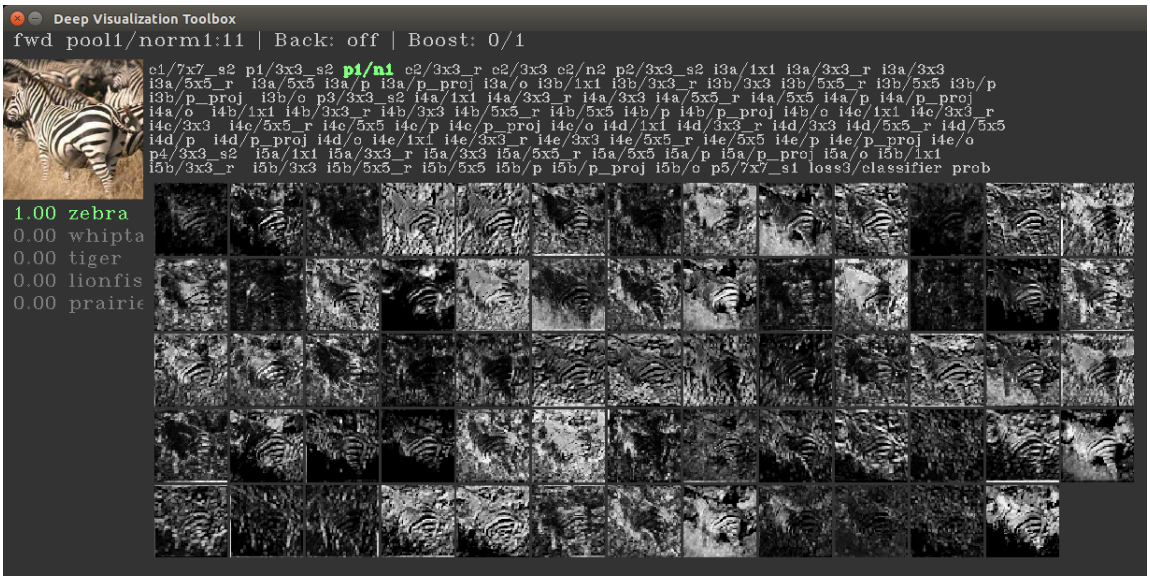

Figura 21: *Deep Visualization Toolbox* adaptado - Fonte: Autor

3.3 Métricas

Nesta Seção, serão apresentadas as métricas utilizadas no projeto, servindo como ferramenta auxiliar para calibrar parâmetros e poder fazer os comparativos propostos pelo trabalho.

3.3.1 *Clustering Error* - CE

Dentre as métricas descritas e analisadas em Patrikainen e Meila (2006) [37] e em Müller *et al.* (2009) [38], foi escolhido o *Clustering Error*(CE) para avaliar a qualidade dos agrupamentos realizados pelo o LARFDSSOM. O CE é uma métrica que leva em conta não apenas os agrupamentos produzidos, mas também as dimensões relevantes encontrados para cada grupo e penaliza resultados com agrupamentos demais. O CE é calculado como uma percentagem de pontos que estão agrupados de maneira diferente, considerando uma coincidência ótima entre os agrupamentos desejados e os obtidos. A métrica utilizada varia entre zero e um, e valores mais altos indicam melhores agrupamentos. O CE pode ser visto como uma generalização do erro de classificação, o qual é frequentemente utilizado para avaliar métodos de agrupamento clássico. De fato, conforme a porcentagem de dimensões relevantes aumenta, CE converge para o complemento do erro de classificação (1 - erro de classificação). Quando tem-se um valor de

CE baixo, deve-se ajustar os parâmetros do algoritmo de agrupamento para que ele tenha uma CE alto e realize a tarefa com melhor qualidade.

3.3.2 Matriz de Confusão

A matriz de confusão é uma ferramenta de visualização que apresenta a performance de classificação ou agrupamento por categoria de aprendizagem [39]. Netes trabalho, para a parte do LARFDSSOM, foi utilizada uma matriz (20x20), que em cada linha tem-se os *clusters* que foram gerados, e em cada coluna, os rótulos representando as categorias da base de dados utilizada. Com este tipo de abordagem, pode-se ver quais *clusters* são bons para identificar as respectivas categorias e também avaliar (Seção 4) o quão bom foi o agrupamento. A matriz foi calculada através das 1000 imagens que serviram de entrada para o LARFDSSOM, e depois foram analisados os *clusters* que ele selecionou para cada imagem separada com a categoria real da imagem (rótulo da categoria). Para a etapa de visualização, como a rede GoogLeNet já é previamente treinada, foi calculada uma matriz de confusão, mas ela não faz muito sentido porque existem 1024 categorias da ImageNet [32], e algumas delas não correspondiam com as categorias do *dataset Easy*. Inicialmente, escolheu-se a classe (zebra), que teve resultados bons em ambas etapas e fez-se um estudo sobre ela. No Capítulo 4, é apresentado os resultados de tal análise, inclusive, a taxa de acerto da classe zebra para a etapa de visualização.

3.3.3 Aplicação da Metodologia

As ferramentas utilizadas, os testes propostos e o material desenvolvido foi de suma importância para a análise das relevâncias, além de possibilitar as comparações apresentadas em 4. Com as métricas, pode-se calibrar a rede não-supervisionada e avaliar a classe proposta, comparando além disto com os outros *clusters*. Em seguida, o processo comparativo, foi uma abordagem proposta que visa responder a seguinte pergunta: Será que as características relevantes para o agrupamento (etapa não-supervisionada - LARFDSSOM), possuem algo em comum com as características necessárias para a classificação (etapa supervisionada)? O estudo ainda está em estágio inicial, por isto foram feitas apenas análises manuais. Um exemplo, é a comparação de N valores de filtros próximos entre a etapa de visualzação (supervisionada) e a etapa do larfdssom (não supervisionada), e com isto dizer o que realmente é relevante para uma rede aprender de maneira não-supervisionada.

Tabela 2: Intervalos de mínimo e máximo - Busca de parâmetros no LARFDSSOM

Parâmetro	Mínimo	Máximo
Limite de ativação (a_t)	0.9500	0.9999
Taxa de aprendizagem do vencedor (e_b)	0.0010	0.0900
Taxa de aprendizagem dos vizinhos (e_n)	0.0010	0.1000
Taxa de aprendizagem da relevância (beta)	0.0010	0.4000
Número máximo de neurônios (n_max)	20	100
Máximo de competições (max_comp)	0.1000	0.9999
Número de épocas (t_max)	10	100
Porcentagem mínima de vitórias de um cluster (lp)	0.0020	0.0100
Suavidade do vetor de relevância (s)	0.0200	2.0000
Limite de conexão (c)	0.0100	0.3000

4 Resultados do Trabalho

Neste capítulo, serão apresentados os resultados da aplicação da metodologia descrita no capítulo anterior e será apresentada um estudo de caso das características relevantes para o grupo Zebras (Seção 4.3).

4.1 Busca de Parâmetros e CE

Para o algoritmo de agrupamento do LARFDSSOM ter resultados satisfatórios, foi necessário fazer uma busca de parâmetros. A busca de parâmetros consiste em um método que utiliza limites de mínimo e máximo dos parâmetros do LARFDSSOM para testar várias combinações de ajustes da rede. Então são utilizados nas análises posteriores, os parâmetros que obtiveram melhores resultados de CE. A Tabela 2, contém os valores utilizados na busca de parâmetros, estes valores foram definidos através de testes e dos trabalhos realizados por Bassani *et al.* [3] [33], e na Tabela 3 encontram-se os 10 melhores resultados encontrados. Os valores dos parâmetros utilizados no trabalho estão em negrito e para estes obteve-se um CE de 0.938, este valor de CE é considerado adequado por estar próximo de 1, como será visto na Seção 4.2.

4.2 Matriz de Confusão

A matriz de confusão (Tabela 4), como explicado na Seção 3.3.2, apresenta a taxa de acerto para cada grupo com relação ao rótulo da imagem. Já a Tabela 5, contém o mapeamento do rótulo e a categoria utilizada, os zeros presentes significam que houve pouca confusão entre as categorias, o que indica que os resultados obtidos são bons, confirmando o que foi antecipado

Tabela 3: *Clustering Error* e Parâmetros do LARFDSSOM - 10 melhores resultados encontrados

CE	a_t	e_b	e_n	beta	n_max	max_comp	t_max	lp	s	c
0.920	0.9795787	0.003	0.090	0.016	42	0.742	56	0.0058791	1.042	0.051
0.929	0.9768653	0.009	0.015	0.167	38	0.774	43	0.0075031	1.712	0.075
0.922	0.9853706	0.003	0.040	0.276	94	0.306	75	0.0068162	0.339	0.034
0.906	0.9847042	0.008	0.001	0.059	54	0.544	84	0.0049726	1.620	0.077
0.935	0.9854707	0.009	0.100	0.279	86	0.340	56	0.0051574	0.212	0.076
0.927	0.9804394	0.012	0.069	0.201	39	0.337	31	0.0031519	1.562	0.036
0.931	0.9846365	0.011	0.055	0.208	73	0.408	82	0.0082015	1.695	0.067
0.929	0.9840222	0.017	0.028	0.188	75	0.475	15	0.0066913	1.555	0.055
0.938	**0.9807178**	**0.012**	**0.066**	**0.135**	**41**	**0.360**	**67**	**0.0092746**	**1.673**	**0.037**
0.929	0.9933000	0.012	0.056	0.285	77	0.252	89	0.0065384	0.058	0.070

Tabela 4: Matriz de Confusão - LARFDSSOM (*Cluster* x Rótulo)

	Rótulo																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Cluster 0	0	0	1	0	0	0	0	0	0	**47**	0	1	0	0	0	0	0	0	0	0
Cluster 1	0	0	**47**	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0
Cluster 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**48**	0	0	0
Cluster 3	0	0	0	0	0	0	**48**	0	0	0	0	0	0	1	0	0	0	0	1	0
Cluster 4	0	0	0	0	**50**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cluster 5	0	0	2	0	0	0	1	0	0	0	0	0	0	1	0	0	2	0	**46**	0
Cluster 6	**50**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cluster 7	0	2	0	0	0	0	0	3	0	0	0	0	0	4	0	0	0	0	0	**17**
Cluster 8	0	0	0	0	0	0	0	0	0	1	0	0	**50**	4	0	0	0	0	0	0
Cluster 9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	**50**	0	0	0	0	0
Cluster 10	0	**46**	0	0	0	0	0	0	**42**	1	0	1	0	0	0	2	0	0	0	1
Cluster 11	0	0	0	**50**	0	0	0	3	6	0	0	0	0	3	0	0	0	0	1	2
Cluster 12	0	0	0	0	0	0	0	0	0	0	**50**	0	0	0	0	0	0	0	0	0
Cluster 13	0	1	0	0	0	**50**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cluster 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**30**
Cluster 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**48**	0	0	0	0
Cluster 16	0	1	0	0	0	0	0	**42**	0	0	0	1	0	0	0	0	0	0	0	0
Cluster 17	0	0	0	0	0	0	0	0	0	0	0	**45**	0	0	0	0	0	0	1	0
Cluster 18	0	0	0	0	0	0	0	2	1	1	0	1	0	**37**	0	0	0	0	0	0
Cluster 19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**50**	0	0

pelo CE. Ao analisar a Tabela 4, observando-se o *cluster* 6, é possível notar que a primeira coluna deste *cluster* tem o rótulo 0, teve um acerto de 50 imagens, então observando-se a Tabela 5 é possível ver que o rótulo 0 está associado a categoria zebra. De acordo com a Tabela 4, o *cluster* 6 teve um ótimo desempenho, acertando as 50 imagens de zebra.

Mesmo o resultado da matriz de confusão tendo sido bom, é possível observar que alguns resultados não foram como o esperado. Nos *clusters* 7 e 14, ve-se de acordo com a Tabela 4, que as imagens do rótulo 19 (*rotary phone*) foram divididas para os *clusters*: 7 (17 imagens) e 14 (30 imagens). Analisando as imagens da categoria *rotary phone*, notou-se que no *Dataset Easy*, há mais de um tipo de telefone (telefones antigos e telefones desmontados), dificultando a tarefa de agrupamento. Uma possível subdivisão para a categoria *rotary phone* é apresentada

Tabela 5: Mapeamento Categoria do *Dataset Easy* e Rótulo

Categoria	**Rótulo**
zebra	0
diamond ring	1
fern	2
pci card	3
faces easy 101	4
ketch 101	5
fireworks	6
roulette wheel	7
dice	8
mandolin	9
leopards 101	10
american flag	11
airplanes 101	12
fire extinguisher	13
motorbikes 101	14
french horn	15
tower pisa	16
killer whale	17
tombstone	18
rotary phone	19

na Figura 22.

O *cluster* 10, segundo a Tabela 4, representa imagens que deveriam pertencer a duas categorias distintas: rótulo 1 (*diamond ring*) com o valor de 46 e rótulo 8 (*dice*) com o valor de 42. Foi realizada a análise em cima das imagens das categorias: *diamond ring* e *dice*. Na Figura 23, se observa uma semelhança visual entre algumas imagens das duas categorias. Além disto, também notou-se a presença de imagens ruins para representar um dado ou um anel. As Figuras 24 e 25 ilustram, respectivamente, imagens das categorias *dice* e *diamond ring* que dificultam o agrupamento, até mesmo por seres humanos.

Os outros *clusters* tiveram um desempenho adequado e por isto não passaram por maiores análises. Para fins de estudo, foi realizado estudos com o *cluster* 6 (zebra), pois esta categoria tem imagens que foram classificadas bem pela etapa de visualização, e também foi agrupada bem pela etapa do LARFDSSOM (coluna 0 contém 50 da imagens de zebras para o *cluster* 6 e nenhuma dos demais *clusters*).

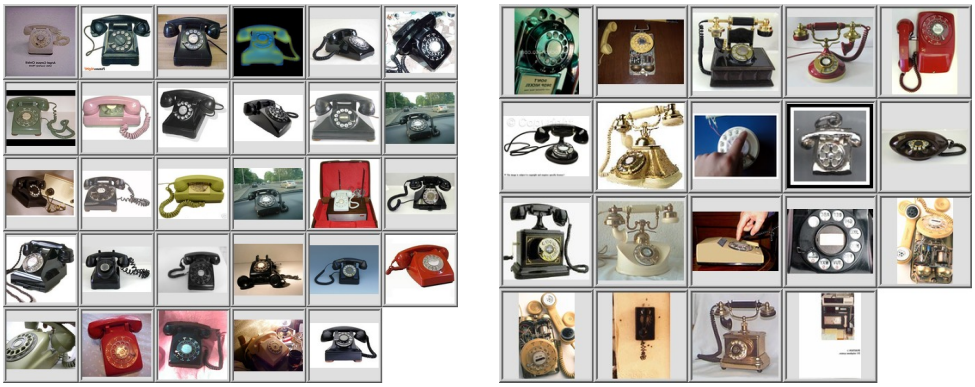

Figura 22: Subdivisão possível da categoria *rotary phone* Fonte: Autor

Figura 23: A imagem da esquerda contém a semelhança no formato cúbico do dado e do anel. A imagem do meio apresenta uma junção sobreposta das duas imagens anteriores. A imagem mais a direita apresenta outra semelhança no formato - Fonte: Autor

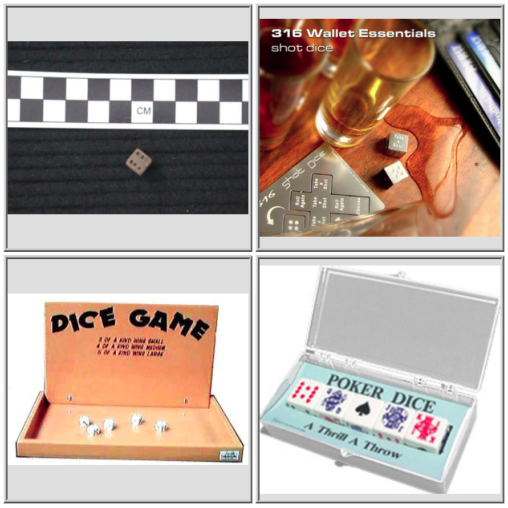

Figura 24: Imagens da categoria *dice* que são difíceis de agrupar, porque há outros objetos nas imagens - Fonte: Autor

Figura 25: Imagens da categoria *diamond ring* que são difíceis de agrupar, porque há outros objetos na imagem, ou a iluminação da imagem está ruim - Fonte: Autor

4.3 Estudo de Caso sobre as Relevâncias Aprendidas pelo LARFDS-SOM

Inicialmente, foram feitos testes com a ferramenta modificada *Deep Visualization Tool*. A Figura 26, ilustra o neurônio 54 (verde) selecionado na camada *inception*_3a da GoogLeNet, a escolha de tal neurônio, se deu por apresentar alta ativação na ferramenta *Deep Visualization Toolbox*. Vale ressaltar que a imagem gerada pela deconvolução, em amarelo, é uma aproximação de como o neurônio enxerga a entrada.

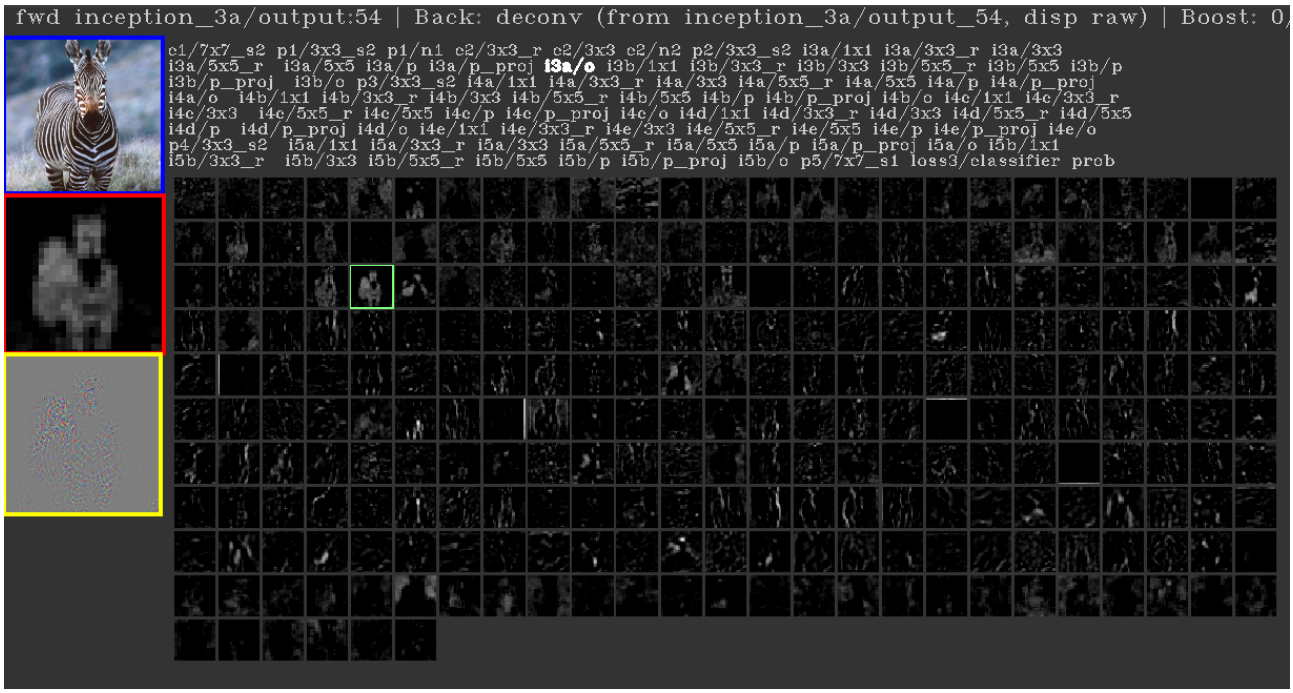

Figura 26: Imagens de treinamento na ferramenta *Deep Visualization Tool*: o neurônio 54 (verde) selecionado na camada *inception*3a, a imagem de entrada (azul), os valores da resposta do neurônio a entrada (vermelho) e a imagem gerada pela deconvolução (amarelo) - Fonte: Autor

Na Figura 27, são ilustradas as respostas de algumas entradas de categorias diferentes para o

neurônio 813 da camada pool5/7x7_s1 da GoogLeNet, vê-se que os maiores valores de resposta dos pesos (escala em tons de cinza - variando do preto ao branco), em amarelo, são para a categoria zebra. As imagens de deconvolução, em vermelho, foram importantes para saber o que realmente os neurônios estavam aprendendo, e qual era a sua resposta para outros tipos de categorias. Verificando as maiores ativações dadas pela ferramenta (Figura 28), também nota-se que foram para a categoria zebra, ou seja, este neurônio é um bom detector de zebras e será usado (assim como os neurônios: 259, 519, 547, 736, 413, 813 e 764), para análise das relevâncias.

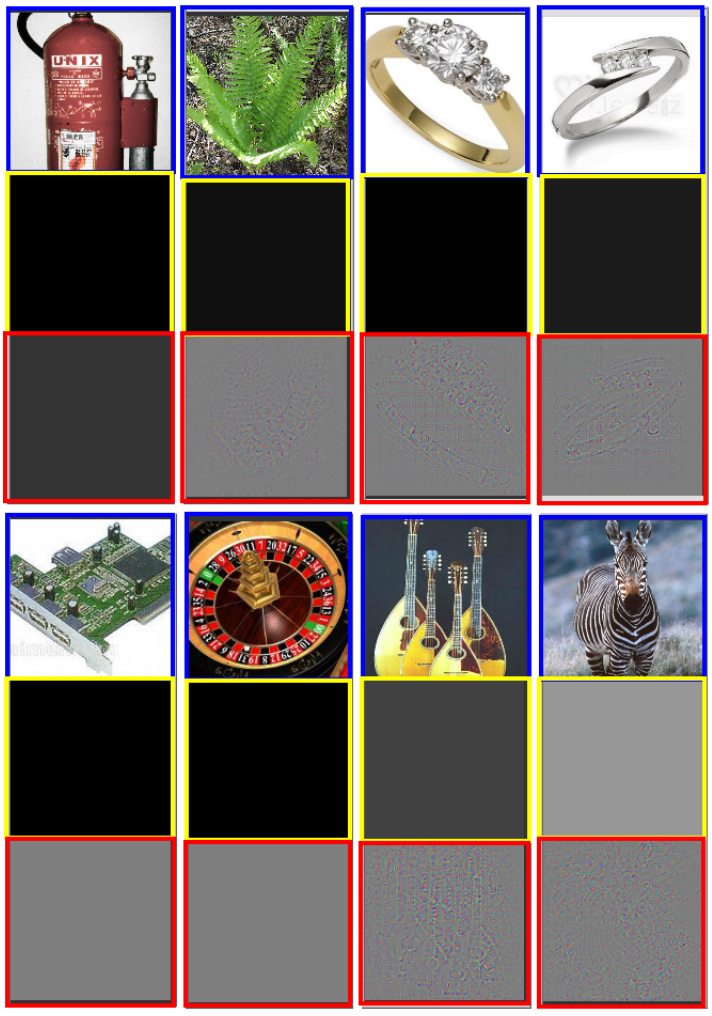

Figura 27: Imagens de algumas categorias para o neurônio 813 camada pool5/7x7_s1. A entrada, em azul, o valor dos pesos (tons de cinza variando de 0 até 1), em amarelo, e as imagens geradas pela deconvolução, em vermelho - Fonte: Autor

Os últimos testes do trabalho, consistiram em analisar os valores reais das relevâncias dadas pela etapa do LARFDSSOM e comparar com os mesmos neurônios da etapa visual. A seguir, são apresentados os passos realizados nesta etapa de análise de relevâncias:

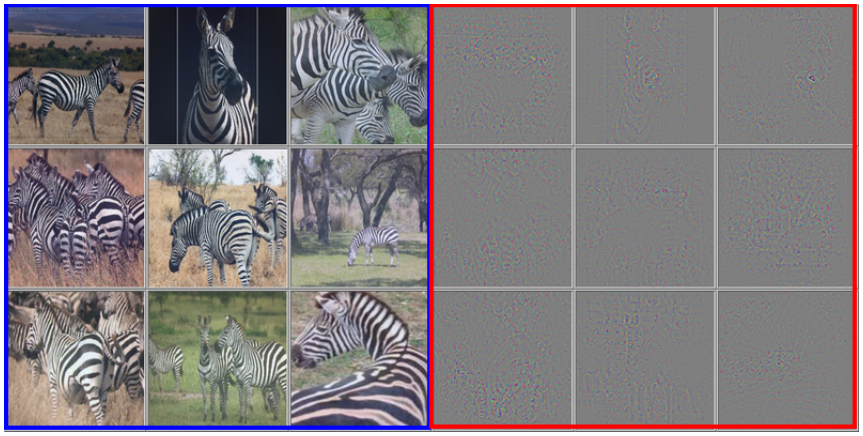

Figura 28: Maiores ativações neurônio 813 camada pool5/7x7_s1, em azul, e as imagens geradas pela deconvolução, em vermelho - Fonte: Autor

1. Partindo da etapa visual (ferramenta de visualização), foi feita uma busca pelas maiores ativações dos neurônios para cada classe, por exemplo: 9 maiores ativações do neurônio 1, se o neurônio tiver as 9 ativações sendo zebra, então ele é considerado relevante para classificar zebras. Logo, ele é escolhido.

2. Dado que esse neurônio 1, escolhido na etapa anterior, é bom para zebras na etapa visual, então procura-se os valores reais das relevâncias para cada *cluster* obtidos pelo LARFDSSOM.

3. Fazer a análise dos valores das relevâncias entre os *clusters* para aquele neurônio, e com isso, concluir se há alguma relação entre as ativações apresentadas pela parte supervisionada e as relevâncias identificadas pela parte não-supervisionada.

O resultado para alguns neurônios escolhidos com as maiores ativações para zebras pode ser visto na Tabela 6).

Seguindo o procedimento citado, foi possível notar um padrão. Os neurônios com as maiores ativações para categoria zebra (parte supervisionada), tinham os menores valores de relevâncias entre os *clusters* (parte não-supervisionada). Foi possível confirmar este padrão realizando o mesmo tipo de análise para as categorias *French Horn* e *Mandolin*. Ou seja, as categorias que tinham maiores ativações dado um neurônio na parte supervisionada, leva aos menores valores de relevância entre as categorias na parte não-supervisionada, fazendo a ressalva, que quanto maior a diferença desse menor valor para os outros, na parte não supervisionada, maior a probabilidade de ter mais de uma ativação alta desta categoria no neurônio na parte supervisionada. Na Tabela 7, pode-se ver o resultado do mesmo procedimento para a categoria *French Horn*.

Tabela 6: Relevâncias do LARFDSSOM por *cluster* x Neurônios de maiores ativações para categoria zebra

	Neurônio						
Cluster	813	259	519	547	736	413	764
0	0.511197	0.536897	0.510632	0.504664	0.510568	0.526994	0.500659
1	0.492763	0.496963	0.51158	0.508797	0.488406	0.497441	0.496771
2	0.51718	0.529952	0.510537	0.494611	0.508241	0.522851	0.502526
3	0.515297	0.51858	0.510267	0.506117	0.495945	0.518512	0.504935
4	0.505207	0.511265	0.512283	0.50933	0.506036	0.51226	0.499251
5	0.520374	0.529894	0.510384	0.511534	0.475866	0.522296	0.494864
6	**0.415667**	**0.45507**	**0.421948**	**0.435603**	**0.435036**	**0.416306**	**0.387213**
7	0.51826	0.528975	0.513344	0.508193	0.511343	0.519686	0.499949
8	0.499101	0.522294	0.517928	0.511353	0.506241	0.521009	0.469353
9	0.503647	0.517986	0.507566	0.50122	0.491827	0.513819	0.49578
10	0.524197	0.543929	0.516805	0.517539	0.519686	0.52982	0.482107
11	0.522653	0.535682	0.522201	0.517585	0.510583	0.528131	0.457821
12	0.510341	0.523209	0.514702	0.513385	0.474706	0.516482	0.504283
13	0.514421	0.523729	0.512481	0.511156	0.507914	0.520347	0.501082
14	0.514615	0.519459	0.513515	0.506193	0.512056	0.508606	0.510413
15	0.517629	0.52792	0.518283	0.509864	0.512066	0.522462	0.508725
16	0.514377	0.530114	0.517091	0.511054	0.50671	0.505971	0.492407
17	0.50343	0.527805	0.51515	0.516653	0.501364	0.515983	0.496173
18	0.514884	0.523802	0.516926	0.506462	0.507903	0.521165	0.501317
19	0.50475	0.52419	0.501236	0.515076	0.507828	0.513164	0.482798

Tabela 7: Relevâncias do LARFDSSOM por *cluster* x Neurônios de maiores ativações para categoria *french horn*

	Neurônio				
Cluster	627	672	767	202	47
0	0.516837	0.46961	0.509909	0.507007	0.474653
1	0.515001	0.463252	0.506264	0.512702	0.507896
2	0.514268	0.474899	0.518691	0.516257	0.495145
3	0.518526	0.474371	0.503722	0.51602	0.508692
4	0.505954	0.482493	0.507878	0.497981	0.503363
5	0.51996	0.446953	0.525404	0.521255	0.501212
6	0.518202	0.424483	0.509284	0.515805	0.493306
7	0.50749	0.471078	0.522099	0.508847	0.51159
8	0.521886	0.473686	0.521538	0.503585	0.513354
9	0.511996	0.458113	0.512005	0.506271	0.504632
10	0.504548	0.453681	0.520216	0.516472	0.501977
11	0.514658	0.480443	0.506486	0.508309	0.509821
12	0.515444	0.444845	0.515609	0.513456	0.490864
13	0.509476	0.479657	0.514724	0.509684	0.499031
14	0.510002	0.495769	0.514466	0.508159	0.510807
15	**0.427935**	**0.395013**	**0.437613**	**0.428668**	**0.394917**
16	0.515147	0.473925	0.516552	0.516316	0.505794
17	0.508232	0.473043	0.501764	0.511452	0.473707
18	0.492586	0.474331	0.517733	0.483041	0.508292
19	0.520036	0.46847	0.519917	0.516688	0.492559

Vale salientar que os menores valores devem ter uma diferença razoável para as outras categorias, e quanto maior esta diferença, maior será a quantidade de maiores ativações na parte supervisionada. Com isto, este procedimento se mostrou útil como uma fase preliminar para descobrir quais neurônios respondem melhor para certas categorias, utilizando apenas a parte não-supervisionada, e em seguida, para se ter uma noção visual, são obtidas imagens de deconvolução relativas ao neurônio identificado.

Por exemplo, escolhendo-se aleatoriamente um neurônio, como o 355, o menor valor é 0.468733 (*cluster* 15), e o segundo menor valor é 0.50603 (os outros valores variam até 0.52), os valores são subjetivamente considerados próximos, então não se pode concluir que este *cluster* vai ter as maiores ativações para a categoria testada. Ao observar suas ativações, nota-se que ele apresenta apenas uma ativação, e a maior categoria ativada nele possui 2 ativações. Por outro lado, um neurônio como o 838, que a menor relevância é 0.416585 (*cluster* 6), a segunda menor tem 0.451527 e os outros valores variam até 0.51, tem-se varias variações, o que leva a crer que ele se ativa diversas vezes, o que é confirmado ao observar que há um total de no mínimo 9 ativações máximas para este *cluster* (9 é o limite da ferramenta de visualização).

5 Conclusão

Neste trabalho são apresentados conceitos importantes de Redes Neurais Convolucionais e Mapas Auto-Organizáveis, além da análise das relevâncias da saída de uma Rede Neural Profunda. Os resultados apresentados no capítulo anterior mostraram que existe uma junção do que é relevante para aprendizagem supervisionada e o que é relevante para aprendizagem não-supervisionada. Mesmo não conseguindo responder quais partes, de um objeto de certa categoria, são relevantes aprender para agrupa-los, o trabalho se mostrou satisfatório, pois apresenta um ponto inicial para tal investigação. A visualização de características das camadas intermediárias é uma técnica nova e poderosa, e com ela, pode-se chegar a um melhor entendimento do funcionamento das redes convolucionais profundas. Como apresentou-se no capítulo anterior, as análises das relevâncias conseguem dizer quais neurônios serão importantes, antes de se realizar o processo de aprendizagem supervisionada completo, pois estes terão alta probabilidade de serem ativados para categoria em teste, na etapa supervisionada.

As adaptações realzadas na *Deep Visualization Toolbox* serão disponibilizadas no GitHub, junto aos códigos implementados e testes realizados para possíveis contribuições de outros pesquisadores. Acredita-se que exista uma conexão entre as etapas supervisionadas e não-supervisionadas, então trabalhos futuros podem realizar testes estatísticos nas relevâncias fornecidas pelo LARFDSSOM e propor outras técnicas para tal análise. Outro possível trabalho, é utilizar o LARFDSSOM com características de camadas iniciais e fazer uma análise se com as relevâncias que o LARDFSSOM disponibiliza para tais camadas, pode-se fazer um mapeamento dos pesos para possíveis partes de um objeto. Com isto, seria possível descobrir bons valores de pesos para agrupar possíveis partes do objeto, e com isto, ter processos não-supervisionados que agrupam bem partes dos objetos, que ao juntá-los, consegue-se agrupar os objetos completos (por exemplo: se utilizar pesos que são bons em agrupar olhos, é bem provável que com este peso, o *cluster*, seja formado por objetos que tenham olhos). Também é bom lembrar, que outra abordagem possível consiste em treinar a rede com o conjunto de dados que será utilizado para agrupamento, e com isto, realizar uma comparação mais fiel com a parte supervisionada.

Por fim, a área é muito nova e os testes ainda estão em fase inicial. O trabalho além de material para pesquisas futuras, marca-se como uma possível abordagem para entender através de aprendizagem não-supervionada e ferramentas supervisionadas, as características relevantes para uma rede.

Referências

[1] Tinne Tuytelaars, Christoph H Lampert, Matthew B Blaschko, and Wray Buntine. Unsupervised object discovery: A comparison. *International journal of computer vision*, 88(2):284–302, 2010.

[2] Antônio de Pádua Braga, André Carlos Ponce de Leon Ferreira, and Teresa Bernarda Ludermir. *Redes neurais artificiais: teoria e aplicações*. LTC Editora, 2007.

[3] Hansenclever F Bassani and Aluizio FR Araújo. Dimension selective self-organizing maps for clustering high dimensional data. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[4] Hubel&wiesel. `http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html`. Accessed: 2017-07-02.

[5] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*, 2013.

[6] Michael A Nielsen. Neural networks and deep learning, 2015.

[7] Andrej Karpathy. Cs231n: Convolutional neural networks for visual recognition. *Neural networks*, 1, 2016.

[8] A Deshpande. A beginner's guide to understanding convolutional neural networks. *Retrieved March*, 31:2017, 2016.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[10] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[11] Andres Munoz. Machine learning and optimization. *URL: https://www. cims. nyu. edu/~ munoz/files/ml_ optimization. pdf [accessed 2016-03-02][WebCite Cache ID 6fiLfZvnG]*, 2014.

[12] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[13] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[14] WS McCulloch. Pitts, wh (1943). a logical calculus of ideas im-manent in nervous activity. *Bull. math. Biophys*, 5:115–33, 86.

[15] Donald O Hebb. The organization of behavior, 1949.

[16] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[17] Solange Oliveira Rezende. *Sistemas inteligentes: fundamentos e aplicações*. Editora Manole Ltda, 2003.

[18] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[19] SIMON HAYKIN. Redes neurais princípios e aplicações. *Segunda Edição. Porto Alegre*, 2001.

[20] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[21] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[22] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[23] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[24] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[25] Momotaz Begum and Fakhri Karray. Visual attention for robotic cognition: a survey. *IEEE Transactions on Autonomous Mental Development*, 3(1):92–105, 2011.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[30] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[31] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[33] Hansenclever F Bassani and Aluizio FR Araujo. Dimension selective self-organizing maps with time-varying structure for subspace and projected clustering. *IEEE transactions on neural networks and learning systems*, 26(3):458–471, 2015.

[34] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[35] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[36] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[37] Anne Patrikainen and Marina Meila. Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):902–916, 2006.

[38] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *Proceedings of the VLDB Endowment*, 2(1):1270–1281, 2009.

[39] Kai Ming Ting. *Confusion Matrix*, pages 209–209. Springer US, Boston, MA, 2010.