UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

LUCAS DE LIMA NOGUEIRA

***LEARNING TO TRANSFER WHAT, WHERE AND WHICH*: Método de *transfer learning* entre redes convolucionais de arquiteturas diferentes**

Recife

2022

LUCAS DE LIMA NOGUEIRA

LEARNING TO TRANSFER WHAT, WHERE AND WHICH: Método de *transfer learning* entre redes convolucionais de arquiteturas diferentes

Trabalho apresentado ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciências da Computação. Área de concentração: Inteligência Computacional.

Orientador: Fernando Maciano de Paula Neto

Co-orientador: Cleber Zanchettin

Recife

2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

N778l Nogueira, Lucas de Lima
Learning to transfer what, where and which: método de transfer *learning* entre redes convolucionais de arquiteturas diferentes / Lucas de Lima Nogueira – 2022.
63 f.: il., fig., tab.

Orientador: Fernando Maciano de Paula Neto.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências.

1. Inteligência computacional. 2. *Transfer learning*. 3. *Deep learning*. 4. Aprendizagem de máquina. 5. Redes neurais convolucionais. I. Paula Neto, Fernando Maciano de (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2023 – 115

LUCAS DE LIMA NOGUEIRA

***LEARNING TO TRANSFER WHAT, WHERE AND WHICH*: Método de *transfer learning* entre redes convolucionais de arquiteturas diferentes**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciências da Computação. Área de concentração: Inteligência Computacional.

Data de aprovação: 12 / Setembro / 2022

Orientador: Prof. Dr. Fernando Maciano de Paula Neto

UFPE – Centro de Informática

BANCA EXAMINADORA

Prof. Dr. Cleber Zanchettin (Co-orientador)

UFPE – Centro de Informática

Prof. Dr. Adriano Lorena Inácio de Oliveira

UFPE – Centro de Informática

Prof. Dr. Bruno José Torres Fernandes

UPE – Departamento de Engenharia da Computação

AGRADECIMENTOS

A Deus, que sempre esteve me guiando em decisões, caminhos e realizações durante minha vida.

Aos meus pais, Rizoneide e Zanoni, que sempre me apoiaram e auxiliaram meus estudos e escolhas. Palavras não são suficientes para descrever o quanto sou grato por todo suporte, além dos ensinamentos e amor que recebi.

Aos meus avôs, avós, tios e tias, que também me guiaram nessa jornada, em especial à tia Rosineide (Neném), que me acolheu e sempre esteve presente para me auxiliar.

A todos meus amigos e amigas, que me apoiaram e sempre me incentivaram a continuar.

RESUMO

Atualmente, os modelos de aprendizagem profunda estão sendo utilizados para solucionar uma grande variedade de problemas. No entanto, esse tipo de algoritmo usualmente necessita de grandes quantidades de dados para alcançar bons desempenhos. Apesar do crescimento da quantidade de dados disponíveis devido à digitalização da informação, essa ainda não é uma realidade para diversos problemas, além da necessidade de um alto custo computacional, dependendo da complexidade envolvida. Nesse sentido, técnicas de transferência de aprendizagem vêm sendo desenvolvidas para superar essa barreira. Algumas técnicas propostas recentemente envolvem utilizar os mapas de ativação a nível de camadas e/ou canais de uma rede pré-treinada, de forma a guiar o treino de uma nova rede. Neste trabalho, é proposto estender essa ideia, incluindo a utilização de informação dos mapas de ativação a nível de *pixels*, além de canais *e* camadas, de forma a refinar a transferência de conhecimento, aumentando o desempenho do processo. Nesse sentido, foram realizados testes utilizando os conjuntos de dados CIFAR-10, CIFAR-100 e STL-10, e o método proposto se demonstrou superior a outros métodos da literatura, como o *L2T-ww* e *UFM*, em um cenário essencial de quantidade limitada de dados de treinamento. A comparação entre os métodos foi analisada utilizando testes de Mann-Whitney, alcançando-se aumento no método proposto de até 3,75% na acurácia, em um dos cenários, com diferença estatística significativa (p-*value* < 0,05).

Palavras-chave: *transfer learning*; *deep learning*; aprendizagem de máquina; redes neurais convolucionais.

ABSTRACT

The deep learning models are being used to solve a wide variety of problems today. However, this type of algorithm usually needs large amounts of data to reach good performance. Despite the growth of availability of data due to digitalization, this is still not a reality for various problems, beyond the need of high computational cost depending of complexity involved. Due to that, transfer learning techniques have been developed to overcome this barrier. Some recently proposed techniques involve using activation maps at the level of layers and/or channels of a pre-trained network, in order to guide the training of a new network. In this work, we propose to extend this idea, including the use of pixel level information from activation maps, in addition to layers and channels, in order to refine knowledge transfer, improving the performance of process. In this regard, we performed tests in CIFAR-10, CIFAR-100 and STL-10 and the proposed method proved to be superior to other methods from literature, in an essential scenario of limited training data. The comparison between both methods was performed based on Mann-Whitney tests, showing an accuracy increase of up to 3.75% in one of the scenarios, with significative statistical difference (p-value < 0.05).

Keywords: transfer learning; deep learning; machine learning; convolutional neural netoworks.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com os recentes avanços na área da tecnologia, a capacidade de processamento dos computadores se amplifica a cada ano. No contexto da área de aprendizagem de máquina, esse aumento está permitindo o processamento de grandes quantidades de dados e a utilização de modelos com bilhões de parâmetros. No entanto, a limitação de recursos computacionais ainda é um problema a ser enfrentado. Problemas de alta complexidade requerem elevados custos computacionais e tempo de processamento, o que pode dificultar, ou até inviabilizar, diversas soluções. Nesse sentido, existem diversas técnicas propostas para diminuir o tempo necessário para a solução de problemas do âmbito da aprendizagem de máquina, como *a* transferência de aprendizagem*.*

No contexto das redes neurais, a transferência de aprendizagem é utilizada para reduzir o tempo de treinamento dos modelos. Uma das técnicas mais utilizadas para isso é o pré-treinamento com *finetuning*, proposta por Sharif Razavian *et al.* (2014), que consiste no aproveitamento dos pesos de uma rede pré-treinada, de forma a reduzir o tempo de aprendizado para uma nova tarefa. Apesar de bastante promissora, uma das limitações dessa abordagem é que a transferência de aprendizagem é realizada entre modelos de redes neurais iguais, limitando as possibilidades de treino. Nesse sentido, a proposição de novas técnicas de transferência de aprendizagem é uma necessidade da área de aprendizagem de máquina, de forma a permitir maiores avanços no treinamento de modelos de redes neurais.

Uma técnica bastante promissora de transferência de aprendizado entre redes neurais de arquiteturas diferentes é a *Learning to Transfer What and Where* (*L2T-ww*), proposta por Jang *et al*. (2019). Essa abordagem consiste na utilização de meta-redes, que conectam duas redes neurais convolucionais, de forma a realizar a transferência de aprendizado entre ambas. Para isso, essas meta-redes conectam as camadas (*layers*) e os canais (*channels)* das redes neurais, aprendendo a enfatizar as regiões mais apropriadas para transferir o aprendizado. Uma das vantagens desse tipo de abordagem, é permitir que a transferência de aprendizagem possa ser realizada entre redes convolucionais de arquiteturas diferentes. Apesar de bastante promissora, essa técnica ainda

pode ser aprimorada, de forma a alcançar resultados superiores aos conhecidos na literatura.

1.1 OBJETIVOS

Os objetivos deste trabalho foram divididos em Objetivo geral, onde foram abordadas as ideias que centralizam a proposta principal, e Objetivo específico, detalhando-se os resultados a serem obtidos.

1.1.1 Objetivo geral

O principal objetivo deste trabalho consiste em um aprimoramento da técnica *L2T-ww*, de forma a obter resultados superiores à abordagem existente. Para isso, é proposta a adição de uma nova meta-rede, que conecta os mapas de ativação a nível de *pixels*. Nesse sentido, será possível aperfeiçoar a transferência de aprendizado, melhorando os resultados dos modelos.

1.1.2 Objetivos específicos

O trabalho possui os seguintes objetivos específicos:

a) Apresentar um aprimoramento da técnica *L2T-ww*;
b) Testar a abordagem proposta em diferentes conjuntos de dados;
c) Comparar os resultados com a abordagem anterior;
d) Aprofundar a análise dos resultados, de forma a entender as vantagens do método proposto

1.2 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado da seguinte maneira:

- **Capítulo 2 – Fundamentação teórica**: Conceitos relacionados a redes convolucionais e transferência de aprendizado, necessários para o entendimento do método proposto.

- **Capítulo 3 – Outros trabalhos relacionados**: Métodos recentes de transferência de aprendizagem, explicando-se os trabalhos realizados e os pontos fracos de cada técnica.
- **Capítulo 4 – Método proposto**: O método *Learning to Transfer What, Where and Which* (*L2T-www*), proposto neste trabalho, é descrito com maiores detalhes.
- **Capítulo 5 – Experimentos e discussões**: São apresentadas as configurações dos testes realizados, detalhando-se os conjuntos de dados, modelos e configurações de treinamento. Além disso, também são descritos os resultados obtidos nos testes realizados, comparando-se o método proposto com a abordagem anterior através de testes estatísticos.
- **Capítulo 6 – Conclusão e trabalhos futuros**: Conclusões gerais e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresenta-se um resumo das teorias necessárias para a elaboração do trabalho proposto. Nesse sentido, são introduzidos conceitos como de aprendizagem de máquina, redes neurais e transferência de aprendizagem, de forma a facilitar o entendimento do método proposto.

2.1 APRENDIZAGEM DE MÁQUINA

Segundo McCarthy (1997), pesquisador que cunhou o termo, a Inteligência Artificial (IA) é uma ciência que tem como objetivo criar máquinas inteligentes, capazes de reproduzir tarefas como humanos. Nesse sentido, a aprendizagem de máquina é um campo, dentro do domínio da IA, que estuda a habilidade de computadores aprenderem tarefas sem serem explicitamente programados. Para isso, os algoritmos de aprendizagem de máquina são desenvolvidos para aprender a identificar padrões através de informações passadas. Nesse contexto, segundo Mitchell (1997), um programa aprende de uma experiência E, com respeito a uma tarefa T e uma métrica de desempenho P, quanto o desempenho na tarefa T, medido por P, melhora com a experiência E. Para isso, diversos algoritmos vêm sendo desenvolvidos, alcançando grande sucesso na resolução de tarefas complexas, como por exemplo as redes neurais artificiais.

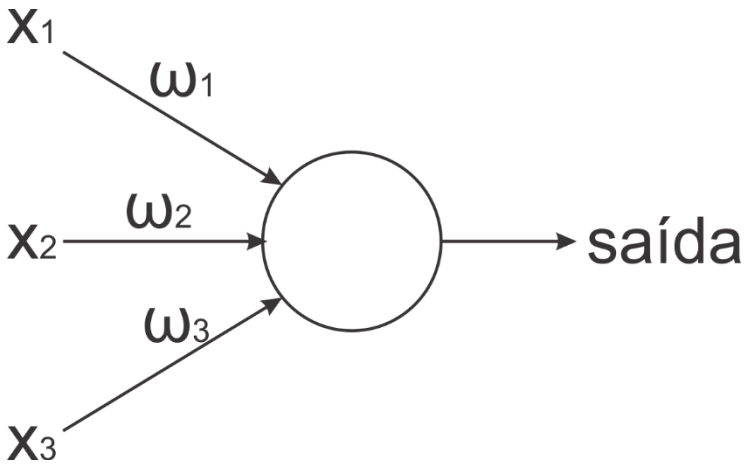
2.2 REDES NEURAIS ARTIFICIAIS

Inspiradas na organização de neurônios de organismos inteligentes, as redes neurais artificiais são técnicas computacionais capazes de realizar o reconhecimento de padrões a partir da aquisição de experiência através de experiência. Segundo Haykin (2009), essas redes são formadas por um conjunto de neurônios artificiais conectados, que se ativam seguindo regras específicas, dependendo da necessidade de cada modelo.

Um dos primeiros modelos de neurônios artificiais é chamado de *Perceptron* (ROSENBLATT, 1958), inspirado nos trabalhos desenvolvidos por McCulloch e Pitts (1943). De forma simplificada, os *perceptrons* recebem

entradas binárias $x_1, x_2, ..., x_n$ produzindo uma saída também binária. Cada entrada é ponderada por pesos $\omega_1, \omega_2, ..., \omega_n$, representando a importância de cada entrada na ativação da saída, Figura 1. Assim, a saída do neurônio recebe os valores 0 ou 1, de acordo com a soma ponderada das entradas ultrapassar ou não um valor limite pré-definido, como descrito algebricamente pela Equação 1.

Figura 1 - Ilustração do *perceptron*

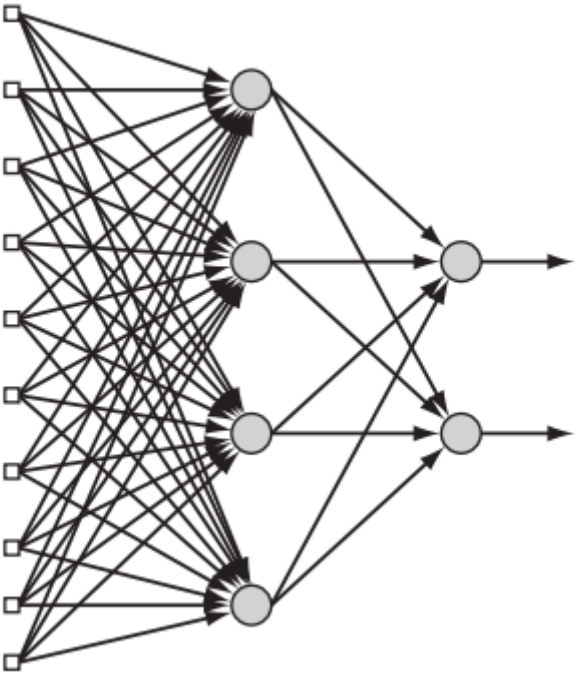


Fonte: o autor (2022)

$$saída = \begin{cases} 0 \; se \; \sum_j \omega_j x_j \leq limiar \\ 1 \; se \; \sum_j \omega_j x_j > limiar \end{cases} \quad (1)$$

Cada neurônio é responsável por realizar decisões a partir das entradas que lhe são ativadas. Nesse sentido, os neurônios são combinados, de forma a combinar decisões para a realização de tarefas mais complexas. Assim, são formadas as redes neurais artificiais, que basicamente são um conjunto de neurônios artificiais, conectados de acordo com alguma arquitetura específica de cada modelo de rede neural, como ilustrado na Figura 2 um modelo *fully-connected*.

Figura 2 - Exemplo de rede neural *fully connected*



Fonte: Haykin (2009)

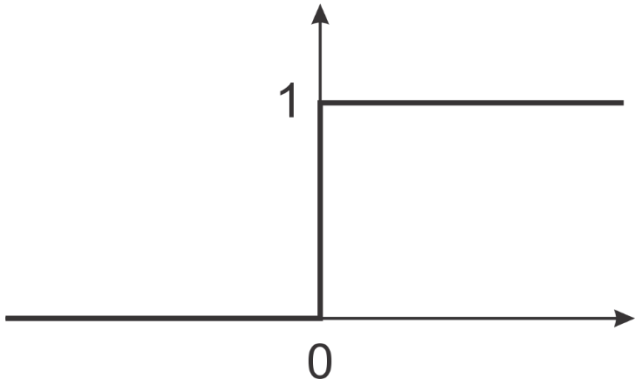
2.2.1 Funções de ativação

Apesar de ter sido a base das redes neurais mais modernas, o *perceptron* apresenta algumas limitações. Uma delas é a extrema sensibilidade da saída, de acordo com os valores de entrada. Por exemplo, devido ao caráter binário apresentado pelos *perceptrons*, uma pequena modificação em alguma das entradas x_i pode ocasionar uma modificação abrupta da saída z (de 0 para 1). Esse efeito pode limitar a eficácia das redes neurais artificiais formadas por *perceptrons* no aprendizado e realização de tarefas mais complexas. De forma a solucionar esse problema, surgiu a implementação das chamadas funções de ativação, de forma a aperfeiçoar os neurônios artificiais.

No intuito de facilitar o entendimento, é possível fazer $b = -limiar$, onde b é conhecido como *bias*. Além disso, utilizando $z = \sum_j \omega_j x_j + b$, é possível reescrever a Equação (1) como $saída = \sigma(z)$, onde σ é a função de ativação, que para o caso dos *perceptrons*, é definida pela Equação 2, ilustrada na Figura 3.

$$saída = \sigma(z) = \begin{cases} 0 \; se \; z \leq 0 \\ 1 \; se \; z > 0 \end{cases} \quad (2)$$

Figura 3 - Função de ativação *step*

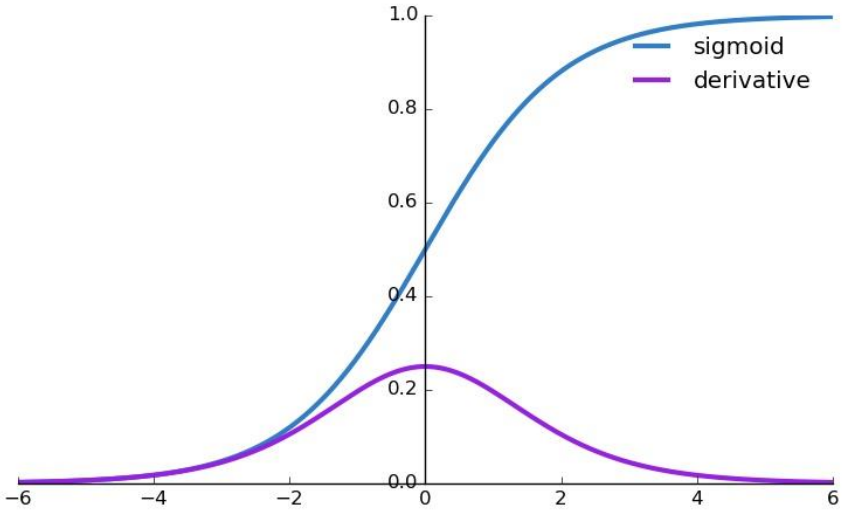


Fonte: o autor (2022)

2.2.1.1 Sigmoid

A função de ativação *sigmoid* (HAYKIN, 2009), Figura 4, é bastante utilizada no âmbito das redes neurais. Devido ao formato suave, essa função de ativação soluciona o problema de sensibilidade da saída dos *perceptrons*, melhorando o desempenho das redes.

Figura 4 - Função de ativação *sigmoid* e derivada



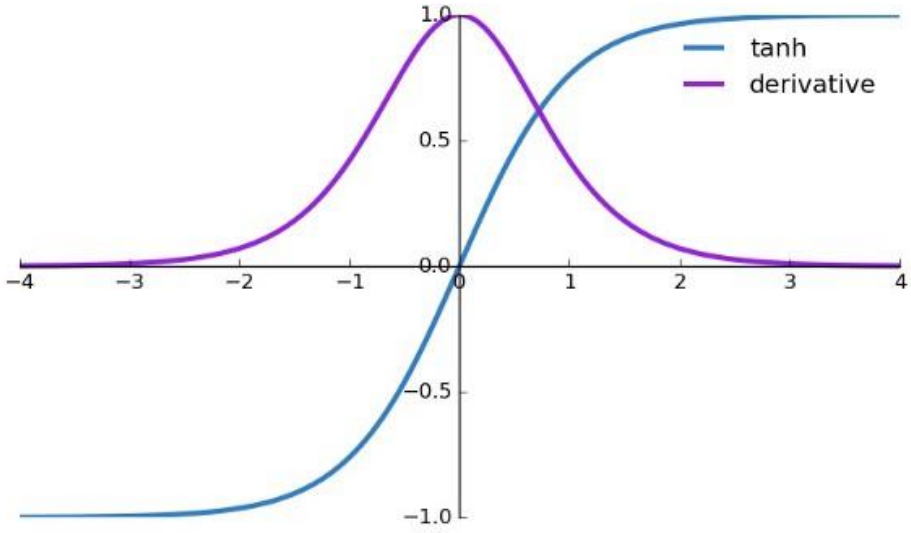
Fonte: Retirado da internet. Disponível em http://ronny.rest/blog/post_2017_08_10_sigmoid/. Acesso em 30 de agosto de 2022.

Apesar de bastante utilizada, esse tipo de função de ativação também apresenta adversidades, como a dissipação do gradiente (BENGIO *et al.*, 1994), que normalmente ocorre quando a função *sigmoid* é utilizada em redes com um número maior de camadas. Esse problema ocorre devido ao algoritmo de *backpropagation* (RUMELHART, 1986), utilizado no aprendizado das redes neurais, se basear na primeira derivada da função de ativação em cada camada para atualização dos pesos da rede neural. Nesse sentido, como é possível observar na Figura 4, a derivada da função *sigmoid* possui valores abaixo de 1 em todo domínio. Dessa forma, os valores de atualização dos pesos são multiplicados por valores abaixo de 1 em cada camada da rede, tornando-se ínfimos nas camadas mais iniciais. Assim, as atualizações dos pesos nas camadas iniciais se tornam insignificantes, nos casos de redes mais profundas com utilização da função de ativação *sigmoid*, limitando o aprendizado desses modelos. Outro problema apresentado por esse tipo de função de ativação é a saturação. Esse problema ocorre quando um neurônio recebe um valor de entrada elevado, resultando em um gradiente com valor insignificante, atenuando o aprendizado desse neurônio.

2.2.1.2 Tangente Hiperbólica

A tangente hiperbólica (HAYKIN, 2009) tem formato similar à *sigmoid*, porém deslocada, podendo assumir valores negativos em um intervalo [-1, 1], Figura 5. Uma das vantagens dessa função em relação à *sigmoid* é a robustez ao problema da dissipação do gradiente. Essa robustez se deve à derivada da tangente hiperbólica, Figura 5, apresentar valores maiores em relação à derivada da função *sigmoid*, com um intervalo [0, 1] contra [0, 0.25] da *sigmoid*.

Figura 5 - Função de ativação tangente hiperbólica e derivada



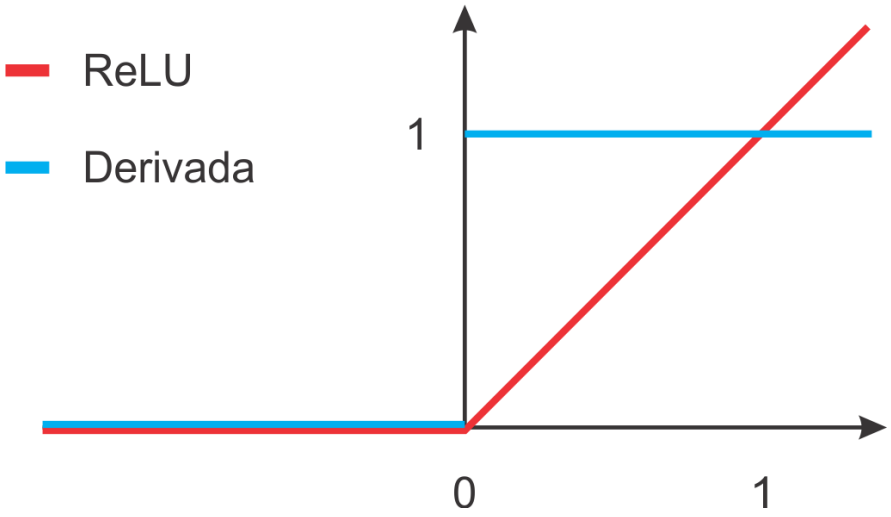
Fonte: Retirado da internet. Disponível em https://medium.com/@omkar.nallagoni/activation-functions-with-derivative-and-python-code-sigmoid-vs-tanh-vs-relu-44d23915c1f4. Acesso em 30 de agosto de 2022.

No entanto, como é possível observar, a derivada da tangente hiperbólica apresenta valores menores que 1 em toda sua extensão, podendo ainda assim ocorrer o problema de dissipação do gradiente. Além disso, o problema de saturação de neurônios ainda persiste nesse tipo de função.

2.2.1.3 ReLU

A função de ativação *ReLU* (GLOROT; BORDES; BENGIO, 2011), Figura 6, é bastante difundida na área de *deep learning*, sobretudo em redes neurais convolucionais. Essa popularidade se deve à robustez em relação ao problema da dissipação ao gradiente, que é uma preocupação bastante considerada em modelos com muitas camadas, como é o caso das redes convolucionais. Além disso, esse tipo de função também apresenta robustez em relação ao problema da dissipação do gradiente, permitindo um aprendizado mais acelerado.

Figura 6 - Função de ativação *ReLU* e derivada



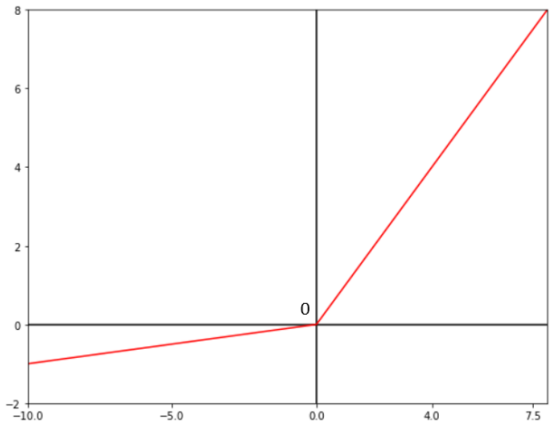
Fonte: o autor (2022)

A função de ativação *ReLU* pode apresentar outros tipos de problemas. Um deles é a explosão do gradiente, em razão das saídas dos neurônios com esse tipo de função de ativação poderem assumir valores elevados. Dessa forma, pode-se acarretar em instabilidade do aprendizado da rede, ou até a ocorrência de valores fora da faixa de representação computacional, impedindo a atualização dos parâmetros da rede. Outro problema é o chamado *dying ReLU* (LU *et al.*, 2019), em razão da característica desse tipo de função apresentar o valor zero para entradas negativas, que pode ocorrer quando um neurônio aprende um valor de *bias* muito negativo. Quando isso ocorre, é difícil para o neurônio sair desse estado, pois a derivada da função *ReLU* nessa faixa negativa também é zero, anulando a atualização do peso desse neurônio.

2.2.1.4 Leaky ReLU

Na intenção de atenuar os problemas apresentados pela função *ReLU*, algumas variações foram desenvolvidas. Uma delas é a *Leaky ReLU* (MAAS, 2013), Figura 7, que apresenta uma pequena inclinação no eixo negativo, acarretando na resolução do problema *dying ReLU.*

Figura 7 - Função de ativação *Leaky ReLU*

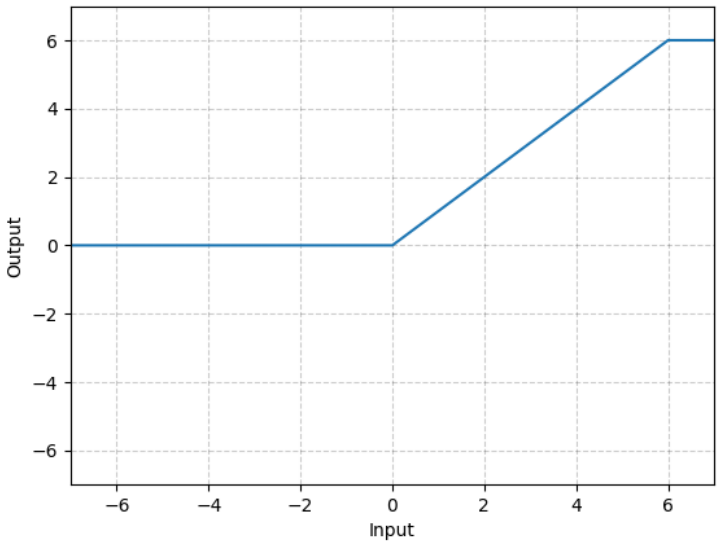


Fonte: Retirado da internet. Disponível em https://towardsdatascience.com/what-are-activation-functions-in-deep-learning-cc4f01e1cf5c. Acesso em 30 de agosto de 2022.

2.2.1.5 ReLU6

Uma outra variante é a função de ativação *ReLU6* (KRIZHEVSKY e HINTON, 2010), Figura 8, que apresenta uma tentativa de resolução do problema da explosão do gradiente. Para isso, essa função de ativação apresenta um limiar de ativação para valores de entradas muito altos.

Figura 8 - Função de ativação *ReLU6*

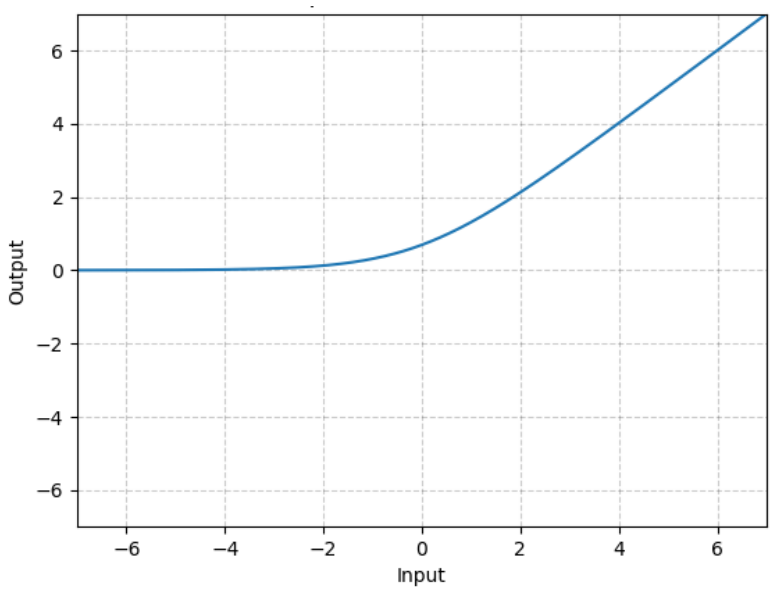


Fonte: Retirado da internet. Disponível em https://paperswithcode.com/method/relu6. Acesso em 30 de agosto de 2022.

2.2.1.6 Softplus

A função de ativação *softplus* (ZHENG *et al.*, 2015), Figura 9, pode ser interpretada como uma versão suavizada da função *ReLU*. Devido a isso, uma de suas vantagens é a suavidade nas regiões próximas de 0, evitando mudanças abruptas ao modelo se deparar com entradas próximas a essa faixa de valores.'

Figura 9 - Função de ativação *softplus*

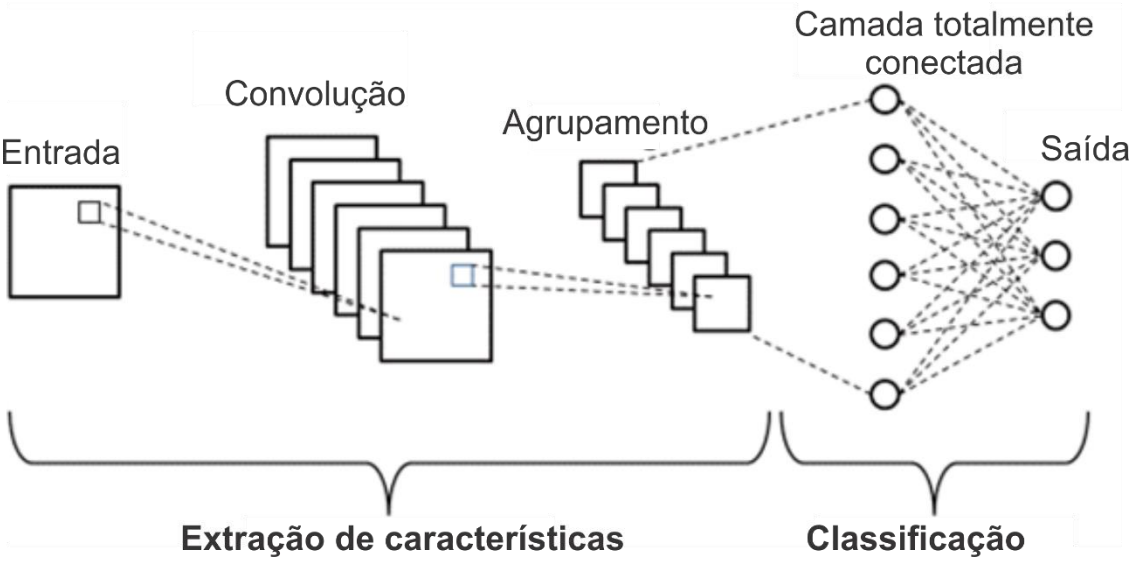


Fonte: Retirado da internet. Disponível em https://pytorch.org/docs/stable/generated/torch.nn.Softplus.html. Acesso em 30 de agosto de 2022.

2.3 REDES NEURAIS CONVOLUCIONAIS

As redes neurais convolucionais, em inglês *convolution neural networks* (*CNN)*, são um tipo rede neural artificial com grande capacidade de aprendizado a partir de um número menor de parâmetros, comparando com redes neurais padrão (*fully connected, ou totalmente conectada*). Essa característica se deve à presença de um conjunto de camadas no início da rede, que permitem extrair características específicas dos dados de entrada, reduzindo a dimensionalidade dos dados para a classificação, Figura 10. Essas camadas iniciais comumente são formadas por camadas convolucionais e camadas de agrupamento.

Figura 10 - Arquitetura de uma rede convolucional

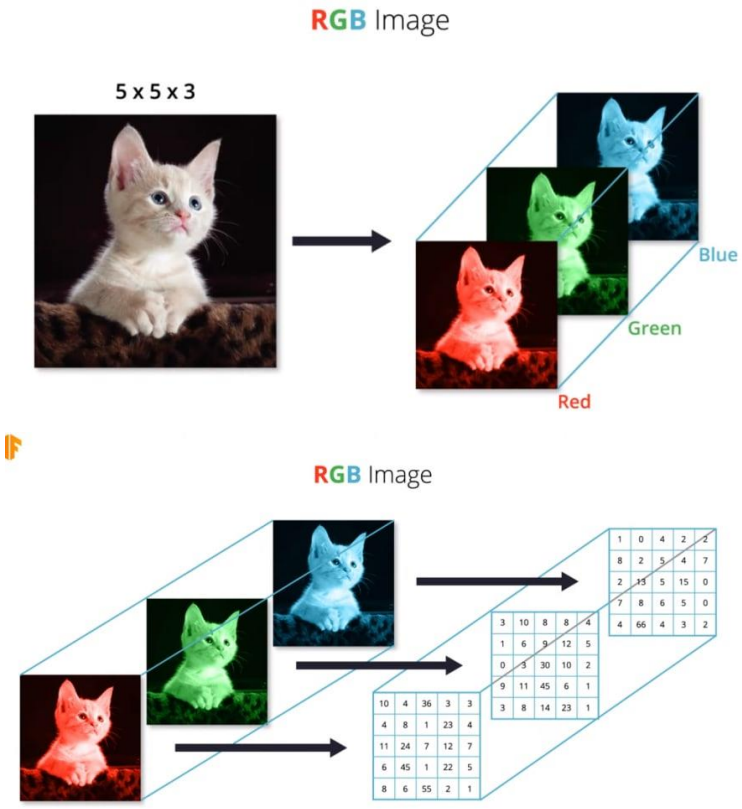


Fonte: Adaptado da internet. Disponível em https://medium.com/mlearning-ai/convolutional-neural-networks-cnns-1ed0f1f8e176. Acesso em 30 de agosto de 2022.

2.3.1 Canais

Diferente de outras redes artificiais, as *CNNs* atuam com dados bidimensionais (como por exemplo imagens), representadas por matrizes 2D. Essas representações bidimensionais podem ser estendidas em uma terceira dimensão, compondo o que são chamados de canais, como no caso de imagens com representações RGB, que possuem 3 canais, Figura 11.

Figura 11 - Representação de uma imagem RGB



Fonte: Retirado da internet. Disponível em https://dev.to/sandeepbalachandran/machine-learning-going-furthur-with-cnn-part-2-41km. Acesso em 30 de agosto de 2022.

2.3.2 Camada convolucional

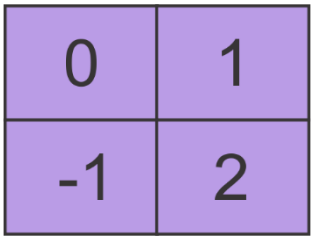
As camadas convolucionais são um dos elementos mais importantes das *CNNs*. Consistem de um conjunto de filtros, chamados de *kernels*, que realizam operações de convolução nos dados de entrada, de forma a extrair características durante o processamento desses dados, gerando saídas em cada camada, comumente chamadas de mapas de ativação, ou mapas de características.

2.3.2.1 Kernel

Um *kernel* pode ser descrito como uma matriz de números, representando cada um os pesos desse *kernel,* Figura 12. Esses pesos são inicializados com valores aleatórios, que são atualizados durante o aprendizado, de forma a

permitir o *kernel* a aprender a extração de uma característica significativa para a classificação dos dados. As *CNNs* possuem múltiplos *kernels* em cada camada, de forma a extrair diferentes características em cada etapa do processamento. Essa extração de características pelos *kernels* é realizada a partir de uma operação de convolução.

Figura 12 - Exemplo de um kernel *2 x 2*

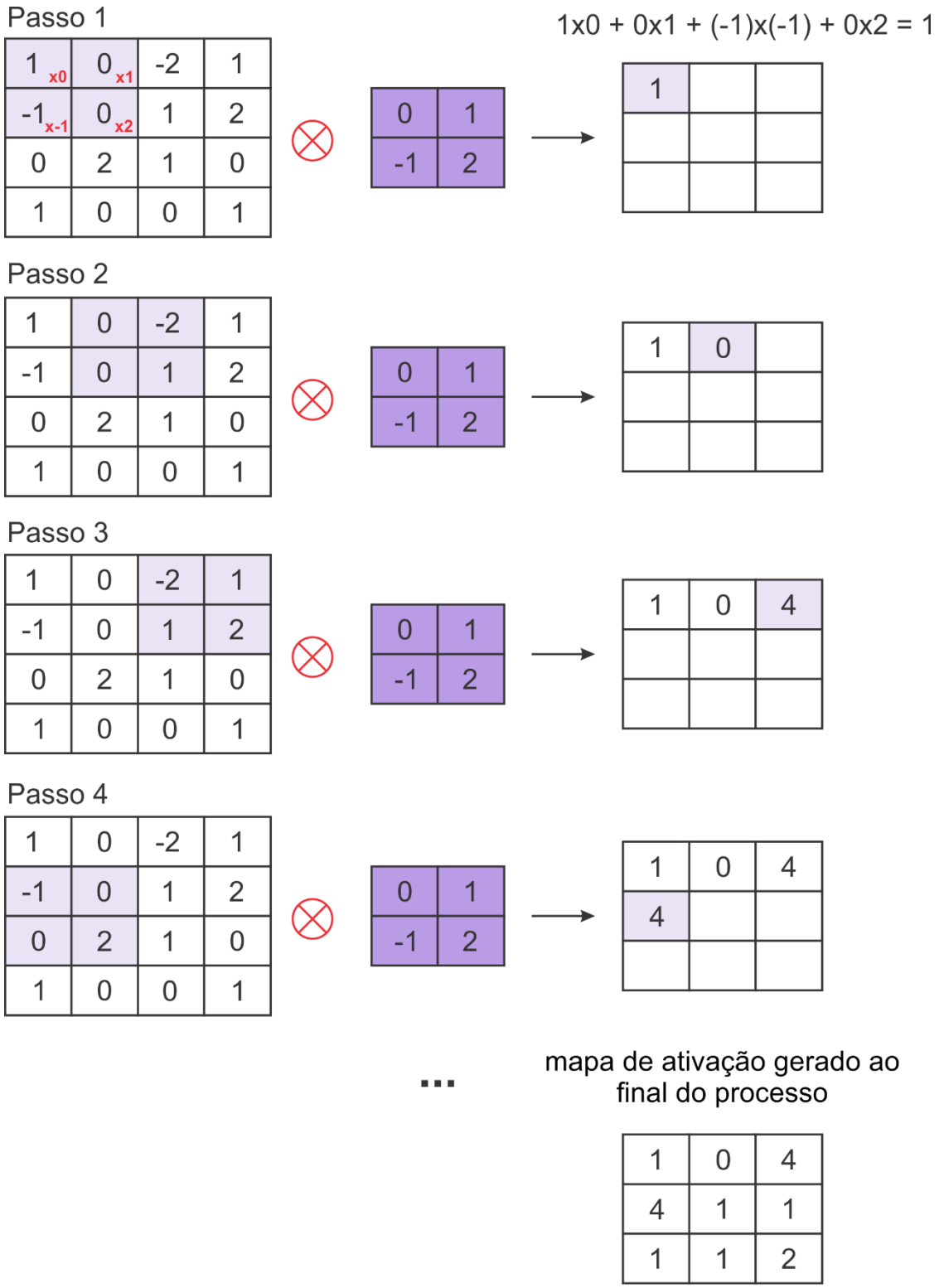


Fonte: adaptado de Ghosh *et al.* (2020)

2.3.2.2 Operação de convolução

Na operação de convolução, cada um dos pesos dos *kernels* multiplicam respectivos valores dos dados de entrada e em seguida somados. Esse processo é repetido deslizando-se horizontalmente e verticalmente o *kernel* ao longo dos valores de entrada, de forma a gerar na saída um mapa de ativação, como ilustrado na Figura 13.

Figura 13 - Ilustração da operação de convolução

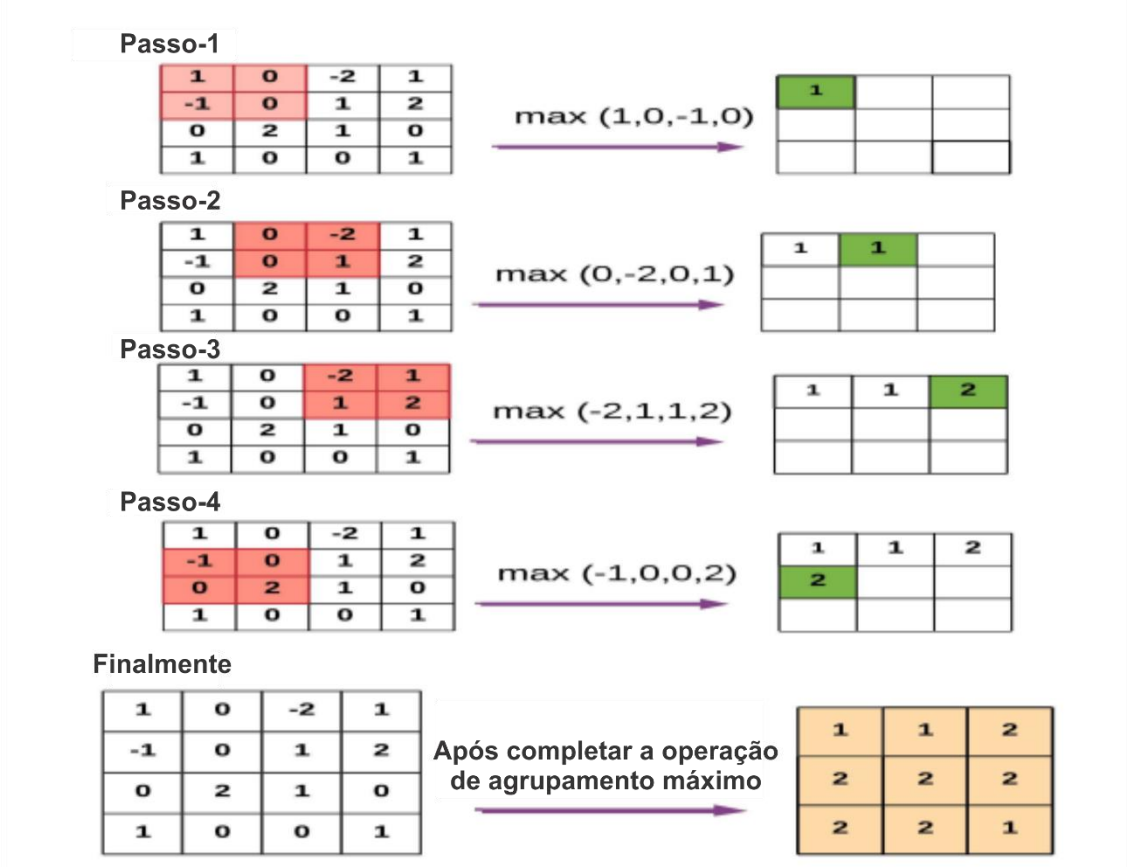


Fonte: Adaptado de Ghosh *et al.* (2020)

2.3.3 Camada de agrupamento (*pooling*)

As camadas de agrupamento são utilizadas para transformar os mapas de ativação em tamanhos menores, de forma a diminuir a dimensionalidade dos dados, mas preservando características dominantes. Similar à operação de convolução, as operações de agrupamento também percorrem horizontalmente e verticalmente pelo mapa de ativação. Diferentes tipos de operação podem ser realizados para o agrupamento, como o agrupamento máximo, que capta apenas o maior valor dentro da janela de agrupamento, Figura 14.

Figura 14 - Ilustração da operação de agrupamento máximo



Fonte: Adaptado de Ghosh *et al.* (2020)

Também existem outras variações, como agrupamento médio, que realiza a média dos valores, ou agrupamento da soma, que realiza a soma, como ilustrado na Figura 15.

Figura 15 - Exemplos de operações de agrupamento

Fonte: Adaptado da internet. Disponível em https://medium.com/analytics-vidhya/your-handbook-to-convolutional-neural-networks-628782b68f7e. Acesso em 30 de agosto de 2022.

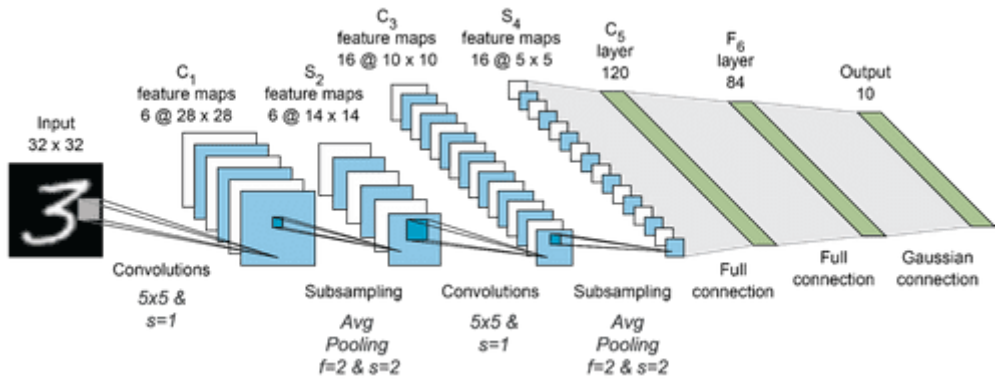
2.3.4 Exemplos de diferentes modelos de redes convolucionais

As redes *CNNs* se tornaram bastante populares para diversos problemas, principalmente relacionados a imagens. Diversas arquiteturas foram propostas na literatura, de forma a resolver problemas cada vez mais complexos. Nesta seção são descritos diferentes modelos de *CNNs* desenvolvidos ao longo dos anos.

2.3.4.1 LeNet-5

Um dos primórdios das redes convolucionais, a LeNet-5 foi desenvolvida por LeCun *et al.* (1998), de forma a solucionar problema de classificação de dígitos escritos à mão. Essa rede consiste de duas camadas convolucionais e duas camadas de agrupamento, seguido de três camadas totalmente conectadas, como ilustrado na Figura 16, possuindo aproximadamente 60 mil parâmetros.

Figura 16 - Arquitetura da LeNet-5

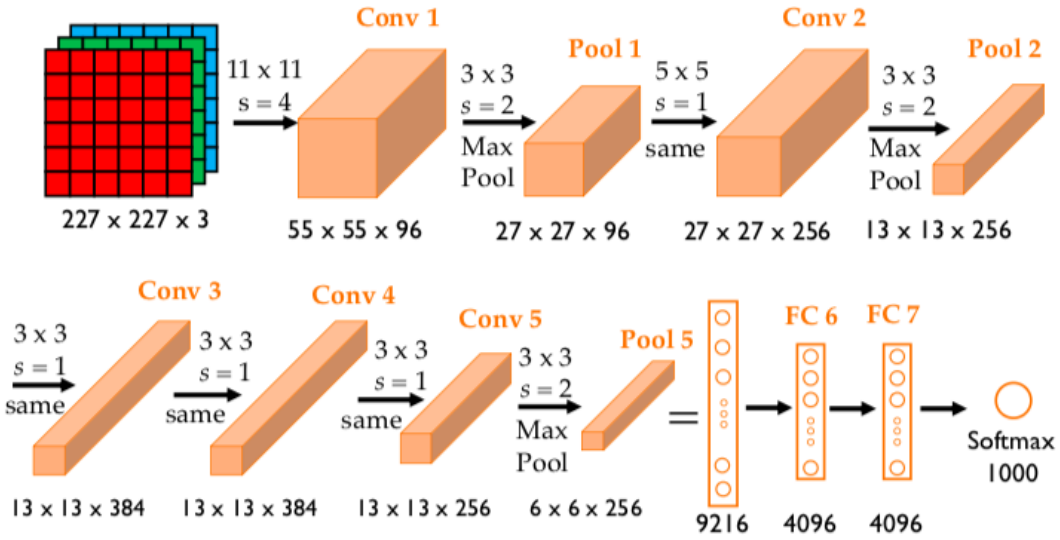


Fonte: Retirado da internet. Disponível em https://www.ismailmebsout.com/Convolutional%20Neural%20Network%20-%20Part%202/. Acesso em 30 de agosto de 2022.

2.3.4.2 AlexNet

Devido ao limite computacional e baixa quantidade de dados da época, as redes convolucionais demoraram alguns anos para se popularizar. A mudança de paradigma veio, principalmente, após a *ILSVRC* (*Image Net Large Scale Visual Recognition Competition*) 2012, que é uma competição de classificação de imagens. Nesse ano, uma nova arquitetura de rede convolucional foi proposta, que permitiu um avanço bastante substancial na pontuação da competição. Essa rede permitiu diminuiu a taxa de erro da competição anterior de 25.8% para 16.4%, sendo a primeira vez que uma rede convolucional ganha a competição. A rede proposta é chamada de AlexNet (KRIZHEVSKY, 2012), Figura 17, e proporcionou avanços significativos na área de aprendizagem profunda, promovendo a popularização das redes neurais convolucionais.

Figura 17 - Arquitetura da AlexNet



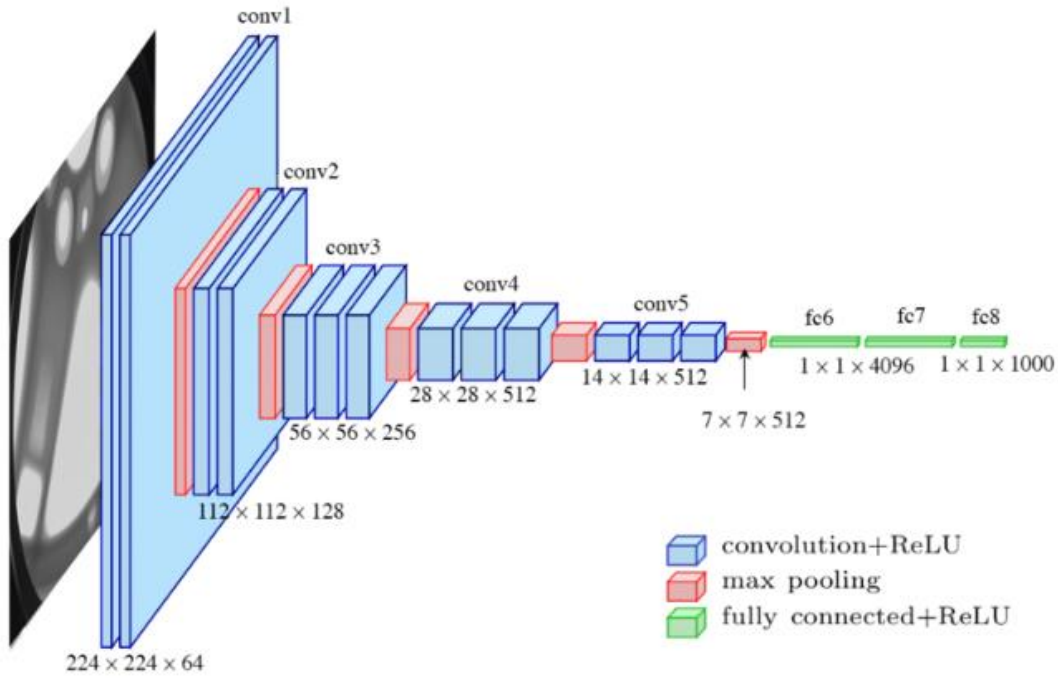
Fonte: Retirado da internet. Disponível em https://blog.devgenius.io/alexnet-the-net-that-surpassed-cnns-5d551ba1b901. Acesso em 30 de agosto de 2022.

Com 60 milhões de parâmetros, a AlexNet possui um número maior de camadas, em relação à LeNet-5, que contém apenas 60 mil parâmetros. No entanto, foi possível treina-la de forma mais robusta. Isso se deve ao fato da disponibilidade de maiores recursos disponíveis na época, com uma maior quantidade de dados, além da utilização de *GPUs* para o processamento das informações.

2.3.4.3 VGG

Em 2014, surgiu outra rede vencedora da ILSVRC, a VGG (SIMONYAN e ZISSERMAN, 2014), Figura 18, que alcançou uma taxa de erro de 7.3% na competição. Nessa nova arquitetura, algumas modificações foram realizadas, em relação à AlexNet. Uma delas é a utilização de apenas *kernels 3x3* nas camadas de convolução. Além disso, essa rede possui, na versão padrão, 16 camadas, com 138 milhões de parâmetros.

Figura 18 - Arquitetura da VGG-16

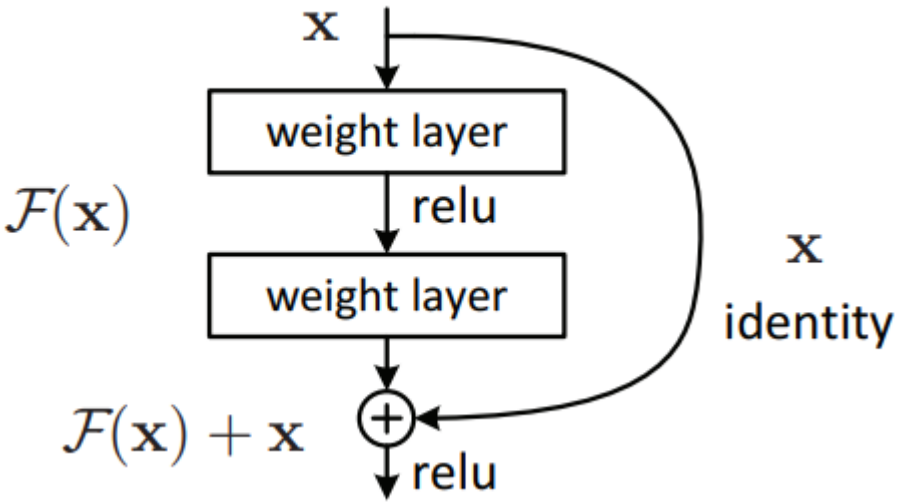


Fonte: Ferguson *et al*. (2017)

2.3.4.4 ResNet

Até então, era evidenciado que o aumento no número de camadas das redes convolucionais, de certa forma, permitia um maior desempenho dos modelos. No entanto, foi observado que com o aumento demasiado da quantidade de camadas provocava o problema da dissipação do gradiente. De forma a solucionar esse problema, foi proposta uma nova mudança na arquitetura das redes convolucionais, dando origem aos modelos ResNet (HE, 2016). Para isso, foi introduzido um conceito chamado de *skip connections,* Figura 19.

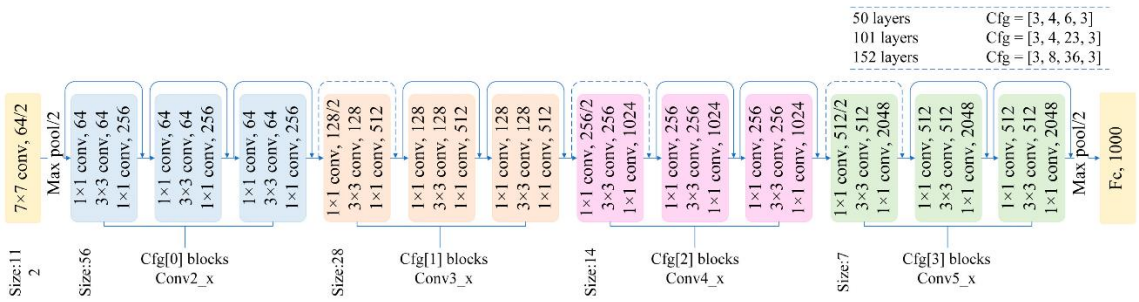
Figura 19 - *Skip connection*



Fonte: Retirado da internet. Disponível em https://programmathically.com/an-introduction-to-residual-skip-connections-and-resnets/. Acesso em 30 de agosto de 2022.

No conceito de *skip connections,* há uma conexão adicional entre camadas calculada pela matriz identidade, ou seja, sem modificação dos valores da ativação anterior. A vantagem dessa abordagem é que diminui o problema da dissipação do gradiente, pois a retropropagação por essas conexões adicionais não é atenuada. Essa mudança possibilitou o treino de redes ainda mais profundas, como a ResNet-152, Figura 20, com 152 camadas e 60 milhões de parâmetros, campeã da ILSVRC 2015, alcançando uma taxa de erro de apenas 3,57%.

Figura 20 - Arquitetura da ResNet



Fonte: Deng, L. *et al.* (2020)

2.4 TRANSFERÊNCIA DE APRENDIZAGEM

Usualmente, os modelos de aprendizagem profunda necessitam de grandes quantidades de dados no treinamento, de forma a alcançar um alto desempenho. No entanto, essa não é uma, realidade possível na maioria das tarefas em problemas de aprendizagem de máquina. De forma a contornar esse problema, métodos de transferência de aprendizado foram propostos, de forma a possibilitar modelos aprendizagem profunda alcançarem alto desempenho com uma quantidade reduzida de dados. Esse tipo de abordagem consiste em aproveitar o aprendizado prévio de um modelo treinado em uma quantidade grande de dados, de forma a aprimorar o aprendizado em uma tarefa que possui quantidade limitada de dados. Algumas técnicas de transferência de aprendizado são descritas nesta seção.

2.4.1 Finetuning

A utilização de pré-treino com *finetuning* (SHARIF RAZAVIAN *et al.*, 2014) é uma das técnicas mais populares de transferência de aprendizado. Basicamente, consiste em inicializar uma rede com os pesos de alguma rede treinada previamente em um conjunto de dados com grande quantidade de amostras. A ideia por trás dessa abordagem, é que as redes convolucionais aprendem nas camadas mais profundas a identificar características mais abstratas. Assim, apenas um pequeno ajuste fino é necessário para a rede aprender a desempenhar bem em um conjunto de dados diferente. Por exemplo, um modelo treinado em uma grande quantidade de dados de objetos pode ser utilizado para o *finetuning* em uma tarefa de identificação de pessoas, mesmo com poucas imagens disponíveis de pessoas. A desvantagem dessa abordagem é a necessidade de manter a mesma arquitetura de modelo para o *finetuning*. Além disso, há uma dependência de similaridade entre as tarefas do modelo pré-treinado e a tarefa a ser aprendida pelo *finetuning*, que em caso de ser bastante discrepante, pode acarretar em um baixo desempenho na nova tarefa.

2.4.2 Learning to Transfer What and Where

Nesta seção, é descrito brevemente o método *Learning to Transfer What and Where* (*L2T-ww*) (JANG *et al.*, 2019), que inspirou a técnica proposta neste trabalho. A ideia por trás do *L2T-ww* é de que os mapas de ativação intermediários, identificadas por uma rede convolucional treinada, possuem conhecimento que pode ser aproveitado. Nesse sentido, forçar uma nova rede a reproduzir os mesmos mapas de ativação de uma rede treinada pode auxiliar no treinamento. Assim, o seguinte objetivo l_2, Equação 3, pode ser minimizado, de forma a ensinar uma rede *objetivo (target*) a identificar características de maneira similar à rede treinada fonte *(source)*:

$$\left|\left|r_{\theta}\left(T_{\theta}^{n}(x)\right) - S^{m}(x)\right|\right|_{2}^{2} \tag{3}$$

onde x representa os dados de entrada, $S^m(x)$ são os mapas de ativação intermediários da m^{th} camada de uma rede fonte pré-treinada e $T_\theta^n(x)$ os mapas de ativação intermediários da n^{th} camada de uma rede objetivo a ser treinada. Além disso, r_θ representa uma transformação linear, como por exemplo uma convolução, de forma a igualar as dimensões dos mapas de ativação de ambas as redes, e θ representa os parâmetros a serem aprendidos da rede objetivo e da transformação linear r.

A partir do conceito descrito acima, os autores propuseram a utilização de *meta-redes*, que aprendem, automaticamente durante o treino, quais regiões de conhecimento da rede fonte são mais importantes a serem transferidas para a rede objetivo. Assim, essas meta-redes aprendem quais canais são mais importantes a serem enfatizados na transferência de conhecimento (*what to transfer*) e também aprendem a decidir de quais camadas da rede fonte para quais camadas da rede objetivo o conhecimento deve ser transferido (*where to transfer*).

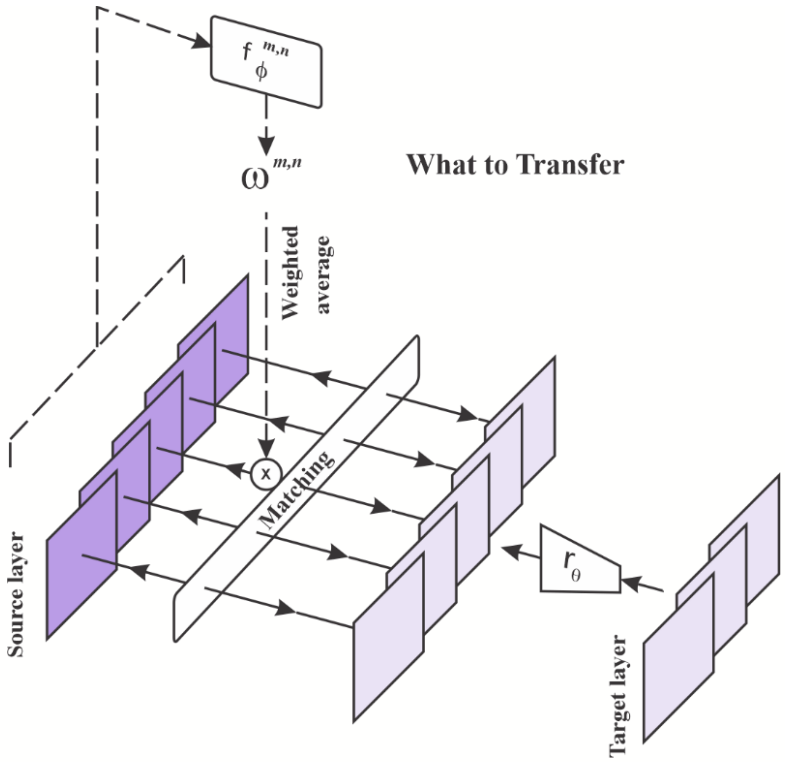
2.4.2.1 What to transfer

De forma a definir quais pares de canais são mais úteis para transferir conhecimento em uma determinada tarefa, é definido os pesos de importância $\omega_c^{m,n}$ do canal c, com com $\sum_c \omega_c^{m,n} = 1$. Assim, a *wheighted feature matching loss* é ponderada por esses pesos, estabelecendo-se assim a Equação 4.

$$L_{\text{wfm}}^{m,n}(\theta|x, \omega^{m,n}) = \frac{1}{HW}\sum_c \omega_c^{m,n} \sum_{i,j} \left(r_\theta\left(T_\theta^n(x)\right)_{c,i,j} - S^m(x)_{c,i,j}\right)^2 \quad (4)$$

onde o somatório interno é em relação a $i \in \{1,2,\dots H\}$ e $j \in \{1,2,\dots W\}$, onde $H \times W$ representa as dimensões espaciais de $r_\theta\left(T_\theta^n(x)\right)$ e $S^m(x)$. Os pesos de importância são obtidos considerando $\omega_c^{m,n}$ como uma função $f_\phi^{m,n}\left(S^m(x)\right)$. Essa função é aprendida utilizando uma meta-rede neural que utiliza os mapas de ativação da rede fonte como entrada e possui uma função *softmax* (BISHOP e NASRABADI, 2006) na saída. O símbolo ϕ representa os parâmetros dessa meta-rede, e o processo descrito é ilustrado na Figura 21.

Figura 21 - Ilustração do processo de *What to transfer*



Fonte: o autor (2022)

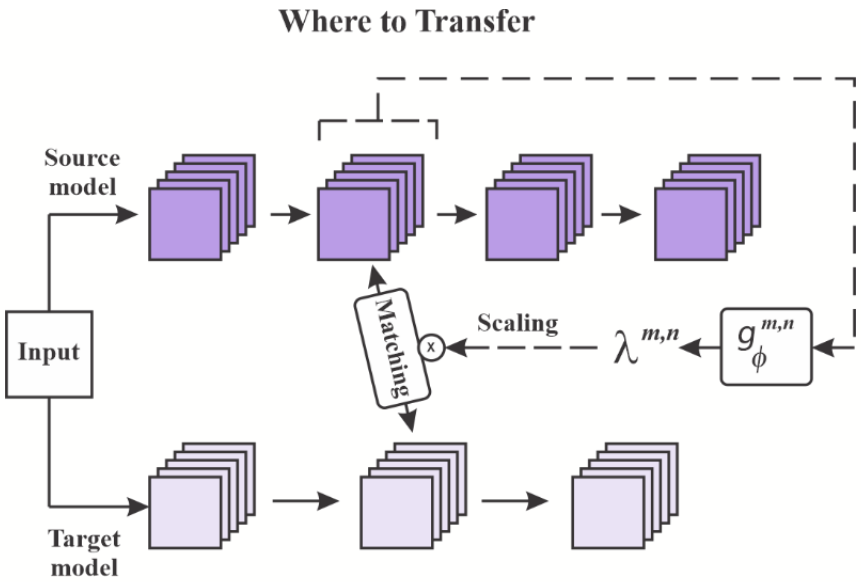
2.4.2.2 Where to transfer

Enquanto o processo de *What to transfer* trata da correspondência entre canais, o *Where to transfer* trata da correspondência entre camadas. Ou seja, o quanto enfatizar a transferência de aprendizado entre pares de camadas da rede fonte para a rede objetivo. Assim, para cada par de camada (m, n) de um conjunto de pares P, tem-se um parâmetro $\lambda^{m,n} \geq 0$ aprendido durante o treinamento. Esse parâmetro é utilizado para decidir quanto de conhecimento da m^{th} camada da rede fonte deve ser enfatizado a ser transferido para a n^{th} camada da rede objetivo. Dessa forma, a *weighted feature matching loss* combinada, ponderada por λ é dada pela Equação 5.

$$L_{\text{wfm}}(\theta|x, \phi) = \sum_{(m,n)\in P} \lambda^{m,n} L_{\text{wfm}}^{m,n}(\theta|x, \omega^{m,n}) \quad (5)$$

Os pesos $\lambda^{m,n}$ também são dados a partir de uma função $\lambda^{m,n} = g_{\phi}^{m,n}\left(S^m(x)\right)$, aprendida durante o processo de aprendizagem utilizando meta-redes. Essas meta-redes recebem os mapas de ativação da rede fonte como entrada e tem como saída os valores de $\lambda^{m,n}$, como ilustrado na Figura 22.

Figura 22 - Ilustração do processo *Where to transfer*



Fonte: o autor (2022)

A *loss* total, utilizada para o treinamento da rede objetivo é dada pela soma da *loss* original com a *loss* de *feature matching*, Equação 6.

$$L_{\text{total}} = L_{\text{org}}(\theta|x, y) + \beta L_{\text{wfm}}(\theta|x, \phi) \quad (6)$$

onde $\beta > 0$ é um hiperparâmetro e y representa o *ground-truth* das amostras.

3 OUTROS TRABALHOS RELACIONADOS

Diversos outros trabalhos também foram desenvolvidos como tentativa de resolver problemas relacionados à escassez de dados de treinamento. Nesta seção são descritos alguns estudos relevantes relacionados a esse problema.

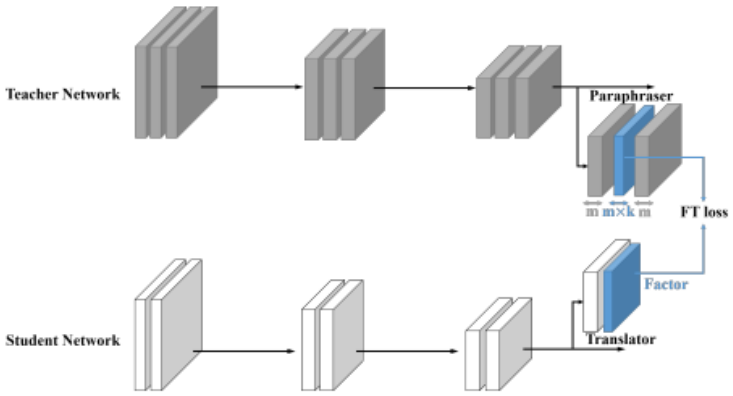
3.1 MÉTODO DE ROMERO *ET AL*. (2014)

De forma a aprimorar a técnica de *finetuning*, Romero *et al.* (2014) propuseram um esquema *professor-aluno*, utilizando uma rede pré-treinada para auxiliar no treinamento de uma nova rede. Para isso, foram utilizados os mapas de ativação da rede pré-treinada, de forma a guiar o treino da nova rede, utilizando l_2 *matching loss* para conduzir a rede aluno a reproduzir as características da rede professor pré-treinada. Um dos problemas dessa abordagem é não haver um esquema automático para decidir quais mapas de ativação são mais importantes para auxiliar no processo.

3.2 MÉTODO DE KIM, PARK E KWAK (2018)

Outro método semelhante ao esquema professor-aluno também foi proposto por Kim, Park e Kwak (2018). No entanto, os autores utilizaram duas redes convolucionais, as quais chamaram de parafraseador e tradutor, de forma a auxiliar a transferência de aprendizagem da rede professor para a rede aluno. Dessa forma, o parafraseador utiliza como entrada os mapas de ativação da rede professor e o tradutor realiza o mesmo processo com a rede aluno. Assim, esses dados são processados por ambas as redes auxiliares, e a saída da rede tradutor é guiada a reproduzir a saída da rede parafraseador, como ilustra a Figura 23.

Figura 23 - Ilustração do esquema parafraseador-tradutor



Fonte: Kim, Park e Kwak (2018)

3.3 MÉTODO DE GUO *ET AL*. (2019)

Uma abordagem baseada no esquema de pré-treino com *finetuning* foi proposto por Guo *et al.* (2019). Nessa técnica, os autores propuseram a utilização e uma rede chamada de *policy network* durante o processo de *finetuning*. Dessa forma, essa *policy network* tem a função de selecionar quais camadas devem ser congeladas ou treinadas, para cada exemplo durante o treinamento. Apesar de bastante promissor, esse método tem a limitação de necessitar que o modelo pré-treinado e o modelo a ser treinado devem possuir a mesma arquitetura.

3.4 MÉTODO DE *FINETUNING* INTERMEDIÁRIO

De forma a transpor a necessidade de correlação entre tarefa fonte e tarefa objetivo durante a transferência de aprendizagem, alguns estudos propuseram a utilização de uma tarefa intermediária durante o processo, antes de realizar o *finetuning* na tarefa objetivo (PRUKSACHATKUN *et al.*, 2020; PHANG, FÉVRY e BOWMAN, 2018; WANG *et al.*, 2018). Embora esse tipo de método resultar em bons resultados em alguns casos, é uma abordagem que demanda um conjunto de dados intermediário de tamanho considerável, além de aumentar significativamente o custo computacional.

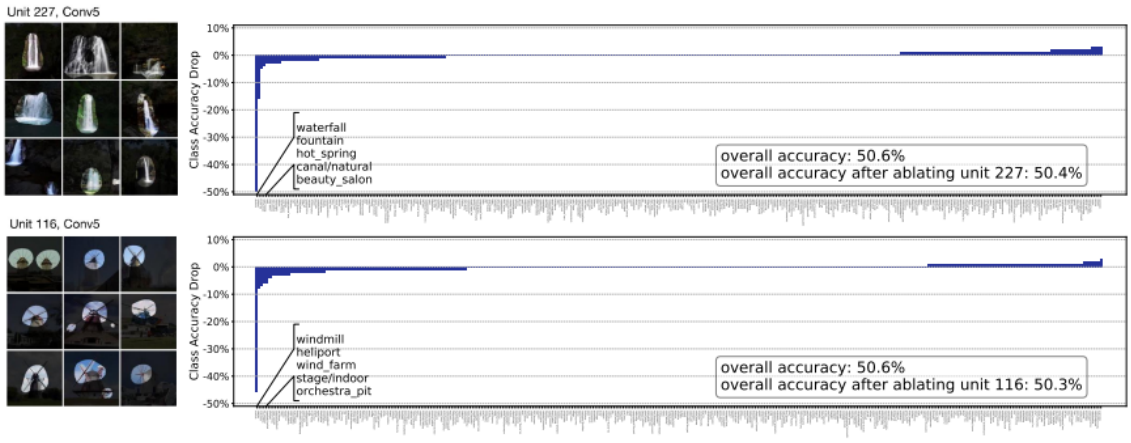
4 MÉTODO PROPOSTO

Nesta seção, é descrito o método proposto, abordando as principais ideias e conceitos que inspiraram a técnica, além de elucidar a formalização matemática que descreve a técnica.

4.1 LEARNING TO TRANSFER WHAT, WHERE AND WHICH

Alguns estudos recentes tem explorado a importância de diferentes parâmetros no desempenho das redes neurais artificiais. Nesse sentido, alguns estudos puderam identificar que diferentes microrregiões das redes convolucionais possuem diferentes influências na saída final da rede (BAU *et al., 2020;* BACH *et al.,* 2015; ZHOU *et al.*, 2018). De forma mais detalhada, esses estudos puderam evidenciar que, analisando os mapas de ativação das redes convolucionais, alguns neurônios têm maior influência que outras na realização de tarefas, como a correta classificação de uma imagem. Por exemplo, nos estudos conduzidos por Zhou *et al.* (2018), os autores identificaram que a remoção de determinados neurônios, apesar de não afetar, de maneira significativa, a classificação geral das imagens, afeta consideravelmente a classificação de determinadas classes, como ilustra a Figura 24. Ou seja, os autores puderam evidenciar que determinadas unidades individuais têm importâncias e papéis diferentes, a depender da tarefa realizada.

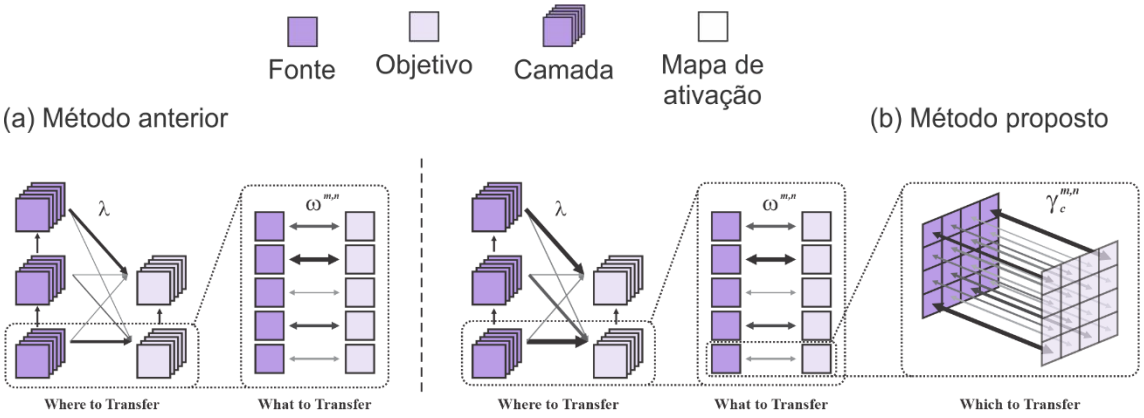
Figura 24 - Consequências da remoção de diferentes unidades individuais



Fonte: Zhou *et al.* (2018)

Essa evidência trouxe a indagação de se diferentes unidades individuais têm diferentes influências, então também diferentes regiões (*pixels*) dos mapas de ativação *gerados* podem ter diferentes importâncias na transferência de conhecimento. Baseado nisso, este trabalho propõe o método *Learning to Transfer What, Where and Which* (*L2T-www*), que é um aprimoramento do método *L2T-ww*, proposto por Jang *et al.* (2019). Par isso, é proposta uma meta-rede adicional (*which to transfer*), que tem o papel de ponderar a transferência de aprendizado ao nível de *pixels*, para além dos níveis de camadas e canais do método anterior (L2T-ww), Figura 25. Na figura, as diferentes espessuras das setas representam diferentes pesos na transferência de conhecimento.

Figura 25 - a) Método anterior e b) Método proposto



Fonte: o autor (2022)

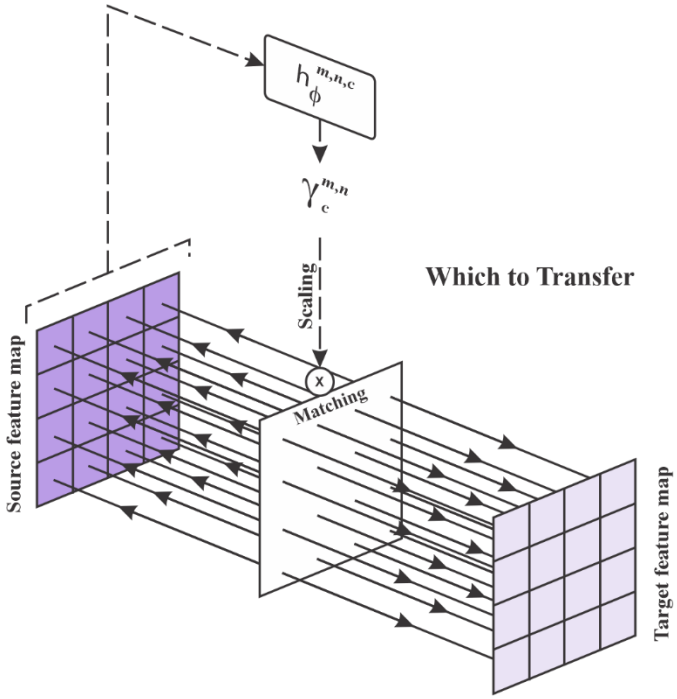
4.1.1 Which to transfer

A ideia principal da proposta é a de ponderar a transferência de aprendizado no nível de *pixels*. Para isso, é considerado um parâmetro $\gamma_{c,i,j}^{m,n}$, que tem por objetivo enfatizar a transferência de conhecimento da unidade de ativação i,j, do canal c, partindo da m^{th} camada da rede fonte para a n^{th} camada da rede objetivo. Assim, de forma a gerar esse efeito, é proposta uma modificação do método anterior, resultando em uma alteração da expressão da *weighted feature matching loss*, Equação 7, abaixo.

$$L_{\text{wfm}}^{m,n}(\theta|x,\omega^{m,n}) = \frac{1}{HW}\sum_c \omega_c^{m,n} \sum_{i,j} \gamma_{c,i,j}^{m,n} \left(r_\theta\left(T_\theta^n(x)\right)_{c,i,j} - S^m(x)_{c,i,j}\right)^2 \qquad (7)$$

Os pesos do parâmetro $\gamma_{c,i,j}^{m,n} > 0$ são dados a partir de uma função $\gamma_{c,i,j}^{m,n} = h_{\phi}^{m,n}\left(S^{m}(x)\right)$. De forma a calcular essa função, é utilizada uma meta-rede neural treinada durante o processo. Para isso, essa meta-rede utiliza como entrada os mapas de ativação da rede fonte e tem como saída os pesos do parâmetro γ, como ilustrado para um canal específico c de um par de camadas (m, n) na Figura 26.

Figura 26 - Ilustração do *Which to transfer*



Fonte: o autor (2022)

Como resultado, essa modificação permite que o método aprenda a transferir conhecimento não só enfatizando canais (*what to transfer) ou* camadas (*where to transfer*) apropriados, mas também quais *pixels* são mais importantes a serem enfatizadas na transferência de conhecimento (*which to transfer*). Dessa forma, o processo se torna mais refinado, permitindo um aumento de desempenho da rede objetivo.

5 EXPERIMENTOS E DISCUSSÕES

Nesta seção são descritos os experimentos realizados para validar o método proposto. Para isso, o método foi utilizado em diversas tarefas diferentes e comparado com o método anterior (JANG *et al.*, 2019), que inspirou a nova técnica.

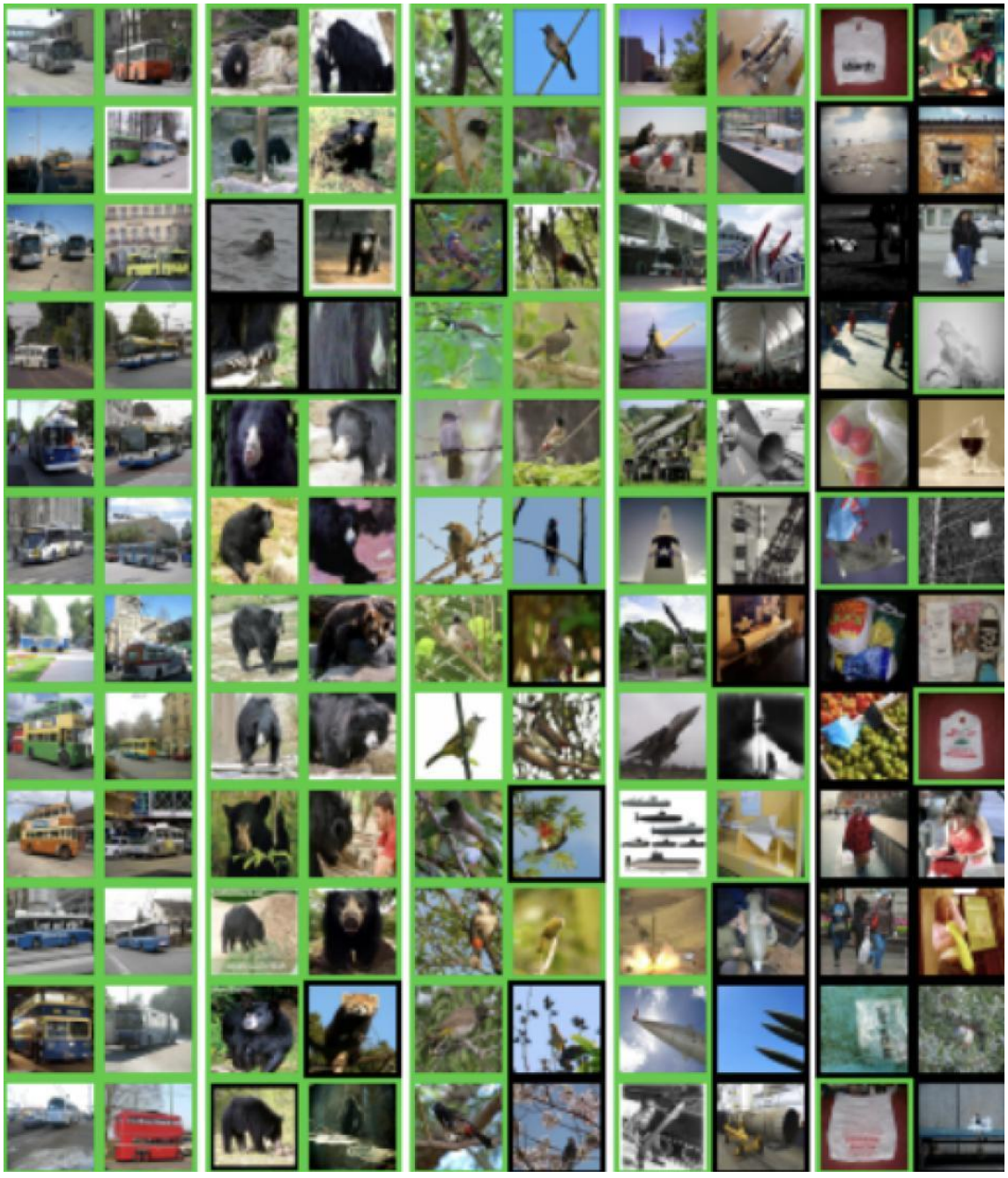
5.1 MODELOS E TAREFAS UTILIZADAS

Em todos experimentos realizados, foram utilizados como rede fonte uma ResNet de 32 camadas, pré-treinada no *Tiny ImageNet* (LE e YANG, 2015), o qual é um conjunto de dados com 100 mil imagens de 200 objetos diferentes (500 imagens para cada objeto). A partir disso, foi utilizada uma rede VGG com 9 camadas como rede objetivo, testando o método proposto para treina-la em diferentes tarefas, como CIFAR-10 (KRIZHEVSKY *et al.*, 2009), CIFAR-100 (KRIZHEVSKY *et al.*, 2009) e STL-10 (COATES, NG e LEE, 2011).

5.1.1 Tiny ImageNet

O *Tiny ImageNet* (LE e YANG, 2015) é uma versão reduzida do conjunto de dados *ImageNet* (DENG, J. *et al.*, 2009), utilizado no desafio do *ILSVRC*. É composto por 100 mil imagens coloridas, de tamanho 64x64, divididas em 200 classes com 500 imagens cada. A Figura 27 ilustra alguns exemplos de imagens contidas no *Tiny ImageNet*.

Figura 27 - Exemplos de imagens do *Tiny ImageNet*



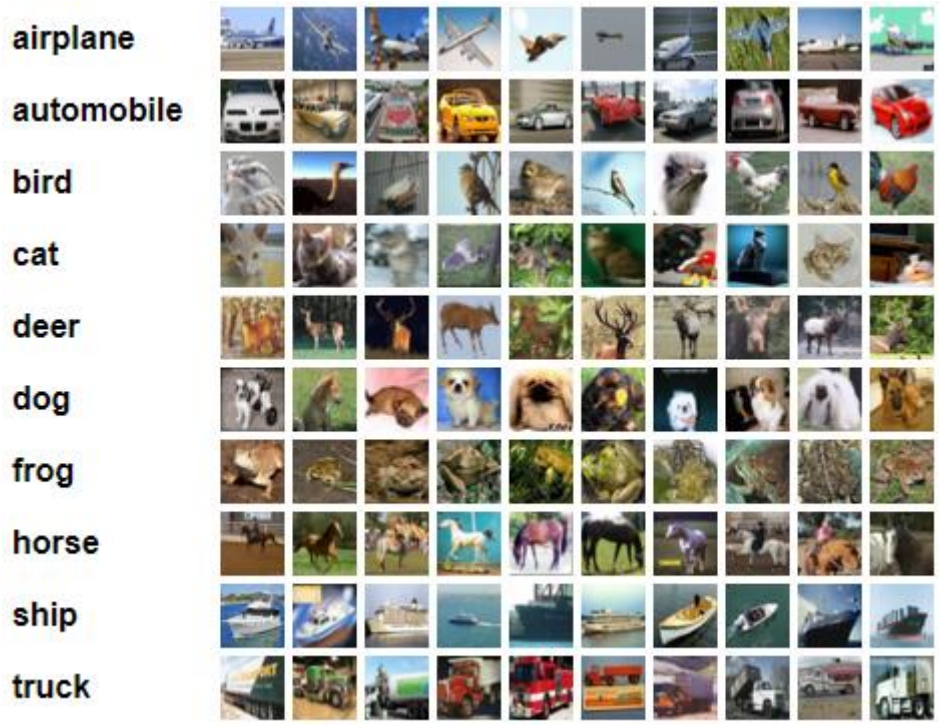
Fonte: Retirado da internet. Disponível em https://paperswithcode.com/dataset/tiny-imagenet. Acesso em 30 de agosto de 2022.

5.1.2 CIFAR-10

Os conjuntos de dados CIFAR são subconjuntos rotulados do conjunto de dados *80 million tiny images* (TORRALBA, FERGUS e FREEMAN, 2008). O CIFAR-10 é um conjunto de dados que possui 60 mil imagens coloridas, de

tamanho 32x32 pixels. Essas imagens são divididas em 10 classes de 6 mil imagens cada. As classes presentes no conjunto de dados são “automóvel”, “avião”, “pássaro”, “veado”, “gato”, “cachorro”, “sapo”, “cavalo” “caminhão” e “navio”. Separado em treino e teste, esse conjunto de dados é dividido, de forma balanceada, em 50 mil imagens de treino, com 5 mil imagens para cada classe e 10 mil imagens de teste, com 1 mil imagens de cada classe. A Figura 28 ilustra algumas imagens presentes no CIFAR-10.

Figura 28 - Exemplos de imagens do CIFAR-10

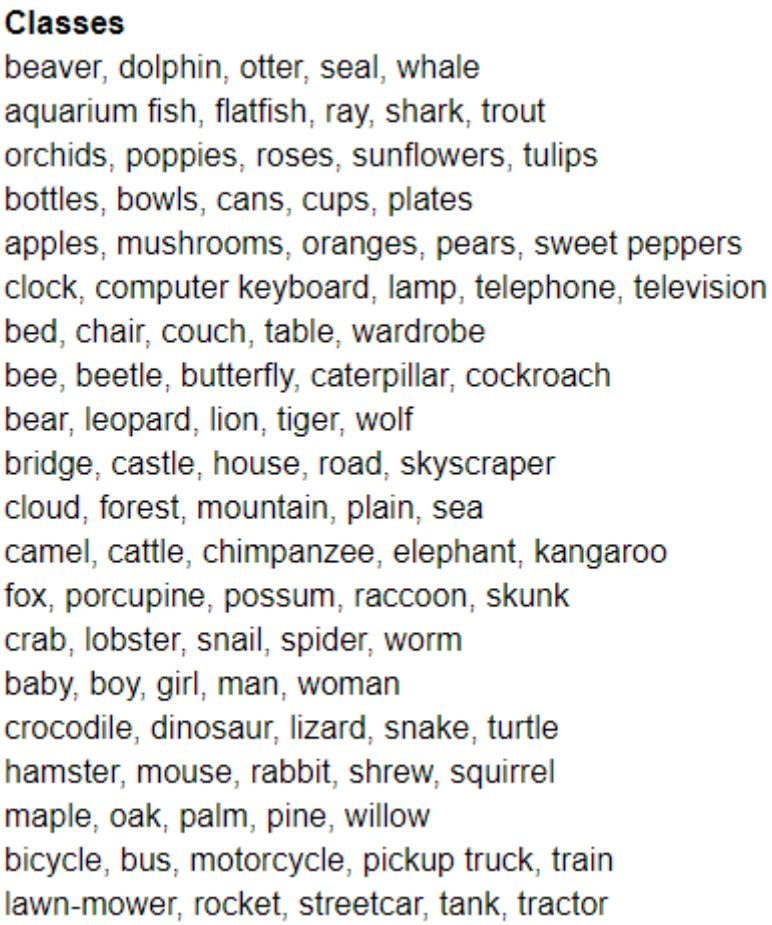


Fonte: Retirado da internet. Disponível em https://www.cs.toronto.edu/~kriz/cifar.html. Acesso em 30 de agosto de 2022.

5.1.3 CIFAR-100

O CIFAR-100 é composto por um outro subconjunto do *80 million tiny images* e contém uma maior variedade de classes, comparado ao CIFAR-10. Dessa forma, esse conjunto de dados possui um total de 100 classes, cada uma composta por 600 imagens, totalizando 60 mil imagens. As classes presentes neste conjunto de dados estão descritas na Figura 29.

Figura 29 - Classes presentes no CIFAR-100

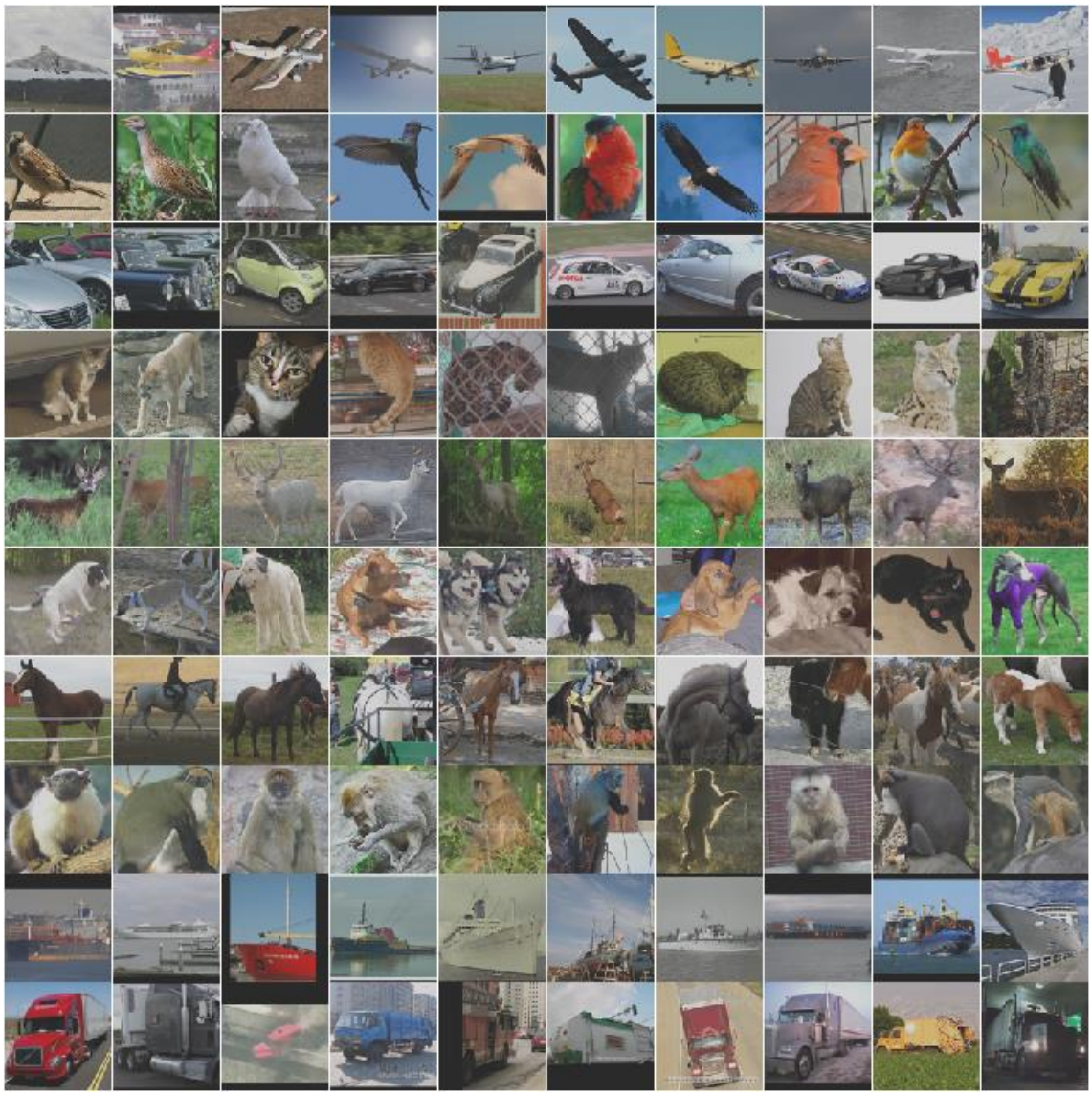


Fonte: Retirado da internet. Disponível em https://www.cs.toronto.edu/~kriz/cifar.html. Acesso em 30 de agosto de 2022.

5.1.4 STL-10

Assim como os CIFARs, o STL-10 também é um conjunto de dados de imagens coloridas. No entanto, é composto por imagens maiores, com dimensões de 96x96 pixels. Além disso, possui 10 classes, com 500 imagens de treino e 800 de teste por classe, totalizando 5 mil imagens para treinamento e 8000 para teste. As classes presentes neste conjunto de dados são: “*airplane*”, “*bird*”, “*car*”, “*cat*”, “*deer*”, “*dog*”, “*horse*”, “*monkey*”, “*ship*” e “*truck*”. A Figura 30 ilustra alguns exemplos de imagens presentes no STL-10.

Figura 30 - Exemplos de imagens do STL-10

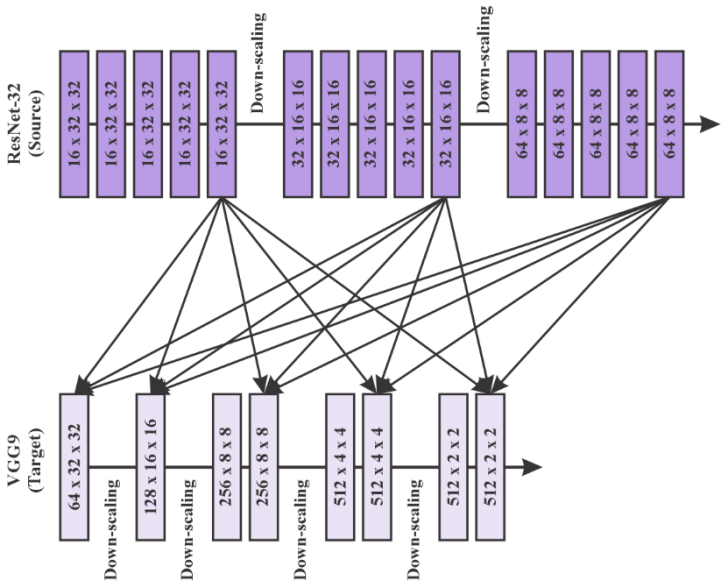


Fonte: Retirado da internet. Disponível em https://ai.stanford.edu/~acoates/stl10/. Acesso em 30 de agosto de 2022.

5.2 CONEXÕES ENTRE AS CAMADAS DOS MODELOS

Em todos experimentos, foram conectados os pares de camadas como ilustrado na Figura 31, resultando em 15 diferentes pares.

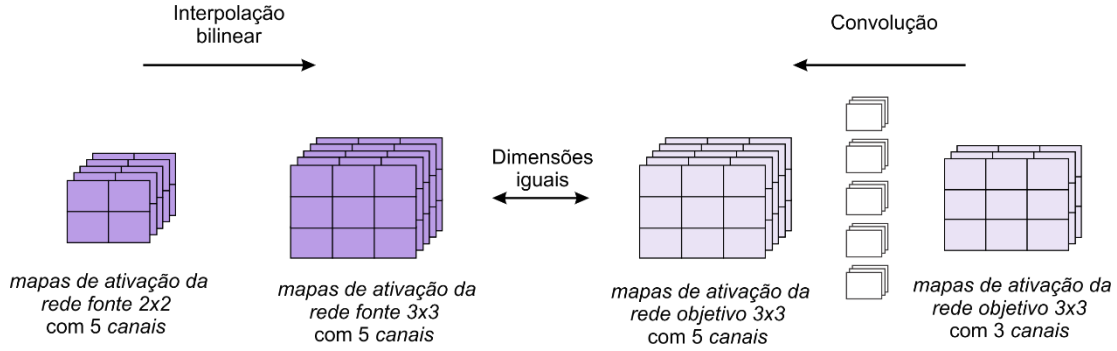
Figura 31 - Conexões entre camadas das redes fonte e objetivo



Fonte: o autor (2022)

Com o intuito de igualar as dimensões dos mapas de ativação de cada camada, foram implementadas algumas transformações. De forma a igualar a quantidade de características (canais), foram utilizadas convoluções, Figura 32, com pesos aprendidos durante o treinamento. Além disso, também foram utilizadas interpolações bilineares para igualar as dimensões espaciais de altura e largura, Figura 32.

Figura 32 - Ilustração do processo para igualar dimensões



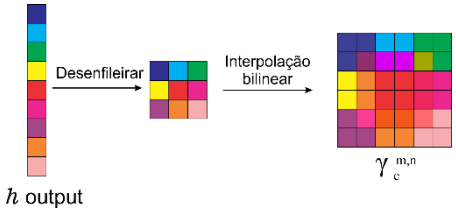
Fonte: o autor (2022)

5.3 META-REDES

Nas meta-redes f (*what to transfer*) e g (*where to transfer*), foram utilizadas redes neurais totalmente conectadas de 1 camada para cada par $(m,n) \in P$, onde P representa o conjunto de pares de camadas das redes fonte e objetivo, Figura 31. Dessa forma, essas meta-redes utilizam como entrada os mapas de ativação advindos da rede fonte, processados através de redes totalmente conectadas de 1 camada, tendo como saída os valores de $\omega_c^{m,n}$ e $\lambda^{m,n}$. Além disso, na meta-rede f foi utilizada a função de ativação ReLU6 (KRIZHEVSKY e HINTON, 2010) e, de forma a satisfazer $\sum_c \omega_c^{m,n} = 1$, utilizou-se *softmax* (BISHOP e NASRABADI, 2006) como função de ativação da meta-rede g.

Quanto à meta-rede h (*which to transfer*), também foram utilizadas redes neurais totalmente conectadas de 1 camada para cada canal c da m^{th} camada da rede fonte, em cada par de camadas $(m,n) \in P$. Essas redes recebem como entrada os mapas de ativação da rede e utilizam *softplus* (ZHENG *et al.*, 2015) como função de ativação. Os vetores de saída dessas redes são desenfileirados, sendo transformados em matrizes. Os elementos dessas matrizes são os pesos de $\gamma_c^{m,n}$, que tem como finalidade enfatizar a transferência de aprendizagem entre pares de unidade de ativação. De forma a diminuir o custo computacional e não elevar significativamente a quantidade de parâmetros necessários a serem treinados, as redes totalmente conectadas da meta-rede h foram implementadas com limitação de apenas 9 saídas, que são transformadas em matrizes $3x3$. Essas matrizes são transformadas nos valores de $\gamma_c^{m,n}$ utilizando interpolações bilineares, de forma a igualar a dimensão dessas matrizes com a da expressão $\left(r_\theta\left(T_\theta^n(x)\right)_c - S^m(x)_c\right)^2$ na Equação 7, como ilustra a Figura 33.

Figura 33 - Processo utilizado para obter $\gamma_c^{m,n}$



Fonte: o autor (2022)

5.4 TREINAMENTO

Durante o treinamento, é necessário que sejam aprendidos os parâmetros θ, referentes ao modelo objetivo e ϕ, que representa os parâmetros das meta-redes. De forma a manter uma comparação justa, o mesmo procedimento de treino proposto anteriormente por Jang *et al.* (2019) foi mantido. Dessa forma, em cada passo, primeiro são atualizados os parâmetros θ, utilizando a função objetivo $L_{\text{total}}(\theta|x, y, \phi)$ e, em seguida, os parâmetros das meta-redes ϕ. Esse procedimento está descrito de forma detalhada no Algoritmo 1.

Algoritmo 1: Procedimento de treinamento

entradas: $\boldsymbol{D_{\text{train}} = \{(x_i, y_i)\}}$**,** ***taxa de aprendizagem*** $\boldsymbol{\alpha}$

repita

  amostre um *batch* $B \subset D_{\text{train}}$, com $|B| = b$

  atualize θ, de forma a minimizar $\frac{1}{b}\sum_{(x,y)\in B} L_{\text{total}}(\theta|x, y, \phi)$

  inicialize $\theta_0 \leftarrow \theta$

  para $\boldsymbol{t = 0}$ **ate** $\boldsymbol{T - 1}$ **faca**

    $$\theta_{t+1} \leftarrow \theta_t - \alpha\nabla_\theta \frac{1}{b} \sum_{(x,y)\in B} L_{\text{wfm}}(\theta_t|x, \phi)$$

  fimpara

  $$\theta_{T+1} \leftarrow \theta_T - \alpha\nabla_\theta \frac{1}{b} \sum_{(x,y)\in B} L_{\text{org}}(\theta_T|x, y)$$

  atualize ϕ utilizando $\nabla_\phi \frac{1}{b}\sum_{(x,y)\in B} L_{\text{org}}(\theta_{T+1}|x, y)$

ate *condicao_de_parada*

Fonte: adaptado de Jang *et al.* (2019)

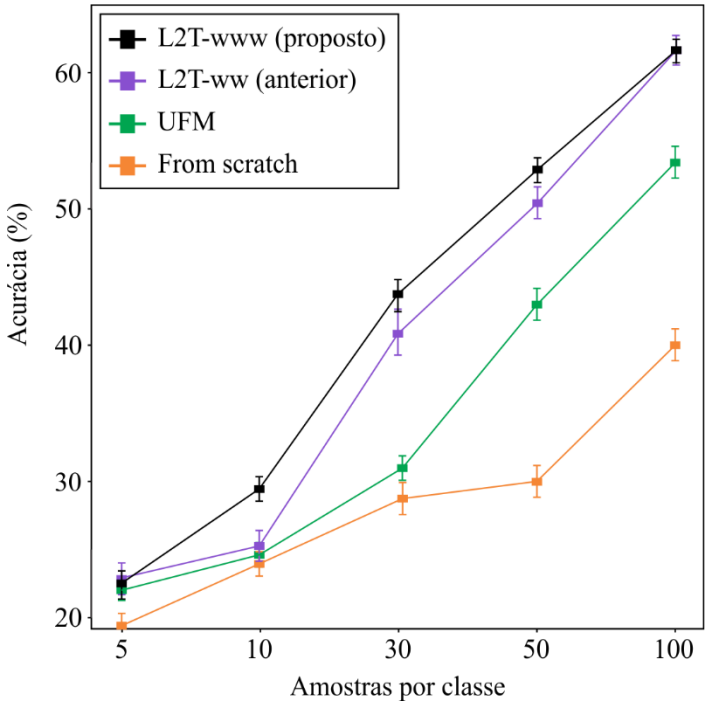
Em todos os experimentos foi utilizado o algoritmo de *Stochastic Gradient Descent* (*SGD*) (BISHOP e NASRABADI, 2006) com *momentum* de 0,9. para o treinamento da rede objetivo. A taxa de aprendizagem α foi inicializada com o valor de 0,1 e com decaimento utilizando *cosine annealing* (LOSHCHILOV e HUTTER, 2016): $\alpha_n = \frac{1}{2}\left(1 + \cos\left(\frac{n}{N}\pi\right)\right)$, onde N representa o número total de épocas e α_n representa a taxa de aprendizagem na época n. Quanto ao treinamento das meta-redes f_ϕ, g_ϕ e h_ϕ, foi utilizado $\beta = 0{,}5$, otimizador *ADAM* (KINGMA e BA, 2014) com taxa de aprendizagem 10^{-4}, taxa de decaimento de 10^{-4} e $T = 2$. Em todos experimentos foram definidos tamanho do *mini-batch* de 128 e número total de épocas $N = 200$. Esses hiperparâmetros foram definidos iguais ao dos experimentos dos estudos realizados por Jang *et al.* (2019), de forma a realizar uma comparação justa entre o método proposto (*L2T-www*) e o método anterior (*L2T-ww*).

5.5 COMPARATIVO DOS MÉTODOS

De forma a avaliar o máximo potencial de transferência de conhecimento prévio de um modelo fonte pré-treinado para o modelo objetivo, o método proposto (*L2T-www*) foi testado em um regime de dados de treino limitado. Nesse sentido, foi testada como tarefa a ser aprendida a correta classificação de imagens dos conjuntos de dados STL-10, CIFAR-10 e CIFAR-100. Para cada um desses conjuntos de dados, foram amostradas diferentes quantidades de dados para o treinamento do modelo, de forma a avaliar o desempenho do método para diferentes disponibilidades de dados de treino. Para efeitos de comparação, o mesmo procedimento foi repetido para o método anterior (*L2T-ww*), para o método proposto por Romero *et al.* (2014) (*Unweighted Feature Matching - UFM*) e para o treino do modelo fonte sem transferência de aprendizagem (*from scratch*). Em todos os métodos, os experimentos foram repetidos $n = 30$ vezes para cada tarefa, de forma a realizar análises estatísticas, nas quais utilizou-se *p-value* < 0,05 para considerar significância estatística. A comparação de desempenho entre o método proposto e o anterior foi realizada através de testes U de *Mann-Whitney* (MANN e WHITNEY, 1947), que demonstraram como evidência a superioridade do método proposto em

certas faixas de quantidades de dados de treino nos três conjuntos de dados, como ilustra a Figura 34 para o STL-10.

Figura 34 - Experimento no *dataset* STL-10



Fonte: o autor (2022)

Como é possível observar, o método proposto (*L2T-www*) supera o método anterior (*L2T-ww*) em uma faixa central de quantidade de amostras de treino. O mesmo comportamento se mantém em outros conjuntos de dados testados, como detalhado nas Tabelas 1, 2 e 3. Esses resultados demonstram evidência da superioridade do método proposto em um regime limitado de dados de treino (por exemplo, de 10 a 50 amostras por classe no STL-10), o qual é um cenário comum em problemas que necessitam de transferência de aprendizagem.

Tabela 1 - Experimentos no STL-10

Amostras por classe	5	10	30	50	100
From scratch	$19{,}70_{\pm 1{,}05}$	$24{,}07_{\pm 0{,}91}$	$28{,}80_{\pm 0{,}98}$	$30{,}11_{\pm 1{,}09}$	$40{,}12_{\pm 0{,}93}$
UFM	$22{,}45_{\pm 0{,}74}$	$24{,}76_{\pm 1{,}03}$	$30{,}94_{\pm 1{,}12}$	$43{,}21_{\pm 0{,}97}$	$53{,}41_{\pm 1{,}01}$
L2T-ww (anterior)	$22{,}94_{\pm 1{,}18}$	$25{,}64_{\pm 1{,}11}$	$41{,}08_{\pm 1{,}63}$	$50{,}49_{\pm 1{,}02}$	$61{,}44_{\pm 1{,}07}$
L2T-www (proposto)	$22{,}54_{\pm 1{,}01}$	$29{,}39_{\pm 0{,}84}$	$43{,}38_{\pm 1{,}04}$	$52{,}53_{\pm 0{,}80}$	$61{,}36_{\pm 0{,}84}$
Diferença	-0,4	**+3,75**	**+2,30**	**+2,04**	-0,08
p-value	$p > 0{,}05$	$\boldsymbol{p = 0{,}000}$	$\boldsymbol{p = 0{,}044}$	$\boldsymbol{p = 0{,}004}$	$p > 0{,}05$

Fonte: o autor (2022)

Tabela 2 - Experimentos no CIFAR-10

Amostras por classe	10	30	50	100	150
From scratch	$20{,}52_{\pm 1{,}63}$	$22{,}55_{\pm 1{,}10}$	$31{,}14_{\pm 0{,}97}$	$43{,}21_{\pm 0{,}69}$	$51{,}17_{\pm 0{,}71}$
UFM	$20{,}69_{\pm 1{,}35}$	$39{,}61_{\pm 1{,}07}$	$48{,}45_{\pm 0{,}99}$	$57{,}29_{\pm 0{,}81}$	$68{,}26_{\pm 0{,}59}$
L2T-ww (anterior)	$24{,}64_{\pm 1{,}85}$	$43{,}16_{\pm 1{,}16}$	$52{,}91_{\pm 1{,}08}$	$63{,}86_{\pm 0{,}79}$	$71{,}88_{\pm 0{,}63}$
L2T-www (proposto)	$26{,}12_{\pm 1{,}99}$	$45{,}18_{\pm 0{,}98}$	$54{,}76_{\pm 0{,}75}$	$64{,}91_{\pm 0{,}62}$	$71{,}29_{\pm 1{,}36}$
Diferença	+1,48	**+2,02**	**+1,85**	**+1,05**	-0,59
p-value	$p > 0{,}05$	$\boldsymbol{p = 0{,}005}$	$\boldsymbol{p = 0{,}006}$	$\boldsymbol{p = 0{,}041}$	$p > 0{,}05$

Fonte: o autor (2022)

Tabela 3 - Experimentos no CIFAR-100

Amostras por classe	5	10	20	30	50
From scratch	$9{,}77_{\pm 0{,}55}$	$10{,}96_{\pm 0{,}59}$	$18{,}08_{\pm 0{,}41}$	$21{,}86_{\pm 0{,}45}$	$29{,}65_{\pm 0{,}48}$
UFM	$12{,}18_{\pm 0{,}49}$	$18{,}94_{\pm 0{,}78}$	$30{,}77_{\pm 0{,}59}$	$37{,}68_{\pm 0{,}77}$	$44{,}47_{\pm 0{,}54}$
L2T-ww (anterior)	$14{,}12_{\pm 0{,}59}$	$24{,}37_{\pm 0{,}81}$	$37{,}35_{\pm 0{,}37}$	$42{,}61_{\pm 0{,}55}$	$49{,}67_{\pm 0{,}78}$
L2T-www (proposto)	$14{,}22_{\pm 0{,}61}$	$25{,}53_{\pm 0{,}46}$	$36{,}91_{\pm 0{,}74}$	$42{,}24_{\pm 0{,}38}$	$49{,}03_{\pm 0{,}59}$
Diferença	+0,1	**+1,16**	-0,44	-0,37	-0,64
p-value	$p > 0{,}05$	$\boldsymbol{p = 0{,}008}$	$p > 0{,}05$	$p > 0{,}05$	$p > 0{,}05$

Fonte: o autor (2022)

No entanto, é importante mencionar que fora das faixas centrais de quantidade de dados de treino, o método proposto não se demonstrou superior ao método anterior para a quantidade de dados testados, com *p-values* > 0,05.

Em relação às regiões acima das faixas centrais mencionadas, ou seja, com uma quantidade considerável de dados, uma possibilidade é de que uma grande quantidade de dados seja suficiente para proporcionar conhecimento ao modelo objetivo em uma abundância que torna a melhoria da adição de uma nova meta-rede h insignificante. Assim, as meta-redes f e g seriam suficientes para transferir o conhecimento nesse cenário de grande quantidade de dados de treino.

Quanto às regiões abaixo das faixas centrais, ou seja, com uma quantidade de dados extremamente limitada, é possível observar uma grande variância em ambos os métodos, não resultando em diferenças estatisticamente significantes. Uma possibilidade é que nesse regime de extrema limitação de dados, a melhoria da transferência de conhecimento proporcionada pela adição da meta-rede $h$seja quase insignificante, em relação à influência da aleatoriedade da inicialização dos pesos das redes.

Apesar das limitações mencionadas, é importante ressaltar novamente que o método proposto se mostrou superior em uma faixa de quantidade de dados intermediários, em relação ao método anterior.

5.6 CLASSES DESBALANCEADAS

Com o intuito de avaliar o potencial do método em dados desbalanceados, nos quais uma das classes apresenta poucas amostras foram realizados experimentos com uma das classes desbalanceada. Para isso, foi utilizado o conjunto de dados STL-10 e as mesmas configurações de treino adotadas para os outros experimentos. Nesse sentido, foram realizados os treinamentos com a seguinte modificação: nove classes com 100 amostras e uma das classes com 10 amostras apenas. Esses experimentos foram realizados utilizando o método proposto (L2T-www), o método anterior (L2T-ww) e sem transferência de aprendizagem (*From scratch*). Assim, as acurácias obtidas para a classe desbalanceada em cada experimento estão reportados na Tabela 4.

Tabela 4 - Acurácias da classe desbalanceada

Técnica	Acurácia da classe desbalanceada
L2T-www (proposto)	$\mathbf{73,56_{\pm 1,03}}$
L2T-ww (anterior)	$65{,}12_{\pm 0{,}94}$
From scratch (Sem transferência de aprendizagem)	$16{,}12_{\pm 0{,}78}$

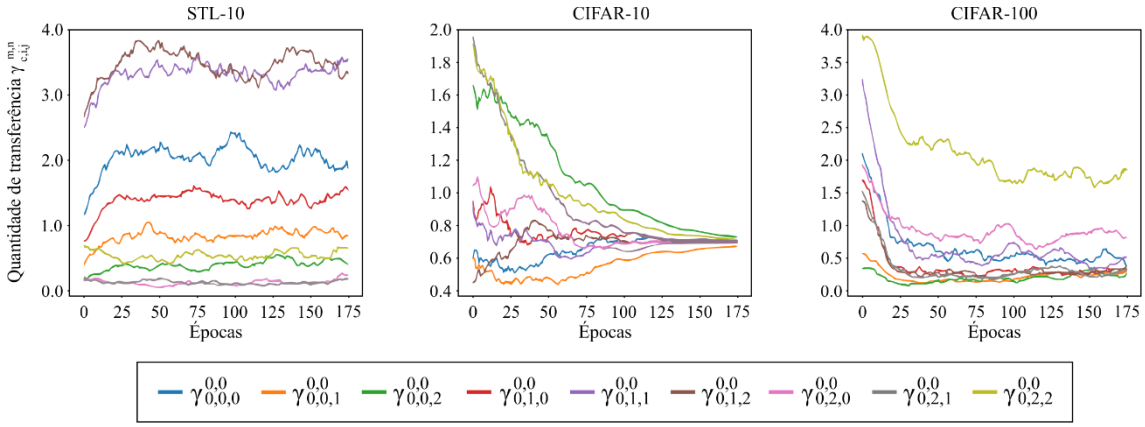
Fonte: o autor (2022)

Como é possível observar, o método proposto apresenta uma vantagem significante em relação ao método anterior, indicando um melhor aproveitamento da transferência de aprendizagem durante o treinamento.

5.7 CONVERGÊNCIA DOS PESOS

De forma a demonstrar outras evidências de utilidade do método proposto, também foram analisados alguns dos valores de $\gamma_{c,i,j}^{m,n}$ ao longo do treinamento, gerados pela adição da meta-rede h, Figura 34. Como é possível observar, os valores convergem ao longo do processo de treinamento, evidenciando que a nova meta-rede h está aprendendo a transferir conhecimento no nível de *pixels*.

Figura 35 - Ilustração dos valores de $\gamma_{c,i,j}^{m,n}$ ao longo do treino.

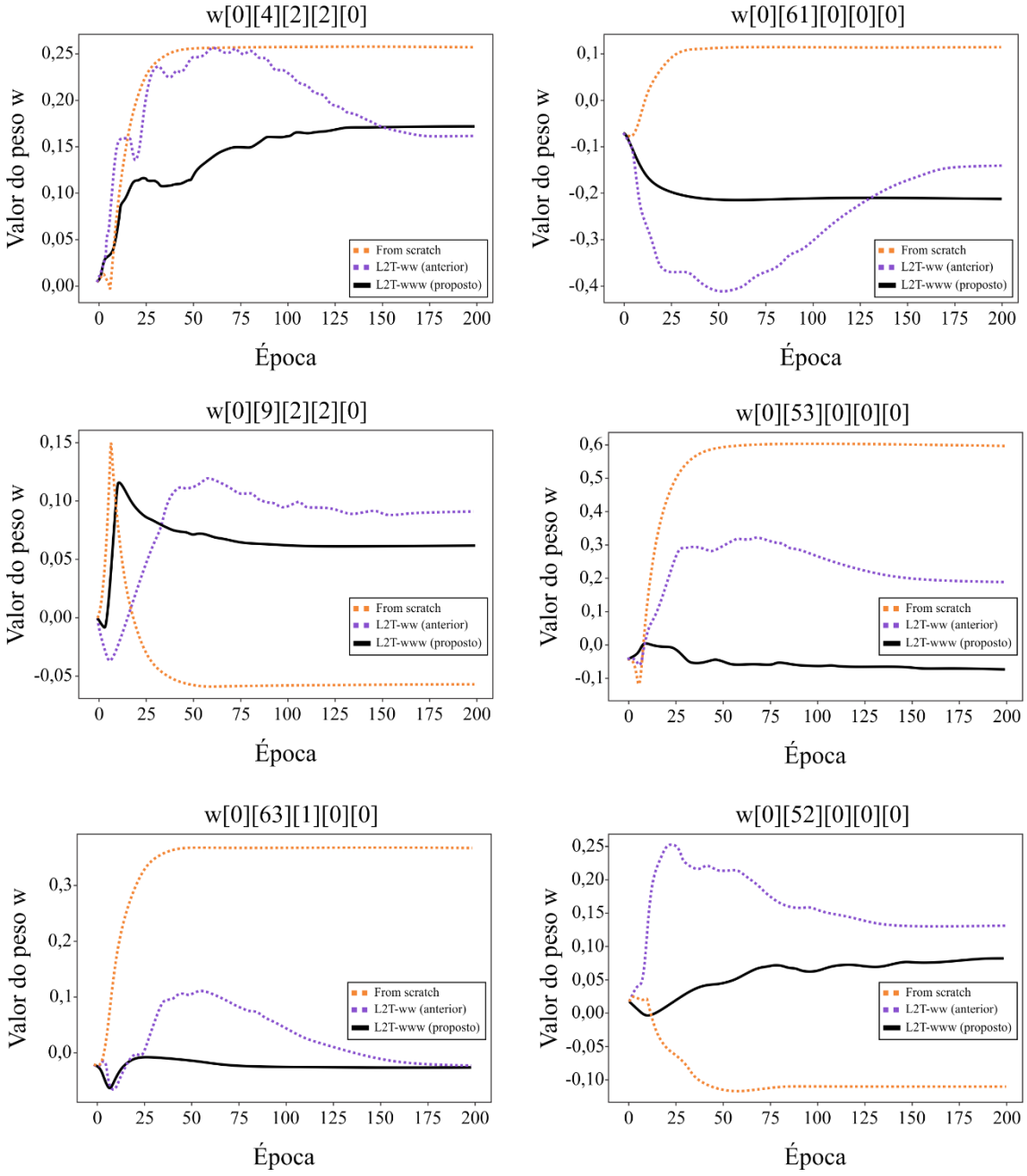


Fonte: o autor (2022)

5.8 ESTUDO DE ABLAÇÃO

De forma a auxiliar o entendimento da contribuição do método proposto para o aprendizado da rede, foram realizados estudos de ablação. Para isso, foram realizados treinamentos em três configurações diferentes: *from scratch* (sem transferência de aprendizagem), *L2T-ww* (método anterior) e *L2T-www* (método proposto). Nos experimentos dessas três configurações, a rede objetivo foi inicializada com os mesmos pesos e a aleatoriedade sincronizada utilizando o mesmo *seed*. Dessa forma, foi analisada a evolução dos pesos de determinados neurônios durante o treinamento, em cada uma das três configurações. Como é possível observar, os pesos dos neurônios utilizando o método proposto (L2T-www) têm uma tendencia a convergir mais rápido, se comparado com as outras duas abordagens, Figura 36. Essa rápida convergência indica uma das causas da superioridade do método proposto em alcançar resultados superiores às outras abordagens.

Figura 36 - Estudo de ablação



Fonte: o autor (2022)

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, foi proposta a técnica *Learning to Transfer What, Where and Which* (*L2T-www*), que é um novo método para transferência de aprendizagem entre redes convolucionais de arquiteturas diferentes. Nesse sentido, foram demonstradas evidências empíricas de que esse novo método tem desempenho superior ao método anterior, proposto por Jang *et al*. (2019), em cenários de quantidades limitadas de dados de treino. Para isso, foram realizados experimentos para avaliar o desempenho de ambos os métodos na classificação dos conjuntos de dados CIFAR-10, CIFAR-100 e STL-10, comparando-se os desempenhos através de análises estatísticas. Nesse sentido, foram utilizados testes de hipóteses de Mann-Whitney, que permitiram ratificar, com *p-values* < 0,05, o desempenho superior do método proposto em relação a outras abordagens, como as propostas por Romero *et al.* (2014) e Jang *et al.* (2019).

Como trabalho futuro, é possível adaptar o método proposto para diferentes arquiteturas de modelos, além das CNNs. Por exemplo, é possível desenvolver um método para transferência de aprendizagem entre diferentes arquiteturas de *transformers* (VASWANI *et al*., 2017). Também é possível avaliar o desempenho do método na transferência de aprendizagem entre diferentes tarefas, como por exemplo, rede fonte pré-treinada na classificação de imagens e rede objetivo com a tarefa de detecção de objetos. Por fim, é possível concluir que o método proposto contribui significativamente para o campo de estudo dos métodos de transferência de aprendizagem, podendo inspirar novos estudos e grandes descobertas.

REFERÊNCIAS

BACH, Sebastian et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. **PloS one**, v. 10, n. 7, p. e0130140, 2015.

BAU, David et al. Understanding the role of individual units in a deep neural network. **Proceedings of the National Academy of Sciences**, v. 117, n. 48, p. 30071-30078, 2020.

BENGIO, Yoshua; SIMARD, Patrice; FRASCONI, Paolo. Learning long-term dependencies with gradient descent is difficult. **IEEE transactions on neural networks**, v. 5, n. 2, p. 157-166, 1994.

BISHOP, Christopher M.; NASRABADI, Nasser M. **Pattern recognition and machine learning**. New York: springer, 2006.

COATES, Adam; NG, Andrew; LEE, Honglak. An analysis of single-layer networks in unsupervised feature learning. In: **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. JMLR Workshop and Conference Proceedings, 2011. p. 215-223.

DENG, Jia et al. Imagenet: A large-scale hierarchical image database. In: **2009 IEEE conference on computer vision and pattern recognition**. Ieee, 2009. p. 248-255.

DENG, Lu et al. Region-based CNN method with deformable modules for visually classifying concrete cracks. **Applied sciences**, v. 10, n. 7, p. 2528, 2020.

FERGUSON, Max et al. Automatic localization of casting defects with convolutional neural networks. In: **2017 IEEE international conference on big data (big data)**. IEEE, 2017. p. 1726-1735.

GHOSH, Anirudha et al. Fundamental concepts of convolutional neural network. In: **Recent trends and advances in artificial intelligence and Internet of Things**. Springer, Cham, 2020. p. 519-567.

GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua. Deep sparse rectifier neural networks. In: **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. JMLR Workshop and Conference Proceedings, 2011. p. 315-323.

GUO, Yunhui et al. Spottune: transfer learning through adaptive fine-tuning. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. 2019. p. 4805-4814.

HAYKIN, Simon. **Neural networks and learning machines, 3/E**. Pearson Education India, 2009.

HE, Kaiming et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 770-778.

JANG, Yunhun et al. Learning what and where to transfer. In: **International Conference on Machine Learning**. PMLR, 2019. p. 3030-3039.

KIM, Jangho; PARK, SeongUk; KWAK, Nojun. Paraphrasing complex network: Network compression via factor transfer. **Advances in neural information processing systems**, v. 31, 2018.

KINGMA, Diederik P.; BA, Jimmy. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.
KRIZHEVSKY, Alex et al. Learning multiple layers of features from tiny images. 2009.

KRIZHEVSKY, Alex; HINTON, Geoff. Convolutional deep belief networks on cifar-10. **Unpublished manuscript**, v. 40, n. 7, p. 1-9, 2010.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012.

LE, Ya; YANG, Xuan. Tiny imagenet visual recognition challenge. **CS 231N**, v. 7, n. 7, p. 3, 2015.

LECUN, Yann et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278-2324, 1998.

LOSHCHILOV, Ilya; HUTTER, Frank. Sgdr: Stochastic gradient descent with warm restarts. **arXiv preprint arXiv:1608.03983**, 2016.

LU, Lu et al. Dying relu and initialization: Theory and numerical examples. **arXiv preprint arXiv:1903.06733**, 2019.

MAAS, Andrew L. et al. Rectifier nonlinearities improve neural network acoustic models. In: **Proc. icml**. 2013. p. 3.

MANN, Henry B.; WHITNEY, Donald R. On a test of whether one of two random variables is stochastically larger than the other. **The annals of mathematical statistics**, p. 50-60, 1947.

MCCARTHY, John. What is artificial intelligence. **URL: http://www-formal. stanford. edu/jmc/whatisai. html**, 2004.

MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115-133, 1943.

MITCHELL, Tom M.; MITCHELL, Tom M. **Machine learning**. New York: McGraw-hill, 1997.

PHANG, Jason; FÉVRY, Thibault; BOWMAN, Samuel R. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. **arXiv preprint arXiv:1811.01088**, 2018.

PRUKSACHATKUN, Yada et al. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?. **arXiv preprint arXiv:2005.00628**, 2020.

ROMERO, Adriana et al. Fitnets: Hints for thin deep nets. **arXiv preprint arXiv:1412.6550**, 2014.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65, n. 6, p. 386, 1958.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **nature**, v. 323, n. 6088, p. 533-536, 1986.

SHARIF RAZAVIAN, Ali *et al*. CNN features off-the-shelf: an astounding baseline for recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition workshops**. 2014. p. 806-813.

SHARIF RAZAVIAN, Ali et al. CNN features off-the-shelf: an astounding baseline for recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition workshops**. 2014. p. 806-813.
SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

TORRALBA, Antonio; FERGUS, Rob; FREEMAN, William T. 80 million tiny images: A large data set for nonparametric object and scene recognition. **IEEE transactions on pattern analysis and machine intelligence**, v. 30, n. 11, p. 1958-1970, 2008.

VASWANI, Ashish et al. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

WANG, Alex et al. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. **arXiv preprint arXiv:1812.10860**, 2018.

ZHENG, Hao et al. Improving deep neural networks using softplus units. In: **2015 International joint conference on neural networks (IJCNN)**. IEEE, 2015. p. 1-4.

ZHOU, Bolei et al. Revisiting the importance of individual units in cnns via ablation. **arXiv preprint arXiv:1806.02891**, 2018.