Apesar das redes neurais artificiais auxiliarem na tarefa de reconhecimento de padrões há algumas décadas, as Redes Neurais Convolucionais sofrem ainda com a grande necessidade de poder computacional, restringindo muitas vezes a sua utilização. Conhecendo a necessidade de se otimizar os parâmetros de uma rede, foi proposto um método de otimização feito para a DenseNet, uma rede neural convolucional que tem como característica ser completamente conectada. Para isto, foi proposto o controle da geração dos mapas de características em relação ao momento em que a rede se encontra, objetivando a redução do tamanho da rede com o mínimo de perda na acurácia. Este controle se dá quando a rede é criada, reduzindo de maneira progressiva o número de mapas de característica. Foram realizados experimentos em bases de imagens de diferentes características: MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, CALTECH-101, Cats vs Dogs e TinyImageNet. Os melhores resultados de cada conjunto foram: para a base MNIST, houve uma redução de parâmetros de 43%, igualmente para a base Fashion-MNIST. Para a base CIFAR-10 alcançou-se uma redução de 44% nos parâmetros da rede, na base CIFAR-100 se obteve uma diferença de 43% de parâmetros a menos. Na base CALTECH-101 a otimização de parâmetros foi de 35%, enquanto a base Cats vs Dogs otimizou 30% dos parâmetros dos modelos. Por fim, a base TinyImageNet reduziu 3,85% dos parâmetros.

Orientador: Dr. André Tavares da Silva

Joinville, 2019

ANO
2019

CRISTIANO R. SIEBERT | OTIMIZANDO PARÂMETROS DE UMA DENSENET ATRAVÉS DO CONTROLE DE GERAÇÃO DE MAPAS DE CARACTERÍSTICAS

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS GRADUAÇÃO EM COMPUTAÇÃO APLICADA

DISSERTAÇÃO DE MESTRADO

OTIMIZANDO PARÂMETROS DE UMA DENSENET ATRAVÉS DO CONTROLE DE GERAÇÃO DE MAPAS DE CARACTERÍSTICAS

CRISTIANO ROBERTO SIEBERT

JOINVILLE, 2019

UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

CRISTIANO ROBERTO SIEBERT

OTIMIZANDO PARÂMETROS DE UMA DENSENET ATRAVÉS DO CONTROLE DE GERAÇÃO DE MAPAS DE CARACTERÍSTICAS

JOINVILLE

2019

CRISTIANO ROBERTO SIEBERT

OTIMIZANDO PARÂMETROS DE UMA DENSENET ATRAVÉS DO CONTROLE DE GERAÇÃO DE MAPAS DE CARACTERÍSTICAS

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Dr. André Tavares da Silva

JOINVILLE

2019

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

com os dados fornecidos pelo(a) autor(a)

Siebert, Cristiano Roberto
Otimizando parâmetros de uma DenseNet através do controle de geração de mapas de características / Cristiano Roberto Siebert. -- 2020.
76 p.

Orientador: André Tavares da Silva
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2020.

1. Otimização. 2. DenseNet. 3. Redes Neurais Convolucionais. I. Tavares da Silva, André. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. III. Título.

Otimizando Parâmetros de uma Densenet Através do Controle de Geração de Mapas de Características

por

Cristiano Roberto Siebert

Esta dissertação foi julgada adequada para obtenção do título de

Mestre em Computação Aplicada

Área de concentração em "Ciência da Computação",
e aprovada em sua forma final pelo

CURSO DE MESTRADO ACADÊMICO EM COMPUTAÇÃO APLICADA
DO CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:

Prof. Dr. André Tavares da Silva
CCT/UDESC (Orientador/Presidente)

Prof. Dr. Paulo Ricardo Lisboa de Almeida
CCT/UDESC

Prof. Dr. Andre Gustavo Hochuli
PUC/PR

Joinville,SC, 02 de dezembro de 2019.

Dedico este trabalho à minha família, aos meus familiares e aos professores que me acompanharam e me instruíram por todo o caminho, este é o resultado do vosso apoio.

AGRADECIMENTOS

Mais do que justo o meu primeiro agradecimento ser direcionado ao Autor da Vida, o pai do conhecimento e da ciência que nos deu capacidade de aprender e nos desenvolver, que me protegeu por todo o tempo em que saí e percorri este caminho quase diário de Balneário Camboriú à Joinville, à Deus, o meu mais sincero agradecimento.

Minha esposa e minha filha, Gabriela e Lyanna, são meus maiores tesouros, a razão pelo qual me esforço para me tornar uma pessoa melhor todos os dias, todas as madrugadas acordadas comigo e o tempo de lazer utilizado para estudo eram apenas temporários, porém minha gratidão pelo seu apoio em todos os momentos, fáceis ou difíceis, será eterna.

Meus pais podem não ter me trazido ao mundo, mas me escolheram e me ensinaram a viver, formaram meu caráter e sua contribuição na minha vida é inigualável, a vocês o meu muito obrigado.

Mesmo quando eu chegava nos eventos da família com a mochila, e estudava enquanto todos se divertiam, todos os meus familiares me compreenderam e me apoiaram, este tipo de apoio é de grande importância e por isso expresso a minha gratidão.

Ao meu orientador, Dr.André, um agradecimento especial, pois ele acreditou em mim e me deu a chance de crescer como pessoa e como profissional durante este tempo, seus conselhos foram extremamente úteis, muito obrigado.

Estendo meu agradecimento à todos os professores que me acompanharam nesta trajetória e compartilharam seus conhecimentos comigo neste caminho, todos vocês são pessoas admiráveis, minha sincera gratidão.

“Se eu vi mais longe, foi por estar sobre ombros de gigantes.”

Issac Newton

RESUMO

Apesar das redes neurais artificiais auxiliarem na tarefa de reconhecimento de padrões há algumas décadas, as Redes Neurais Convolucionais sofrem ainda com a grande necessidade de poder computacional, restringindo muitas vezes a sua utilização em plataformas variadas. Conhecendo a necessidade de se otimizar os parâmetros de uma rede e no que isto implica, foi proposto um método de otimização feito para a DenseNet, uma rede neural convolucional que tem como característica ser completamente conectada, inclusive nas camadas convolucionais. Para isto, foi proposto o controle da geração dos mapas de características em relação ao momento em que a rede se encontra, objetivando a redução do tamanho da rede com o mínimo de perda na acurácia. Este controle se dá no momento em que a rede é criada, reduzindo de maneira progressiva o número de mapas de característica. O que torna este controle possível é a adição de um novo parâmetro chamado de controle de decréscimo (*Decrease Control*) ou valor dc, que é definido pela profundidade da rede e pelo valor k que é o fator de crescimento dos mapas de características. O decréscimo se dá a partir da metade das camadas, baseado na orientação do autor em iniciar a redução da taxa de aprendizado a partir da metade do treinamento da rede. Gerenciar a criação dos mapas de característica permite que um modelo que já possui em sua natureza um número de parâmetros reduzido possa ficar ainda mais compacto mantendo de forma aproximada, por vezes igual, os valores de acurácia nos experimentos realizados. A fim de validar o comportamento do modelo proposto, foram realizados experimentos em bases de imagens de diferentes características: MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, CALTECH-101, Cats vs Dogs e TinyImageNet. Os melhores resultados de cada conjunto foram: para a base MNIST, com a configuração $k = 12, d = 100$ a diferença de acurácia foi de 0,08%, com uma aceleração de 14% e uma redução de parâmetros de 43%, já para a base Fashion-MNIST a redução de parâmetros foi de 43%, com uma aceleração de 14% e uma diferença na acurácia de 0,62%. Para a base CIFAR-10, a configuração $k = 12, d = 100$ com uma diferença de acurácia de 1,1% com uma redução de 44% nos parâmetros da rede, além de uma melhoria de 23% no tempo de treino, na base CIFAR-100 a melhor configuração foi a $k = 24, d = 100$, com 77,15% de acurácia, uma diferença de 3,55% e 43% de parâmetros a menos, com 37% de redução do tempo. Na base CALTECH-101 a otimização de parâmetros foi de 35%, com uma aceleração de 11% com acurácia de 99,21%, enquanto a base Cats vs Dogs obteve uma diferença de 1,55% na acurácia, com uma aceleração de 13% e otimizou 30% dos parâmetros dos modelos. Por fim, a diferença de acurácia na base TinyImageNet foi de 3,85%, com uma redução de 31% dos parâmetros.

Palavras-chaves: Rede Neural Convolucional, Otimização, DenseNet.

ABSTRACT

Although artificial neural networks have helped in the pattern recognition task for a few decades, Convolutional Neural Networks still suffer from a great need for computational power, often restricting their use on various platforms. Knowing the need to optimize the parameters of a network and what it implies, it was proposed an optimization method made for DenseNet, a convolutional neural network that has the characteristic of being completely connected, including in the convolutional layers. For this, it was proposed to control the generation of the characteristic maps in relation to the moment it is in the network, aiming to reduce the network size with the minimum loss in accuracy. This control occurs at the moment the network is created, progressively reducing the number of feature maps. What makes this control possible is the addition of a new parameter called the Decrease Control or dc value, which is defined by the depth of the grid and the value k that is the growth factor of the feature maps. The decrease occurs from half of the layers, based on the author's orientation to start reducing the learning rate from half of the network training. Managing the creation of feature maps allows a model that already has a reduced number of parameters in its nature to become even more compact while maintaining approximate accuracy values in the experiments performed. In order to validate the behavior of the proposed model, experiments were performed using different image bases: MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, CALTECH-101, Cats vs Dogs and TinyImageNet. The best results from each set were: for the MNIST base, with the setting $k = 12, d = 100$ the accuracy difference was 0.08%, with an acceleration of 14% and a parameter reduction of 43%, already for the Fashion-MNIST base the parameter reduction was 43%, with an acceleration of 14% and a difference in accuracy of 0.62%. For the CIFAR-10 base, the setting $k = 12, d = 100$ with a 1.1% accuracy difference with a 44% reduction in network parameters, plus a 23% improvement in training time in base CIFAR-100 the best setting was a $k = 24, d = 100$, with 77.15% accuracy, a difference of 3.55% and 43% fewer parameters, with a 37% reduction in time. In the CALTECH-101 base the parameter optimization was 35%, with an 11% acceleration with 99.21% accuracy, while the Cats vs Dogs base obtained a 1.55% difference in accuracy, with an acceleration of 13%. and optimized 30% of model parameters. Finally, the accuracy difference in the TinyImageNet base was 3.85%, with a 31% reduction of the parameters.

Key-words: Convolutional Neural Network, Optimization, DenseNet.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BN	*Batch Normalization*
CNN	*Convolutional Neural Networks*
CPU	*Central Processing Unit*
FV	*Feature Vector*
GPU	*Graphics Processing Unit*
ILSVRC	*Imagenet Large Scale Visual Recognition Challenge*
MLP	*MultiLayer Perceptron*
ReLU	*Rectified Linear Unit*
RNA	Redes Neurais Artificiais

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, as redes neurais se tornaram alvo de um número expressivo de pesquisas, com as mais variadas aplicações e alcançando altos níveis de acurácia. É sabido que redes profundas possuem milhões ou até mesmo bilhões de parâmetros, como no trabalho de Dean et al. (2012), com 1,7 bilhões de parâmetros.

A AlexNet de Krizhevski, Sutskever e Hinton (2012) foi considerada o estado-da-arte em 2012 quando atingiu bons resultados no conjunto de dados ImageNet, utilizando uma rede de 60 milhões de parâmetros com apenas cinco camadas convolucionais e três camadas totalmente conectadas. Atualmente, algumas redes podem chegar a possuir centenas de milhões de parâmetros, e isso pode trazer resultados significativos nos índices de acurácia, porém tem como efeito contrário o aumento do tempo de treinamento e da necessidade de maior poder computacional.

Além disso, os recentes progressos em realidade virtual e aumentada e os dispositivos vestíveis e inteligentes trouxeram uma oportunidade de enfrentar desafios fundamentais no desenvolvimento de redes profundas para dispositivos com recursos limitados, sejam eles memória, CPU, energia ou largura de banda.

Métodos eficientes de otimização tem impactos significativos em todas estas áreas, por exemplo, a uma rede ResNet (HE et al., 2016) com 50 camadas consome cerca de 95MB de memória de armazenamento, porém descartando alguns pesos redundantes, esta rede permanece íntegra e com resultados semelhantes, apesar da redução de até 75%.

A importância da otimização de redes neurais não está apenas na possibilidade dos dispositivos portáteis, embarcados ou vestíveis utilizarem redes treinadas para propósitos específicos, mas também permite que todo o processo de treinamento seja acelerado e demande menos recursos do que outras redes que possuam o mesmo propósito.

Objetivando ter uma menor quantidade de parâmetros sem comprometer os índices de acurácia, Huang et al. (2017) criou a DenseNet, uma rede neural convolucional utilizada para classificação que traz o conceito de camadas densas em todas as camadas da rede, inclusive as convolucionais, compartilhando entre suas camadas os mapas de características e assim diminuindo de maneira significativa a quantidade de parâmetros da rede.

Esta rede possui alguns diferenciais, como a redução de parâmetros, o fortalecimento da propagação das suas características, uma vez que o compartilhamento

das mesmas entre as camadas impede que a rede gere e aprenda mapas redundantes e a redução do *Gradient Vanishing*, um problema que ocorre com redes profundas quando os gradientes da função de perda começam a se aproximar de zero, deixando a rede menos sensível as atualizações geradas na retropropagação do erro durante o treinamento. Sua estrutura densa permite que um número menor, porém fixo de mapas de características sejam gerados a cada camada, sendo este número denominado k, sendo que estes mapas são concatenados à camada subsequente, em algo denominado pelo autor de "conhecimento coletivo".

Este acúmulo de mapas de características gerado pelo valor fixo k traz uma série de novas e profundas características conforme a execução da rede, porém é possível observar uma redução na capacidade de aprendizado da rede a partir de determinado ponto do treinamento, neste caso uma das atitude necessárias para combater este problema é a redução da taxa de aprendizagem conforme a proximidade dos valores de perda se aproxima de zero, isso traz a hipótese de que a geração fixa de novos mapas de características em determinados pontos da rede pouco influencia no seu aprendizado, uma vez que a mesma está menos sensível a mudanças e portanto alguns destes mapas poderiam ser descartados sem prejudicar de maneira significativa a acurácia da rede, e ainda ter um efeito otimizador através desta prática.

Pode-se verificar um número expressivo de trabalhos publicados para otimização de redes neurais devido à sua necessidade de poder computacional para sua utilização. Diversos algoritmos para este fim têm sido propostos, sendo que algumas destas soluções serão apresentados e discutidos nesse trabalho, divididos entre os temas decomposição de camada, redes de poda, projeção bloco-circulante e *Knowledge Distillation*.

Apesar da existência de uma série de soluções voltadas para a aceleração de redes neurais, todas as soluções encontradas têm uma característica em comum, que é a alteração ou redução de uma rede previamente existente a fim de torná-la mais simples. Muitas vezes isso implica na necessidade de retreinar a rede, gerar uma segunda rede a partir da primeira ou transformar os valores de uma rede treinada e verificar os seus efeitos, ou seja, sempre há necessidade de uma primeira execução para que a compactação da rede possa acontecer.

Diferente das demais soluções, este projeto de mestrado criou uma nova abordagem para criação de uma rede neural convolucional já com parâmetros reduzidos. É uma técnica de otimização para a DenseNet que permite definir uma menor quantidade de parâmetros com o mínimo de impacto em sua acurácia. Esta técnica envolve o controle da geração dos mapas de características, trazendo como contribuição o decréscimo sistemático da quantidade de mapas a cada camada a partir de um determinado momento da rede, através de uma nova variável denominada dc (*Decrease*

Control). Os resultados demonstram que a nova abordagem consegue gerar uma rede com redução de 31% a 46% no tamanho da rede em relação ao método original nas bases testadas e com redução de 4% a 39% no tempo de treinamento. A redução no espaço de armazenamento foi na ordem de 28% a 39%.

1.1 OBJETIVOS

Este projeto traz como objetivo geral a criação de uma técnica para otimização da rede DenseNet, uma rede neural convolucional que possui como principal característica sua arquitetura totalmente conectada, e que por isso, consegue compartilhar entre suas camadas as características geradas durante sua execução. A solução propõe o controle sistematizado da geração de mapas de característica para cada camada a partir de um momento determinado da estrutura da rede.

Também se destacam os seguintes objetivos específicos:

- Pesquisar e conhecer as técnicas existentes de otimização de redes neurais e suas aplicações, bem como sua categorização e sub-divisões;
- Manter proximidade ou equidade dos índices de acurácia dos modelos utilizados nos experimentos feitos na versão original da DenseNet;
- Investigar os impactos da nova técnica em outros fatores além da acurácia, como o tempo de treinamento e tamanho de armazenamento do modelo.

1.2 ESCOPO

Para definir o escopo desta pesquisa, a rede a ser otimizada será a DenseNet, sendo que a técnica não abrange até o momento outros tipos de arquiteturas. A tarefa escolhida para os experimentos comparativos da rede será a classificação de imagens.

Os conjuntos de dados escolhidos para os experimentos são: MNIST, Fashion-MINIST, Cats vs Dogs, CALTECH-101, CIFAR-10, CIFAR-100 e TinyImageNet, escolhidos pela utilização no artigo original e pela disponibilidade de acesso.

1.3 METODOLOGIA

Primeiramente foi realizada uma pesquisa exploratória com o intuito de conhecer o estado-da-arte em otimização de parâmetros para redes neurais convolucionais, além de conhecer as técnicas relacionadas a este tema. Em seguida, o modelo utilizado neste projeto foi projetado.

O desenvolvimento do modelo se deu em três etapas: a primeira consistiu em criar o modelo matemático necessário para que a otimização fosse possível, este modelo possui uma série de operações necessárias para encontrar o ponto de partida da otimização e qual o fator de decréscimo utilizado na rede. Em seguida foram feitos testes comparativos, criando redes comuns e otimizadas e comparando sua quantidade de parâmetros, validando o novo modelo.

A partir daí, treinamentos utilizando algumas das configurações previstas no artigo original foram realizados, e após o término destes treinamentos e da avaliação de cada modelo no seu respectivo conjunto de testes, foram feitas as tabulações e geração de gráficos dos resultados da rede.

1.4 ESTRUTURA

Esta dissertação está organizada da seguinte maneira: No Capítulo 2 é apresentada a Fundamentação Teórica que detalha os conceito necessários para entendimento da proposta. No Capítulo 3 são descritos trabalhos relacionados com o tema, ou seja, trabalhos que também trazem soluções para otimização de redes neurais. No Capítulo 4 está a descrição do desenvolvimento da solução proposta e as particularidades dos experimentos a serem realizados. No Capítulo 5, os experimentos realizados e seus resultados são apresentados, comparando-os com as versões originais. Por fim, o Capítulo 6 apresenta a conclusão do projeto e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão expostos os conceitos necessários para compreensão dos termos, técnicas e modelos utilizados na pesquisa e no desenvolvimento deste trabalho. Na Seção 2.1 são abordados conteúdos sobre reconhecimento de padrões como conceitos básicos e como é realizada a classificação. Na Seção 2.2 são apresentadas as redes neurais artificiais e seu elemento fundamental, o Perceptron. Por último, a Seção 2.3 traz as redes neurais convolucionais como alvo, descrevendo seu funcionamento e apresentando algumas arquiteturas existentes, entre elas a DenseNet que é otimizada pelo método proposto neste trabalho.

2.1 RECONHECIMENTO DE PADRÕES

Nesta Seção serão discutidos os conceitos básicos do reconhecimento de padrões, o que é um padrão, como funciona sua categorização e de que maneira outros tipos de padrões que não visuais podem ser reconhecidos. Vale ressaltar que apesar do reconhecimento de padrões ser uma área que não está atrelada exclusivamente à classificação em imagens e quadros de vídeos, os conceitos apresentados neste capítulo e em todo o texto possuirão este foco, já que o projeto envolve redes neurais classificadoras de imagem.

2.1.1 Padrões

Utilizando os conceitos definidos por Gonzales e Woods (2011, p.568) e Duda, Hart e Stork (2001, p.7), um padrão é um descritor arranjado que revela as características contidas em uma imagem. Por isso muitas vezes ele é conhecido como Vetor de Característica ou *Feature Vector (FV)*.

Um vetor de característica tem como objetivo descrever informações visuais de uma imagem como cor, forma, textura e movimento, e é comumente representada pela letra $\boldsymbol{x}$ minúscula, em negrito, onde cada item x_i representa uma característica e o valor n representa o número total de características. A representação e natureza de cada componente do vetor de característica varia conforme a metodologia utilizada. Como o próprio nome já diz, o vetor de característica é dado por um vetor de tamanho n, conforme pode ser visto na formulação abaixo.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$$

2.1.2 Classes de Padrões

Quando um conjunto, uma família de padrões compartilham propriedades em comum, ali se encontra uma classe de padrão, conforme dito por Gonzales e Woods (2011, p.568). Uma classe contém objetos similares, enquanto outras podem possuir diferenças marcantes, ou em casos mais simples tornando-se mutuamente exclusivas. Por exemplo, assinaturas podem ser verdadeiras ou falsificadas, independente da capacidade de compreensão do observador, uma das duas classes será a correta. Outros problemas podem ser mais complexos de se definir, por exemplo, reconhecer pessoas destras ou canhotas (ignorando a possibilidade de ambidestria). Inclusive, pesquisas na área da saúde são especialmente difíceis pela dificuldade de interpretação dos dados por causa da sua variabilidade. Por exemplo, a necessidade de classificar pacientes de baixo, médio e alto risco é difícil, pois não é trivial definir as características que definem estas classes. (KUNCHEVA, 2004)

De maneira geral, é possível indicar padrões de imagens por $w_1, w_2, w_3, ..., w_c$, onde c é o número de classes. O objetivo das técnicas de reconhecimento de padrões é fazer a associação entre os vetores de características não rotulados e suas respectivas classes de maneira automatizada e com o menor erro possível.

2.1.3 Reconhecimento de Padrões

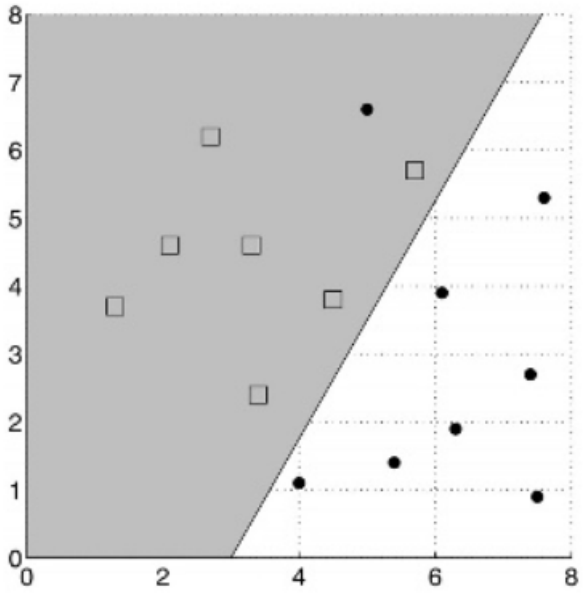
Um dos meios pelo qual é realizada o reconhecimento de padrões é através das funções de decisão ou função discriminante, como explicam Gonzales e Woods (2011, p.570). Se tivermos um vetor de característica $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$ tendo n dimensões e um conjunto de classes de padrões w, o objetivo deste reconhecimento é encontrar funções de decisão W de acordo com a dimensão dos vetores de característica, como $d_1(\boldsymbol{x}), d_2(\boldsymbol{x}), ..., d_n(\boldsymbol{x})$, sendo que o padrão pertence a classe w_i, então ele atende a Equação (2.1).

$$d_i(x) > d_j(x) \qquad para \qquad j = 1, 2, ..., W \qquad e \qquad j \neq i \tag{2.1}$$

Explicando, um padrão $\boldsymbol{x}$ com classe desconhecida passa a pertencer a i-ésima classe se ao substituir o $\boldsymbol{x}$ em todas as funções de decisão façam com que $d_i(x)$ tenha um valor numérico maior que $d_j(x)$.

Já as classes, tem sua separação definida por uma fronteira de decisão onde os valores de $\boldsymbol{x}$ vão resultar em $d_i(x) = d_j(x)$ ou $d_i(x) - d_j(x) = 0$. Logo, a fronteira de decisão pode ter a seguinte estrutura: $d_{ij}(x) = d_i(x) - d_j(x) = 0$, conforme Figura 1.

Figura 1 – Exemplo de classificação de duas classes com função discriminante linear.



Fonte: Kuncheva (2004)

Além de reconhecimentos em imagens, outros tipos de padrões, como sequências de números, podem ser encontrados quando outras técnicas são aplicadas, podendo por exemplo predizer variações em bolsas de valores ou quedas no setor imobiliário, ou mesmo encontrar informações que delimitem a localização de um objeto.

Existem uma série de outros métodos de classificação e redes classificadoras, como as máquinas de vetor de suporte, soluções baseadas em árvores e algoritmos de *clustering*, classificadores estatísticos, entre outros. |Outra abordagem de reconhecimento de padrões para classificação são as redes neurais artificiais, que são capazes de aproximar a fronteira de decisão até mesmo em problemas de alta dimensionalidade. Por esta razão o projeto utilizou as redes neurais convolucionais, pois vários trabalhos utilizando esta abordagem alcançaram o estado-da-arte para reconhecimento de padrões em imagens.

2.2 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNA) foram desenvolvidas com o intuito de modelar as habilidades intelectuais e cognitivas humanas, ou seja, desenvolver de forma

biologicamente inspirada e plausível um algoritmo capaz de aprender de forma similar a um ser humano.

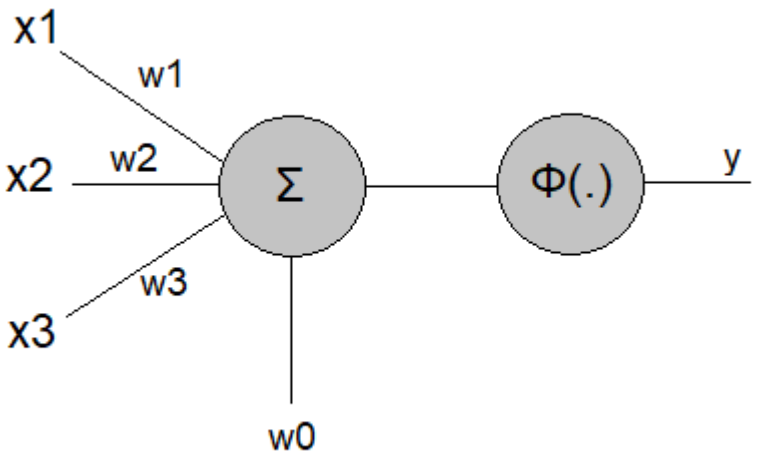
Concebidos para serem esquemas computacionais massivamente paralelos, semelhantes a um cérebro real, as redes neurais evoluíram para se tornar uma valiosa ferramenta de classificação, com uma influência significativa na teoria e na prática do reconhecimento de padrões. As redes neurais são frequentemente utilizadas como classificadores em vários sistemas. Da mesma forma que os classificadores em árvore, as redes neurais são instáveis, ou seja, pequenas alterações nos dados de treinamento podem levar a uma grande alteração no classificador, tanto em sua estrutura quanto em seus parâmetros. (KUNCHEVA, 2004)

Um dos conceitos fundamentais de redes neurais é o modelo *Perceptron*, que foi desenvolvido por Rosenblatt (1958), e inspirado em trabalhos de McCulloch e Pitts (1943), onde mesmo com o passar do tempo e com a utilização de modelos de neurônios artificiais mais desenvolvidos, ele se mantém base do funcionamento das redes neurais.

2.2.1 Algoritmo *Perceptron* e o MLP

O algoritmo *Perceptron* é um modelo matemático que recebe uma certa quantidade de entradas e produz uma saída binária, ou seja, é um modelo capaz de realizar uma discriminação linear dos dados apresentados à ele, e para isso o problema em questão deve ser linearmente separável.

Figura 2 – Estrutura do modelo Perceptron.



Fonte: Elaborado pelo autor. (2019)

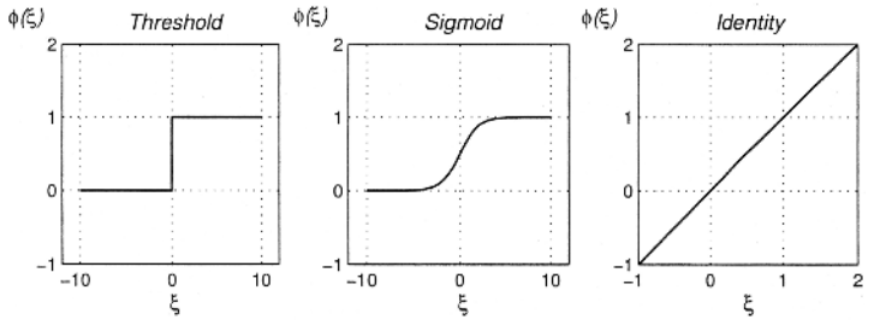
Na Figura 2, podemos ver a estrutura básica de um *Perceptron*, iniciando com

as entradas, neste caso três, um neurônio (unidade) artificial e uma saída. Para calcular esta saída, Rosenblatt (1958) trouxe como proposta a introdução de pesos, ou seja, valores numéricos que expressam o grau de importância de uma entrada em relação a sua saída. Como o modelo inicial do *Perceptron* é um modelo linear, a saída esperada do neurônio são os valores zero e um, e estes valores são determinados pela soma ponderada da Equação (2.2):

$$y = \phi(\sum w_i \times x_i + w_0) \tag{2.2}$$

Onde w_i são os pesos, x_i são os valores de entrada e w_0 é conhecido como viés, ou *Bias*, um valor que desloca a função linear da origem. ϕ é conhecido como função de ativação e é responsável por determinar a forma e a intensidade da alteração dos valores transmitidos de um neurônio a outro. Uma função de ativação utilizada no *Perceptron* é a *Step Function*, função degrau ou ainda função *Threshold*, que caso receba como saída um valor maior que zero, o valor final de y será um, caso contrário será zero. Este tipo de função é, como dito anteriormente, utilizada para gerar uma saída de um problema binário, porém no decorrer do capítulo será descrito como a evolução das redes neurais e o aumento da não-linearidade dos problemas fez com que novas funções passassem a ser utilizadas. Outra função de ativação amplamente utilizada é a função Sigmoid (Figura 3b) que possui derivada positiva em todos os pontos.

Figura 3 – Projeção da funções de ativação *Step Function*/*Threshold*, *Sigmoid* e Linear/*Identity*.

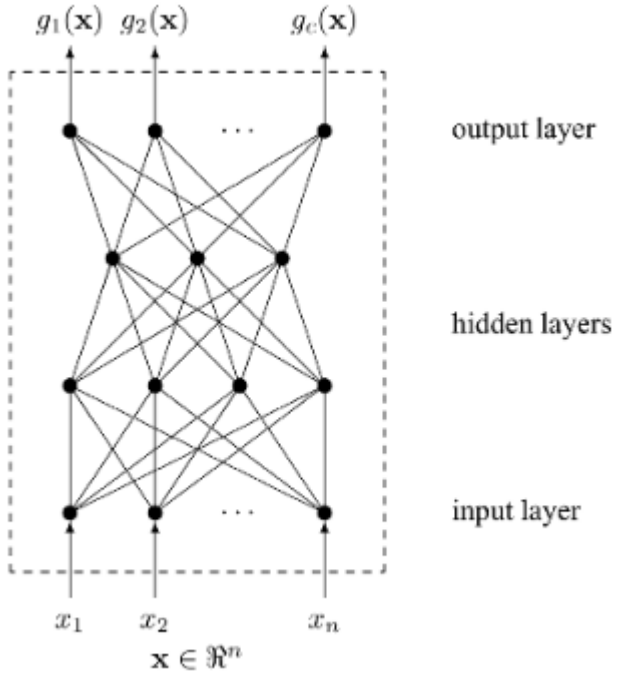


Fonte: Kuncheva (2004)

Rumelhart et al. (1988) traz em seu artigo o conceito de retropropagação (descrito ainda nesta Seção) e a possibilidade de conectar diversos modelos Perceptron em múltiplas camadas, esta estrutura ficou conhecida como *MultiLayer Perceptron*, ou MLP. Sua estrutura denominada *feed-forward* se dá pois a saída de um neurônio e todos os seus intermediários sempre estão conectados a todos os neurônios existentes na camada superior. Esta estrutura também pode ser chamada de Densa ou Comple-

tamente Conectada. (KUNCHEVA, 2004) É possível ver a estrutura densa conforme Figura 4, onde $g_n(x)$ são as saídas da rede e x_n são as entradas;

Figura 4 – Modelo de classificador MLP. Vale notar a estrutura que conecta todos os neurônios de uma camada à sua camada anterior, ou seja, a estrutura *feed-forward*.



Fonte: Kuncheva (2004)

Para treinar uma rede MLP foi necessário desenvolver um novo método de treinamento, que ficou conhecido como treinamento por retropropagação de erro (back-propagation), que recalcula os valores do gradiente para cada peso da rede a cada iteração até um determinado limiar. Sua forma geral pode ser descrita da seguinte maneira:

$$w \leftarrow w - \eta \frac{\delta E}{\delta w} \tag{2.3}$$

Aqui, w representam os pesos, η é valor da taxa de aprendizado e a expressão $\frac{\delta E}{\delta w}$ consiste em calcular as derivadas parciais da função de erro E para os pesos do vetor w. Porém o algoritmo *Perceptron* necessitava de uma modificação para que a retropropagação se tornasse viável. Os valores gerados pela função de ativação *Step Function* não eram parcialmente deriváveis e portanto não era possível atualizar os pesos, isso foi solucionado permitindo que valores entre zero e um pudessem ser dados como saída de uma nova função de ativação conhecida como função Sigmóide.

A função sigmóide é uma função não-linear que, conforme mostra a Figura 3, tem na sua projeção os valores de Y de forma íngreme, portanto ligeiras alterações

no valor de X vão fazer os valores de Y sempre tenderem à uma extremidade da curva. Isso é bom para a classificação pois permite uma inferência mais clara. Outra vantagem é que os valores sempre estarão contidos em um intervalo entre zero e um, e não tendendo ao infinito, como em uma ativação linear. Conforme Han e Moraga (2005), a formulação da função de ativação sigmóide é dada pela Equação (2.4):

$$A = \frac{1}{1 + e^{-x}} \tag{2.4}$$

Onde e é o número de Euler e x o valor da entrada. Porém ainda há uma desvantagem, conforme os valores vão se aproximando das extremidades da curva, Y já não responde de maneira tão drástica às mudanças de X, portanto o gradiente desta região é reduzido, e a rede passa a não ter mais um aumento de aprendizado, ou caso tenha, mudanças significativas não ocorrerão em um curto período de tempo. Porém a função sigmóide ainda é utilizada em muitos problemas de classificação até hoje.

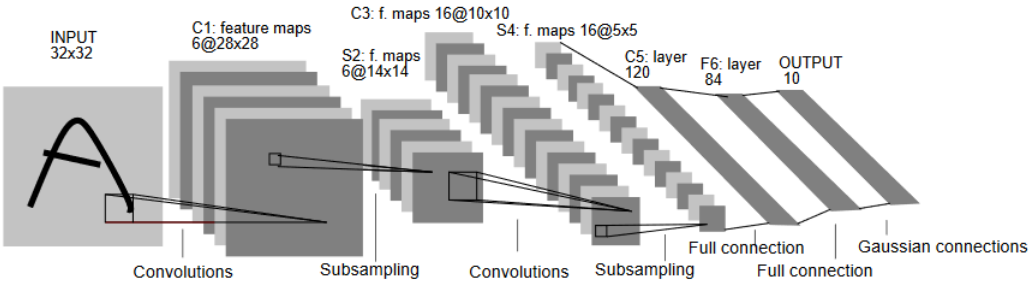
2.3 REDES NEURAIS CONVOLUCIONAIS

Com o desenvolvimento das redes neurais, uma nova abordagem surgiu criando modelos que fossem invariantes a certas transformações nas entradas. Esta pode ser considerada a base das **Redes Neurais Convolucionais**, ou *Convolutional Neural Networks (CNN)*, que é amplamente utilizada em problemas de visão computacional. A **LeNet**, uma CNN desenvolvida por LeCun et al. (1998) é considerada a primeira do gênero e seus autores os pioneiros nesta área de pesquisa. (BISHOP, 2006)

Tomando como exemplo a tarefa de reconhecimento de dígitos manuscritos, cada imagem possui um conjunto de pixels, que contém valores que determinam a intensidade de cada um e a saída esperada é a distribuição das probabilidades entre os dez possíveis dígitos. A identidade da entrada é invariante a translações, escala, pequenas rotações e uma abordagem simples envolveria enviar a imagem para uma rede MLP, e dado um conjunto de treino em larga escala, a rede aprenderia as invariâncias apropriadas pela repetição da exposição aos exemplos.

Porém esta abordagem deixa passar uma propriedade chave das imagens, pixels próximos possuem uma correlação mais intensa do que pixels distantes. Boa parte das soluções modernas em visão computacional exploram estas propriedades através dos extratores de características locais em sub regiões de uma imagem. Estas características podem ser processadas ou mixadas em estágios avançados com o intuito de encontrar novas características de alto nível e retornar informações sobre a imagem como um todo. (Bishop (2006))

Figura 5 – Estrutura da rede LeNet, utilizada para detectar dígitos manuscritos.



Fonte: LeCun et al. (1998)

Utilizando como exemplo a estrutura da rede LeNet é possível identificar o funcionamento básico de uma CNN, sua estrutura pode ser vista na Figura 5. Em uma camada convolucional as unidades são organizadas em planos onde cada um deles é denominado **Mapa de Característica**, ou *Feature Map (FM)*. Cada unidade de um mapa de característica recebe como entrada uma subregião da imagem, e todas as unidades de um mapa de característica são restringidos a compartilhar os mesmos pesos (parâmetros). Por exemplo, um mapa de característica consistindo de cem unidades dispostos em uma matriz 10x10, e recebendo em cada unidade como entrada uma área de 5x5 *pixels* da imagem terá um total de 25 parâmetros ajustáveis mais um parâmetro ajustável para o viés. Lembrando que este não é o cálculo da quantidade de parâmetros que uma camada convolucional possui, apenas a quantidade de parâmetros de um único mapa de característica. (LECUN et al., 1998)

Para calcular a quantidade de parâmetros de uma camada convolucional é necessário utilizar a Equação (2.5):

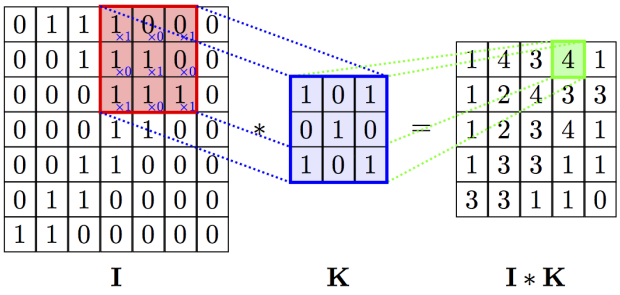
$$p = ((i \cdot m \cdot n) + 1) \cdot o \tag{2.5}$$

Sendo que i é a quantidade de mapas de característica de entrada, m e n são a largura e altura da subárea a ser convolucionada, também conhecido como tamanho do filtro (*kernel*), a constante 1 é o viés e o é a quantidade de mapas de característica gerados na saída da camada. Os valores de entrada extraídas de uma parte da imagem são linearmente combinadas utilizando os pesos e viéses, e seu resultado é transformado por uma função de ativação não-linear, conforme Figura 6. Se tratarmos as unidades como detectores de características, então todas as unidades de um mapa de característica detectam o mesmo padrão em diferentes locais da imagem.

Durante o compartilhamento dos pesos, se uma imagem for modificada, a ativação do mapa de característica será modificada na mesma intensidade. Isto permite

alcançar de maneira aproximada a invariância das saídas da rede à translações e distorções na entrada.

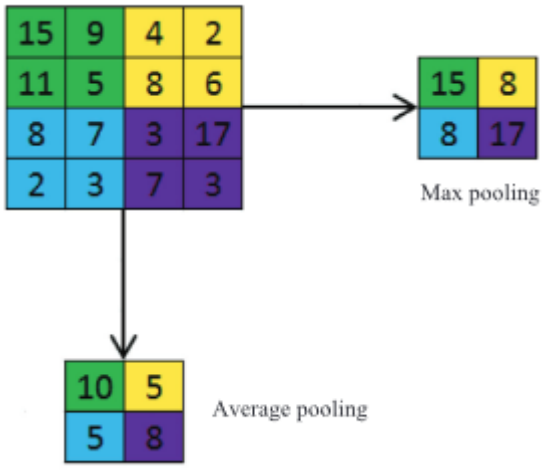
Figura 6 – Operação de convolulção em uma camada convolucional. Cada unidade do filtro (em azul) é multiplicado pelos valores contidos na subárea da imagem (em vermelho), em seguida os valores são somados para gerar um mapa de característica novo (em verde).



Fonte: GitHub/TikZ (2019)

As saídas de uma camada convolucional formam a entrada de uma camada de Subamostragem (*Subsampling/Pooling*), uma camada que tem como objetivo reduzir a dimensionalidade dos mapas de características, podendo efetuar esta redução através da computação da média ou do valor máximo de uma área de filtro que serve de entrada, multiplicado por um peso e um viés. Esta camada tem como saída uma série de mapas de característica menores que os originais, conforme Figura 7. Tem como vantagem da redução de resolução a necessidade de menor poder computacional para processar os mapas subsequentes, conforme Figura 7.

Figura 7 – Operação de subamostragem. No *pooling* máximo o maior valor da subárea é enviado para o novo mapa de característica, enquanto que no *pooling* médio é feita a média dos valores da subárea.



Fonte: Rawat e Wang (2017)

Falando sobre as funções de ativação, como citado ao final da Seção 2.2.1, a função sigmóide tem uma tendência a diminuir seu aprendizado conforme se aproxima à uma das extremidades da curva, isso foi contornado com a utilização da função de ativação de unidade linear retificada, ou ReLU, que é uma função que possibilita retornar valores entre zero e infinito de maneira linear.

Esta função trouxe algumas vantagem, conforme dito por Glorot, Bordes e Bengio (2011): Funções como a sigmóide utilizam cálculo exponencial, enquanto o cálculo do ReLU é linear, portanto os cálculos são mais simples e baratos computacionalmente. Outra vantagem é a capacidade de gerar valores zerados, ao invés de aproximados de zero em caso de números negativos, pois isto permite a ativação das unidades ocultas mesmo contendo o valor zero. Isto se chama representação esparsa e permite acelerar e simplificar o modelo. Além disto o comportamento linear da função facilita o trabalho de otimização e contorna o desaparecimento do gradiente que ocorre com outras funções não-lineares e a possibilidade de treinar redes ainda mais profundas utilizando a retropropagação do erro.

Já a função de ativação ReLU, ou Unidade de Retificação Linear é uma função de ativação definida por $y = max(0, x)$, possuindo a seguinte projeção, conforme Figura 19;

Esta função de ativação é linear para valores positivos e retorna o valor zero para valores negativos, isto significa que a utilização desta função de ativação permite diminuir o tempo de treinamento e execução, devido a facilitação dos cálculos providos pela função ReLU. Além disto esta função converge rapidamente, pois a linearidade evita a saturação, evitando também a dissipação do gradiente.

Em uma arquitetura prática, existem uma série de pares de camadas convolucionais e camadas de *pooling*. Cada estágio traz um aumento do grau de invariância às transformações da camada anterior. Cada camada pode gerar um número maior de mapas de característica, percebendo que a redução gradual da resolução é compensada com o crescimento do número de camadas. Ao final da rede, tipicamente são utilizadas as camadas densas, ou seja, camadas totalmente conectadas de uma MLP simples, com alguma função de ativação relacionada à tarefa desejada, como a *Softmax*, no caso de classificação multiclasse. A função *Softmax* é uma função de ativação que retornam as probabilidades de distribuição de classes.

A função de ativação *Softmax* é uma função que transforma números em probabilidades, mas especificamente, tem como retorno um vetor com a distribuição das probabilidades das classes. A formulação desta função é dada na Equação (2.6):

$$S(y_i) = e^{y_i} / \sum_j e^{y_i} \tag{2.6}$$

Sendo y_i representa cada item representado no vetor y e e é o número de Euler. Colocar o y_i no expoente é necessário para evitar valores negativos, uma vez que a saída varia de -infinito a +infinito, e saídas negativas fazem com que a soma não seja correta.

Para calcular a quantidade de parâmetros em uma camada densa é necessário aplicar a Equação (2.7):

$$(n + 1) \cdot m \tag{2.7}$$

Sendo n a quantidade de unidades na entrada e m a quantidade de unidades na saída. A constante 1 é referente ao viés. Além das camadas convolucionais e de subamostragem, outros tipos de camadas podem ser utilizados para regularizar os valores dos pesos, auxiliando no aprendizado da rede.

2.3.1 Camada de normalização em lote

A camada de normalização em lote (*Batch Normalization - BN*) foi desenvolvida por Ioffe e Szegedy (2015) para acelerar o aprendizado de uma CNN. A normalização em lote normaliza a distribuição das entradas nas camadas da rede, mantendo os valores das entradas dentro do mesmo intervalo. Algumas das vantagens da BN são o aumento da independência entre as camadas, a possibilidade de uso de altas taxas de aprendizado, que podem treinar problemas mais complexos devido à restrição nos valores da ativação e reduz a adaptação excessiva (*Overfitting*), de maneira similar à camada *Dropout*, descrita na Seção 2.3.2.

A normalização em lote adiciona ruído para cada ativação nas camadas. Tende-se a utilizar menos camadas *Dropout* conforme o aumento da utilização da BN, pelo menor número de perda de informações. Porém é bom ressaltar que não se deve depender exclusivamente da BN para regularização, utilizando-o em conjunto com outras técnicas como o *Dropout*. (IOFFE; SZEGEDY, 2015)

Ainda segundo Ioffe e Szegedy (2015), para efetuar a normalização, é necessário acompanhar as distribuições de cada dimensão normalizada, para isto são calculados quatro parâmetros por recurso da camada anterior, estes parâmetros garantem a propagação e a retropropagação dos dados de maneira correta. Portanto, para calcular a quantidade de parâmetros em uma camada de normalização em lote basta multiplicar a quantidade de mapas de entrada por 4.

2.3.2 Camada *dropout*

A camada *Dropout* é um método de regularização de rede que ignora neurônios aleatoriamente durante a fase de treino. Por ignorar entende-se que estas uni-

dades não são consideradas durante algum processo particular, seja na otimização ou na retropropagação. A cada estágio do treino, unidades podem ser desconsideradas em uma escala de probabilidade $1 - p$ ou mantidas com uma probabilidade p. (SRIVASTAVA et al., 2014)

A principal utilização da camada *Dropout* é a prevenção do ajuste excessivo, pois uma camada totalmente conectada utiliza muitos parâmetros e, muitas vezes os neurônios desenvolvem codependência entre si durante o treinamento, o que restringe o poder individual de cada unidade, levando ao ajuste excessivo.

O *Dropout* é uma abordagem que visa a redução do aprendizado interdependente entre as unidades, seja na fase de treino, onde para cada iteração se ignora uma fração aleatória de nós e suas ativações; ou na fase de teste, onde todas as ativações são utilizadas, mas primeiramente reduzidas a um fator p.

2.3.3 Principais arquiteturas de Redes Neurais Convolucionais

Além da LeNet, outras arquiteturas de redes classificadoras foram desenvolvidas conforme a evolução da área, inserindo novos conceitos e aumentando a qualidade e profundidade das características extraídas a fim de reduzir ainda mais as taxas de erro durante o período de treinamento. Nas próximas seções serão descritas algumas das principais arquiteturas existentes das CNN.

2.3.3.1 AlexNet

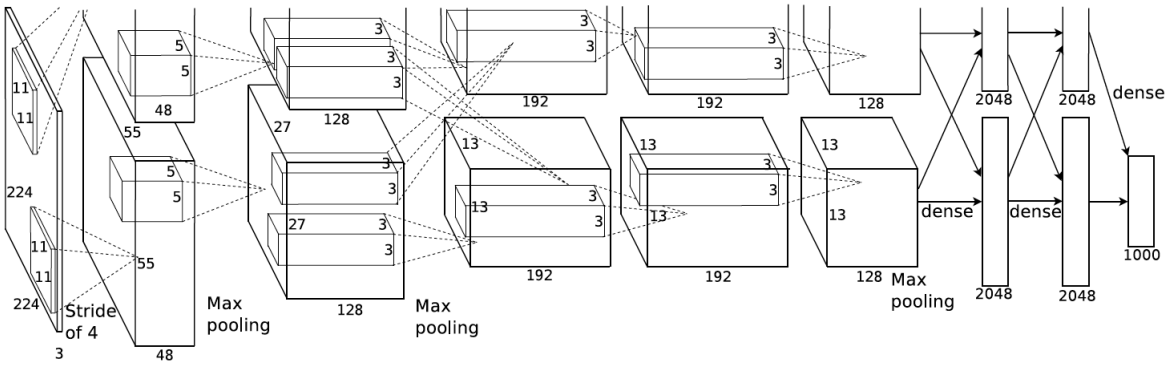
Com uma arquitetura similiar ao LeNet, porém mais profunda, com mais filtros por camadas e dividido em dois pipelines, o AlexNet de Krizhevski, Sutskever e Hinton (2012) é composto de camadas convolucionais de filtros 11x11, 5x5 e 3x3, seguidos de camadas de *pooling* e *dropout*. Além da utilização do aumento artificial de dados como método de treinamento, que são transformações aplicadas no conjunto de dados original que auxiliam a rede a aprender as variações de uma mesma imagem. Ao final de cada camada é aplicada a função de ativação ReLU e as últimas camadas são camadas densas, possuindo uma função de ativação *softmax* na última delas.

Possuindo 60 milhões de parâmetros, o AlexNet obteve a melhor taxa de acurácia na competição ILSVRC 2012 conseguindo 63.3% de acurácia para classificações *Top*-1 (considerando o primeiro resultado) e 84.7% de acurácia para classificações *Top*-5.

2.3.3.2 GoogLeNet/Inception

Criado por Szegedy et al. (2014), a rede *Inception* também foi desenvolvida com base na rede LeNet e tem como objetivo encontrar uma estrutura de ótimos lo-

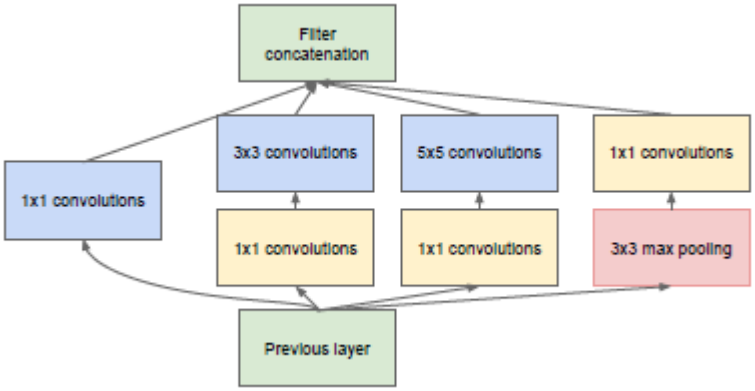
Figura 8 – Arquitetura da rede AlexNet.



Fonte: Krizhevski, Sutskever e Hinton (2012)

cais no espaço de busca e repeti-las. Além disso, como o nome sugere, possuem os módulos denominados *Inception* que são camadas convolucionais encapsuladas, ou seja, que possuem outras camadas dentro dela e suas saídas são concatenadas com outra convolução que ocorre em paralelo a execução do módulo, conforme Figura 9.

Figura 9 – Módulo Inception, composto por uma série de convoluções que são concatenadas, podendo extrair uma maior quantidade de características com menos parâmetros.



Fonte: Szegedy et al. (2014)

A arquitetura desta rede consiste de camadas de convolução e pooling seguidos destes módulos *Incpetion*, que conforme vão executando permitem extrair características de objetos em diversas escalas, ou seja, é possível detectar diversas escalas de objetos em uma execução da rede.

Nas versões posteriores, sendo a quarta versão a mais atual, foram adicionadas as camadas *Dropout* nos módulos e nas camadas densas para prevenir o excesso de treinamento e as camadas de atalho, que concatenam mapas de características de camadas anteriores, chamados de resíduos, para recuperar informações que possam

ter sido perdidas no processo de convolução, sendo uma união entre a arquitetura *Inception* e a arquitetura ResNet, que será descrita neste capítulo. Estas novas implementações permitiram que a *GoogLeNet* se tornasse duas a três vezes mais veloz do que sua primeira versão, além de alcançar 80.1% de acurácia para o ILSVRC 2016 na classificação *Top*-1 e 95.1% de acurácia para a classificação *Top*-5, possuindo 55.8 milhões de parâmetros.

2.3.3.3 VGGNet

É uma arquitetura de rede criada por Simonyan e Zisserman (2015), através do Visual Geometry Group (VGG), que acabou nomeando a rede com sua sigla. É composta de dezesseis ou dezenove camadas onde a entrada possui 224 x 224 pixels, e o único pré-processamento existente é a extração dos valores médios RGB para cada pixel. Em seguida a entrada passa por uma série de camadas convolucionais e de max-pooling utilizando filtros pequenos de tamanho 3 x 3 e valor de passo 1 para as camadas convolucionais e valor de passo 2 para o max-pooling.

Figura 10 – Estrutura da rede VGGNet.

Fonte: Basaveswara (2019)

Após as camadas convolucionais ficam as camadas densas ou totalmente conectadas, as duas primeiras com 4096 saídas, seguido da última camada com o número de saídas necessárias para a classificação. Além disso todas as camadas possuem a função de ativação ReLU para dar não-linearidade ao modelo, com exceção da última que possui a função de ativação Softmax. E em nenhum momento há normalização dos dados pois segundo o autor, não houve nenhuma melhora significativa de acurácia, ao contrário, apenas o consumo de memória e cálculos.

A versão de 19 camadas alcançou com o conjunto ILSVRC 2014 um total de 74.5% de acurácia na classificação *Top*-1 e 92.0% de acurácia para a classificação *Top*-5, possuindo 144 milhões de parâmetros, uma quantidade que pode tornar o uso desta rede um desafio para máquinas com baixo poder computacional.

2.3.3.4 ResNet

He et al. (2016) trouxe um novo conceito, as redes residuais, ou ResNet. Conforme as convoluções ocorrem, algumas informações da imagem se perdem ou se

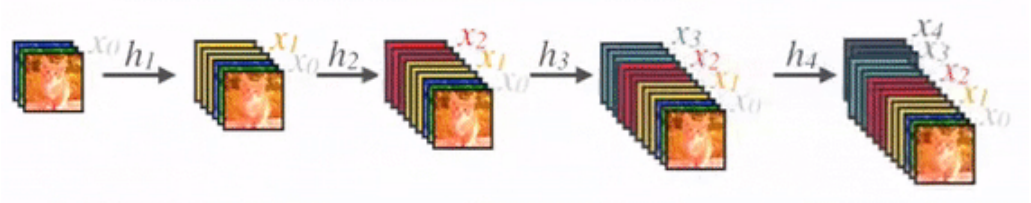
degradam, com isto a ResNet trouxe as camadas de atalho, que guardam mapas de características de camadas anteriores e concatenam com as mais recentes, recuperando algumas das informações perdidas. Isto permitiu que características ainda mais profundas pudessem ser extraídas, e juntamente com a grande utilização de normalização em lote, esta rede conseguiu um total de 79.9% de acurácia na classificação *Top*-1 e 95.2% de acurácia para a classificação *Top*-5 no conjunto de dados ILSVRC 2015.

2.3.3.5 DenseNet

Desenvolvida por Huang et al. (2017), a DenseNet trouxe o conceito de CNN totalmente conectadas, ou seja, todas as camadas convolucionais estariam conectadas a seus antecessores, compartilhando seus mapas de característica e reduzindo o número de parâmetros necessários para o aprendizado da rede. Devido a esta arquitetura conectada e a aquisição de resultados comparáveis as versões mais recentes da ResNet, esta foi a arquitetura escolhida para a implementação do método de otimização deste projeto.

Para compreender o funcionamento da DenseNet é necessário compreender os seus conceitos mais básicos que são os Blocos Densos e a propagação de mapas de característica. Em uma rede comum, uma imagem passa através das camadas através de múltiplas convoluções e assim obtém-se as características de alto nível. Já na DenseNet, cada camada possui como entrada todos os mapas predecessores e sua saída é compartilhada com todas as camadas subsequentes através da concatenação, como uma espécie de “conhecimento coletivo” entre as camadas, assim como mostra a Figura 11.

Figura 11 – Conectividade entre os canais da rede, cada camada recebe os mapas de características da camada anterior, evitando o recálculo do mesmo e produzindo novos mapas a cada camada.



Fonte: Tsang (2018)

Assim uma quantidade fixa de mapas de características é gerada para cada camada, através do valor denominado taxa de crescimento ou k. Com esta arquitetura, as redes podem ser mais compactas e o número de canais podem ser menores, além de ser mais eficiente em termos de cálculo e memória.

Dentro de um bloco denso existem duas possíveis sub-camadas, a camada de gargalo e a camada densa, ou camada de composição. A camada de gargalo reduz a complexidade e o tamanho da rede realizando uma normalização em lote seguido de uma ativação ReLU, e por fim uma convolução 1x1, que terá uma saída de $4 \times k$ canais. Já a camada densa se diferencia na execução de uma convolução 3x3 com k saídas, que serão concatenados com os mapas já produzidos.

Entre os blocos densos, existem as camadas de transição, que são responsáveis por reduzir a resolução dos mapas e o custo computacional da rede. Para deixar a rede mais compacta, é possível reduzir a quantidade de mapas de características durante a transição, se um bloco denso gera m mapas de características, permite-se que a camada de transição seguinte gere até θm mapas de características, onde θ está entre zero e um e é denominado fator de compressão. (HUANG et al., 2017)

A DenseNet conseguiu um total de 79.2% de acurácia na classificação *Top*-1 e 94.71% de acurácia para a classificação *Top*-5 no conjunto de dados ILSVRC 2012.

2.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Dentro dos diversos tipos de classificadores existentes, destaca-se a robustez e eficiência das CNN frente à problemas de grande escala como o aprendizado de grandes conjuntos de dados. Além disso a utilização de camadas reguladoras como a *Dropout* e a normalização em lote permitem que se previna o ajuste excessivo e problemas de generalização da rede.

Enquanto outras soluções exploravam características mais profundas através do aumento exponencial de características e de parâmetros, a arquitetura DenseNet destacou-se pela possibilidade de explorar características profundas utilizando-se de poucos parâmetros ajustáveis, alcançando resultados significativos não apenas para o conjunto de dados ImageNet, como para conjuntos como CIFAR 10, com 96.54% de acurácia e CIFAR 100, com 82.62% de acurácia. Uma tabela com os resultados alcançados pelas arquiteturas citadas neste capítulo para a base ImageNet via método *Top*-1 (considerando apenas o primeiro resultado da classificação) pode ser vista na Tabela 1.

Tabela 1 – Desempenho das redes citadas na base ImageNet.

Rede	**Acurácia (%)**	**Parâmetros**
AlexNet	63,3	60 Milhões
GoogLeNet	80,1	55.8 Milhões
VGGNet	74,5	144 Milhões
ResNet	81,2	25 Milhões
DenseNet	79,2	3.5 Milhões

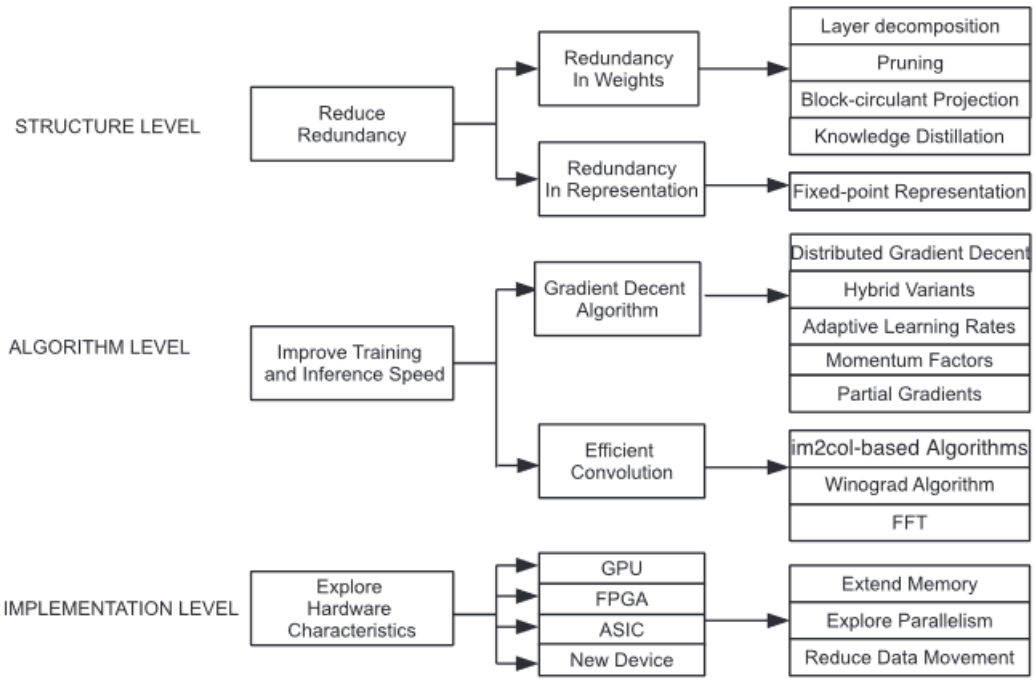
Fonte: Elaborado pelo autor. (2019)

3 TRABALHOS RELACIONADOS

Como o objetivo deste trabalho é realizar otimizações em redes neurais convolucionais, buscou-se conhecer os trabalhos que norteiam a área e as publicações mais recentes que ditam o estado da arte. Foi realizada uma pesquisa de literatura com foco no tema da pesquisa que são os métodos de aceleração e redução de parâmetros para redes neurais convolucionais.

A pesquisa analisou outras revisões sistemáticas e artigos primários com o objetivo de compreender as técnicas e metodologias mais recentes, sem esquecer de compreender as origens do problema e quais foram as primeiras abordagens para a sua resolução. Para a realização da pesquisa foi elaborado um protocolo de pesquisa baseado no trabalho de Mian et al. (2005) utilizando-se de alguns critérios para aceitar ou rejeitar artigos, como palavras-chave, leitura do resumo, tema relevante, entre outros.

Figura 12 – Taxonomia dos métodos de aceleração para CNN.



Fonte: Zhang et al. (2019)

Uma taxonomia que categorizou os métodos de aceleração foi adotada com base na revisão de literatura produzida por Zhang et al. (2019), mostrada na Figura 12, sendo que os artigos descritos neste capítulo são contemplados nesta taxonomia e são categorizados especificamente no nível de estrutura e redução de redundân-

cia, focado nas redundâncias em pesos. Além disso, os artigos descritos servem para mostrar as técnicas e métodos mais atuais relacionados ao tema, embora estas soluções não possuam necessariamente algum tipo de similaridade com o projeto de mestrado desenvolvido.

Muitos processos de treinamento e inferências podem ser acelerados reduzindo as redundâncias nas estruturas das redes, sendo estas redundâncias na forma de pesos e suas representações. Segundo Denil et al. (2013) e Sainath et al. (2013), alguns pesos aprendidos pelas redes neurais são correlacionados, assim tais pesos podem ser previamente preditos ou mesmo dispensados do processo de aprendizagem. Nas próximas seções serão descritos alguns métodos de otimização de redes através da remoção das redundâncias em pesos, nas quais seguirão a taxonomia de Zhang et al, e será dividido entre os temas de decomposição de camada, redes de poda, projeção bloco-circulante e *Knowledge Distillation*.

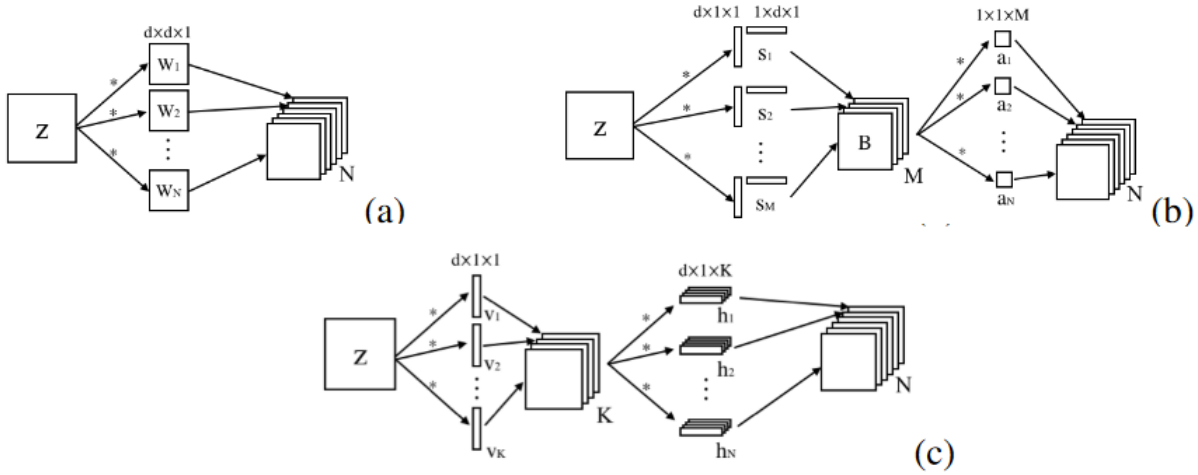
3.1 DECOMPOSIÇÃO DE CAMADAS

A decomposição de camadas é um método aplicado em camadas convolucionais ou densas e tem o intuito de comprimir o tamanho de uma rede decompondo as matrizes de pesos das camadas da rede. Como por exemplo, Jaderberg, Vedaldi e Zisserman (2014) sugerem a decomposição dos mapas de características para a implementação de um banco de filtros intermediário, explorando redundâncias de pesos entre diferentes filtros e canais de uma camada convolucional e gerando mais mapas de características com menos parâmetros.

Os autores descrevem dois métodos, ambos com funcionalidade similar, consistindo em gerar **S** vetores de características para cada canal da entrada com forma $d \times 1 \times 1$ e $1 \times d \times 1$, e combiná-los linearmente para gerar um banco de filtros intermediário **M**, além disto a entrada gera um segundo conjunto de características com forma $1 \times 1 \times 1$ que é combinado linearmente com o banco **M**, gerando os N mapas de características finais, utilizando um quantidade menor de filtros para realizar a convolução na entrada. Em um cenário de reconhecimento de texto, este método permitiu uma redução de 4,5x no tempo de execução da rede com uma queda de apenas 1% na acurácia. Um exemplo deste trabalho está apresentado na Figura 13.

Já no artigo de Ding et al. (2017), que desenvolveu uma rede para reconhecimento *offline* de escrita à mão baseada na arquitetura VGGNet, a técnica utilizada para decompor as camadas foi a decomposição de Tucker, aplicado sobre os tensores de entrada de cada camada. Uma camada convolucional x pode ser dada por um tensor de 3 dimensões $x \in R^{H \times W \times I}$, onde H,W e I representam, altura, largura e canais de entrada, respectivamente. Esta entrada é mapeada por O filtros que pertencem a

Figura 13 – Exemplos da funcionalidade da decomposição de camadas. O exemplo *a* mostra o funcionamento comum de uma camada convolucional, os exemplos *b* e *c*, o funcionamento das camadas através da decomposição.



Fonte: Jaderberg, Vedaldi e Zisserman (2014)

um tensor $R^{k_h \times k_w \times I}$ e tem como saída um tensor $Y \in R^{H' \times W' \times O}$. Essa operação de convolução pode ser dada pela Equação (3.1):

$$\begin{aligned} Y^{h',w',o} &= \sum_{p=1}^{k_h} \sum_{q=1}^{k_w} \sum_{i=1}^{I} F_{p,q,i,o} X_{h_p,w_q,i} \\ h_p &= (h'-1)s_h + p - p_h \\ w_q &= (w'-1)s_w + q - p_w \end{aligned} \tag{3.1}$$

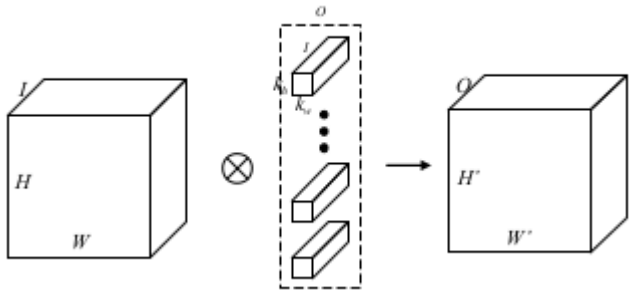
Comumente, a decomposição SVD é utilizada para comprimir matrizes de pesos de duas dimensões, porém a decomposição de Tucker é uma extensão do SVD que consegue comprimir um tensor de quatro dimensões e pode ser representado neste caso conforme a Equação (3.2):

$$F_{p,q,i,o} = \sum_{r3}^{R3} \sum_{r4}^{R4} C_{p,q,r3,r4} W_{i,r3} Z_{0,r4} \tag{3.2}$$

Onde $C \in R^{k_h \times k_w \times R3 \times R4}$ é um tensor e W e Z são matrizes fatoradas. Após substituir a Equação (3.2) na Equação (3.1), realizando os somatórios e rearranajando a equação, obtém-se uma série de expressões que equivalem a convolução original, apresentada na Figura 14.

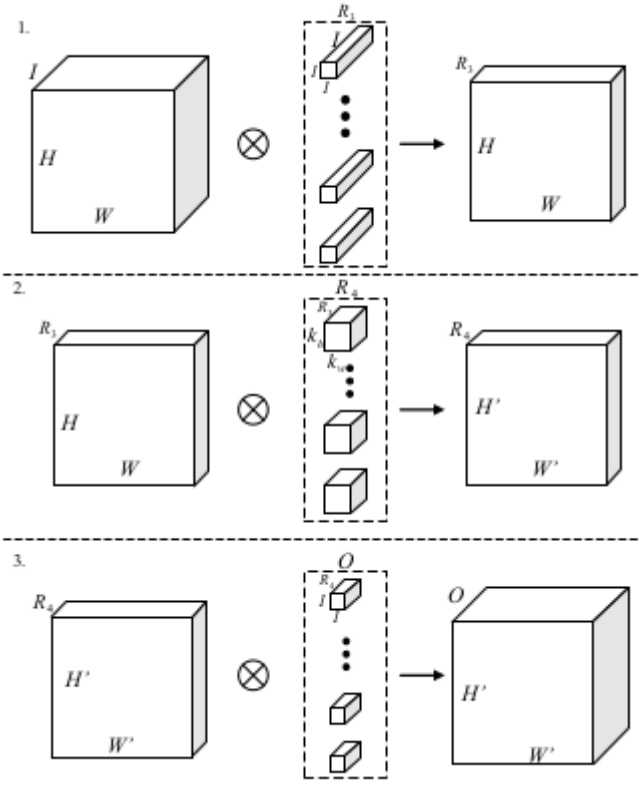
$$Y^{(1)}_{h,w,r3} = \sum_{i=1}^{I} W_{i,r3} X_{w,h,i} \tag{3.3}$$

Figura 14 – Operação de convolução tradicional.



Fonte: LeCun, Denker e Solla (1990)

Figura 15 – Ilustração da decomposição de Tucker em uma camada convolucional.



Fonte: Ding et al. (2017)

$$Y^{(2)}_{h',w',r4} = \sum_{p=1}^{k_h} \sum_{q=1}^{k_w} \sum_{r3=1}^{R3} C_{p,q,r3,r4} Y^{(1)}_{h_p,w_q,r3} \tag{3.4}$$

$$Y_{h',w',o} = \sum_{r4=1}^{R4} Z_{o,r4} Y^{(2)}_{h',w',r4} \tag{3.5}$$

Assim o resultado é a camada original decomposta em três camadas menores, e a perda da acurácia é compensada com ajuste-fino da rede e o aumento de épocas de treinamento com um pequeno valor na taxa de aprendizado.

3.2 REDES DE PODA

Pruning ou redes de poda é uma técnica que envolve a criação de uma rede neural nova a partir de outra já existente, desativando unidades e removendo conexões menos relevantes para o aprendizado. Esta abordagem pode reduzir a quantidade de parâmetros de uma rede em grande escala (com casos de até 60% de otimização dos pesos), porém esta otimização tem um custo de acurácia que por vezes é considerado irrelevante ao ser comparado com os resultados de otimização da abordagem.

Pioneiro das redes neurais convolucionais, LeCun, Denker e Solla (1990) também iniciaram as pesquisas sobre otimização e aceleração de redes neurais alguns anos antes, com seu trabalho *Optimal Brain Damage*. Em seu artigo, os autores conseguiram remover mais de 50% dos pesos existentes de redes densas (a LeNet, neste caso) tendo os mesmos valores de acurácia do modelo original.

Seu funcionamento ocorre da seguinte maneira: primeiramente a rede completa é treinada até alcançar um resultado satisfatório, e uma vez alcançado a rede calcula a segunda derivada para cada parâmetro da rede, e estes valores é que serão retropropagados para a atualização de pesos, em seguida, cada parâmetro têm sua "saliência" calculada e por fim os parâmetros são ordenados de acordo com a saliência para serem mantidos ou removidos de acordo com um limite mínimo. Após a remoção dos parâmetros a rede é retreinada para comparação de resultados.

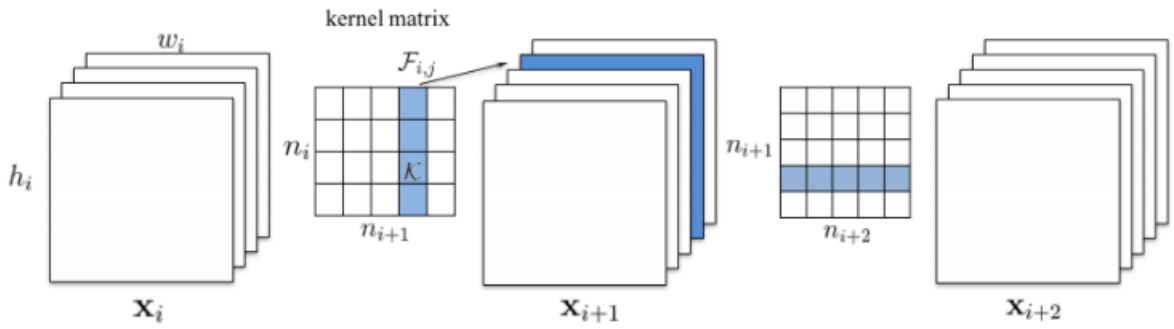
Diferente do modelo de LeCun, Denker e Solla (1990), Li et al. (2016) agora trazem uma abordagem para redes neurais convolucionais onde os filtros são categorizados de maneira a remover os filtros e mapas de características menos relevantes.

Para cada filtro F_i, j é calculado a soma do pesos $s_j = \sum_{l=1}^{n_i} \sum |K_l|$, sendo n_i o número de canais de entrada e K_l os valores dos filtros daquela camada. Em seguida s_j é ordenado e m filtros são removidos de acordo com o menor valor, juntamente com seus respectivos mapas de característica, os filtros correspondentes ao mapa deletado também são removidos em camadas posteriores, conforme Figura 16. Por último um novo filtro e mapa são gerados e colocados no lugar do que foi removido.

Como resultado este tipo de abordagem conseguiu manter os valores da taxa de erro, tendo um aumento médio de 1%, enquanto algumas redes conseguiram 64% de otimização de parâmetros.

Já He et al. (2017) traz uma solução para redes de poda em duas etapas, a

Figura 16 – Podar um filtro resulta na remoção das cararcterísticas correspondentes dos mapas de dos filtros nas camadas seguintes.



Fonte: Li et al. (2016)

primeira etapa envolve selecionar os canais mais representativos e redundantes da rede através de uma regressão LASSO, e em seguida reconstruir as saídas utilizando os mínimos quadrados lineares. Com uma arquitetura baseada da VGGNet, a rede obteve como resultado um redução de 20% no tempo de processamento e se tornou 50% mais veloz na sua versão baseada na GoogLeNet.

3.3 PROJEÇÃO BLOCO-CIRCULANTE

Esta abordagem utiliza o princípio de matrizes circulantes para otimização de redes, no caso de matrizes circulantes quadradas, cada linha ou coluna é a reformatação circulantes das outras linhas ou colunas. Caso não seja uma matriz quadrada, sua representação pode se dar em um conjunto de submatrizes quadradas, denominadas blocos.

Um cálculo básico em uma camada de uma rede neural densa é dado pela Equação (3.6):

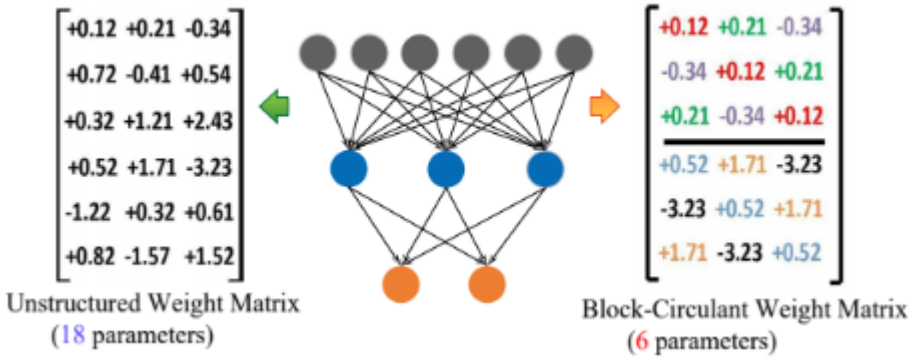
$$h(x) = \phi(Rx) \tag{3.6}$$

onde $R \in \Re^{k \times d}$, e $\phi(*)$ é uma função de ativação não-linear em um elemento. A operação conecta uma camada com d nós em uma camada com k nós. Nas CNN, as camadas densas são costumeiramente utilizadas antes da camada final, para capturar as propriedades globais de uma imagem. A complexidade destas operações são $O(dk)$ ou no m. Na prática, k costuma ser no mínimo $O(d^2)$. Isto cria um gargalo para muitas arquiteturas de rede. Porém Cheng et al. (2015) propõem uma estrutura circulante que permanece idêntica a da Equação (3.6), porém tendo R como uma matriz circulante.

Esta abordagem reduziu dramaticamente o número de parâmetros, e para acelerar os cálculos o autor utilizou a transformação rápida de Fourier, e como agora tanto a entrada como a saída das camadas possuem d nós, a complexidade de espaço foi reduzida para $O(d)$ e a complexidade de tempo foi reduzida para $O(d \log d)$, sem impactar de forma significativa na performance final da rede.

No trabalho de Ding et al. (2017), conforme a Figura 17, a utilização das matrizes circulantes reduziu em até 3x o tamanho da rede. Nesta figura, a matriz 6x3 da esquerda possui dezoito parâmetros, enquanto na matriz à direita utilizando duas matrizes circulantes 3x3 foi possível fazer a representação dos pesos com apenas seis parâmetros. De maneira intuitiva, a taxa de redução foi determinada pelo tamanho dos blocos das matrizes circulantes, logo quanto maior o bloco, maior a taxa de compressão.

Figura 17 – Matrizes bloco-circulantes para representação de pesos.



Fonte: Ding et al. (2017)

É importante compreender que o artigo de Ding et al. (2017) e sua rede CirCNN não pretende fazer uma conversão entre as matrizes não estruturadas e as bloco-circulantes. Ao invés disto, assume-se que a camadas podem ser representadas por matrizes bloco-circulantes e seu treinamento gera um subvetor para cada matriz. Algumas diferenças entre esta solução e outras são: enquanto outras abordagens utilizam outras técnicas como a poda nas matrizes não estruturadas e exigem um retreino completo, enquanto a CirCNN treina a rede diretamente com a estrutura bloco-circulante. Isto traz duas vantagens, redução de complexidade e fator de redução fixo da rede.

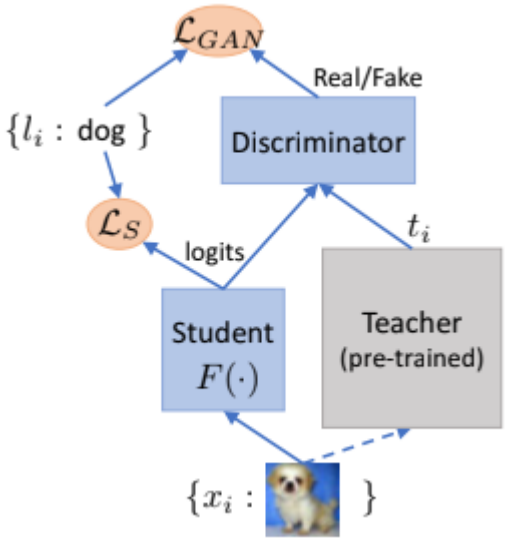
3.4 KNOWLEDGE DISTILLATION

Knowledge Distillation é uma abordagem baseada no princípio de treinar pequenas redes neurais aceleradas, tendo seu conhecimento repassado por uma rede

maior, ou rede professor. Buciluǎ, Caruana e Niculescu-Mizil (2006) foram os pioneiros nesta área e em seguida Caruana et al. (2004) treinaram redes rasas e amplas que aprenderam através de uma rede profunda não necessariamente projetada para ser acelerada.

Já o trabalho de Xu, Hsu e Huang (2018) propõe a utilização de redes adversárias condicionais para aprender a melhor função de perda para transferir o conhecimento da rede professor para a nova rede. O método proposto foi eficaz para gerar novas redes pequenas. Ambas as redes são CNNs baseadas na ResNet, ou seja, com arquitetura residual (que contém camadas de atalho), sendo que a nova rede é menor e rasa, mantendo a capacidade de inferir de maneira correta, porém mais rapidamente que a original.

Figura 18 – Treinamento da rede adversária proposta. O treino da rede professor é feito de modo *offline*. A nova rede (estudante) e o discriminador são atualizados alternadamente.



Fonte: Xu, Hsu e Huang (2018)

Conforme a Figura 18 mostra, ao invés de adotar a estratégia usual de forçar a nova rede a replicar o resultado da rede professor, o artigo vai por outro caminho: O conhecimento é transferido do professor através de um discriminador que é treinado para distinguir a quem pertence a saída da rede, se ao professor ou a nova rede. Isto tudo enquanto a rede é treinada para enganar o discriminador, ou seja, para que as saídas sejam indistinguíveis entre si, sem poder atribuir a qual rede ela foi gerada.

Isto traz algumas vantagens como a eficácia do aprendizado da função de perda, pois o discriminador fica responsável por ajustar estes valores, e a permanência da multi-modalidade, pois é difícil fazer a rede replicar os mesmos resultados da rede professor, porém um discriminador bem treinado pode capturar semelhanças entre as

categorias das saídas multimodais do professor, direcionando a nova rede a produzir saída corretas, ainda que não sejam as mesmas da original.

3.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

É possível perceber a existência de uma série de soluções voltadas para a aceleração de redes neurais, apesar disto, todas as soluções têm uma característica em comum, alterar ou remover a estrutura de uma rede a fim de torná-la mais simples. A listagem de alguns trabalhos citados e seus resultados podem ser vistos na Tabela 2, excluindo apenas os trabalhos clássicos também citados neste capítulo.

Tabela 2 – Trabalhos relacionados descritos no capítulo e seus resultados.

Autor	**Abordagem**	**Otimização** (%)	**Aceleração** (%)	**Acurácia** (%)
Ding et al. (2017)	Decomposição de camadas	-	18	-1
LeCun, Denker e Solla (1990)	Rede de poda	50	-	-
Li et al. (2016)	Rede de poda	64	-	-1
He et al. (2017)	Rede de poda	-	20	-
Ding et al. (2017)	Projeção Bloco-Circulante	14	-	96 (MNIST)
Xu, Hsu e Huang (2018)	*Knowledge Distillation*	28	-	-2,5

Fonte: Elaborado pelo autor. (2019)

No próximo capítulo, será descrito o projeto de mestrado que foi desenvolvido para otimizar os parâmetros de uma DenseNet, e que possui como diferencial realizar a otimização na fase de criação da rede, controlando a quantidade de mapas de características gerados a cada camada.

4 OTIMIZANDO PARÂMETROS DE UMA DENSENET ATRAVÉS DO CONTROLE DE GERAÇÃO DE MAPAS DE CARACTERÍSTICA

As redes neurais artificiais auxiliam na tarefa de reconhecimento de padrões há algumas décadas e estão cada vez mais presentes no dia-a-dia da população, atuando em diversos dispositivos. As Redes Neurais Convolucionais (CNN) sofrem ainda com a grande necessidade de poder computacional, restringindo muitas vezes a sua utilização em plataformas variadas. Métodos para melhorar a eficiência das Redes Neurais Convolucionais eficientes pode ter um impacto significativo em sistemas distribuídos, dispositivos embarcados ou móveis. Por exemplo, uma ResNet pode ser otimizada em até 75% de parâmetros utilizando alguns métodos de otimização existentes. (CHENG et al., 2015)

Conhecendo a necessidade de se otimizar os parâmetros de uma rede e no que isto implica, foi proposto um método de otimização feito para a DenseNet, uma rede neural convolucional que tem como característica ser completamente conectada, inclusive nas camadas convolucionais.

Dentre as razões que levaram à escolha desta arquitetura estão o conhecimento das principais vantagens da DenseNet em relação as outas redes, que são a redução do desaparecimento do gradiente (*Gradient Vanishing*), o fortalecimento da propagação das características, o incentivo a reutilização e a redução drástica de parâmetros.

A DenseNet requer aprender menos parâmetros que outras CNN pois não há a necessidade de aprender mapas de características redundantes. A arquitetura *feed-forward* podem ser vistas como um algoritmo com um estado, que é passado de camada à camada, gravando o estado anterior para a camada subsequente, alterando este estado e preservando informações. ResNets fazem isto de maneira explícita através das transformações aditivas de identidade. Algumas variações da ResNet descartam certas camadas durante o treinamento devido à pouca contribuição, similares as redes neurais recorrentes. Ainda assim, as ResNet possuem mais parâmetros devido ao fato de cada camada possuir os seus próprios pesos. A DenseNet faz uma diferenciação explícita do que é informação adicionada e preservada, e mantém as suas camadas com saídas pequenas (12 mapas gerados por camada, por exemplo), adicionando este pequeno grupo de novos mapas ao “conhecimento coletivo” existente. Ao final o classificador toma sua decisão baseado nos mapas gerados em toda a rede. Outra vantagem é o fluxo aprimorado de informações e gradientes pela rede, o que facilita o treinamento. Cada camada tem acesso ao gradiente pela função de perda e pelo sinal original, o que ajuda a treinar redes mais profundas, além disso,

as conexões densas geram um efeito regularizador, reduzindo o ajuste excessivo em conjuntos de treino menores. A concatenação de mapas de caraterística de diferentes camadas aumenta a variedade de características nas entradas e melhora a eficiência, sendo uma das principais diferenças entre a DenseNet e a ResNet, por exemplo. Já em comparação a arquitetura Inception, que concatena mapas de diferentes camadas, a DenseNet se mostra mais simples e eficiente, conforme Huang et al. (2017).

Revisitar a estrutura de uma DenseNet é necessária para a compreensão da técnica de otimização criada para ela. A DenseNet é uma rede que possui como característica o compartilhamento entre camadas dos mapas de características gerados, ou seja, enquanto nas CNN comuns uma camada convolucional se conecta apenas à camada a sua frente, aqui a camada se conecta com todas as camadas convolucionais anteriores e posteriores a ela, exatamente como uma rede densa funciona.

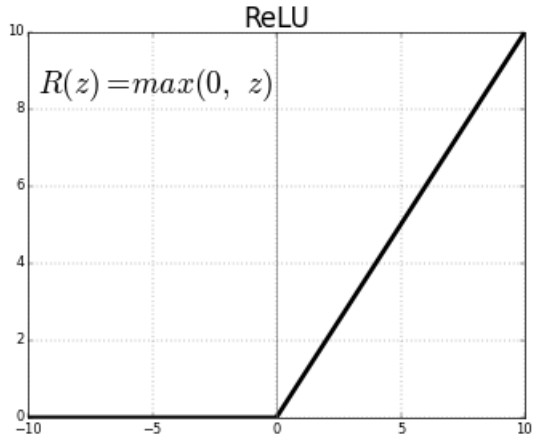
Cada camada convolucional gera um número fixo de mapas de características que é concatenado ao conjunto de mapas recebidos das camadas anteriores, fazendo com que a cada camada sejam criados novos mapas que exploram características mais profundas, evitando o recálculo e já diminuindo o número de parâmetros da rede. Este valor que define a quantidade de mapas de características gerados a cada camada é chamado de taxa de crescimento ou valor k.

Ao final de toda a estrutura da rede são realizadas novas operações de normalização, função de ativação e *Pooling* médio, em seguida os dados são enviados para uma camada densa que através da função de ativação *Softmax* retorna a distribuição das probabilidades entre as classes.

Ao executar o treinamento da rede, é possível perceber que a rede se torna menos sensível a atualizações conforme sua taxa de erro se aproxima de zero, limite mínimo da função de ativação ReLU, conforme projeção da Figura 19. Além disso a redução da resolução dos mapas de características durante a execução da rede diminui a variedade de características que podem ser representadas. Para resolver o problema da estagnação do treinamento conforme a redução do erro, é utilizada a técnica de diminuição programada da taxa de aprendizado de Robbins e Monro (1951), que modifica os valores de aprendizado conforme a rede sai da fase de exploração e entra na fase de intensificação. Isso demonstra que quanto mais próximo do melhor valor global, menos útil se torna o excesso de informação repassado à ela, pois se tornará pouco efetivo ou não utilizado. Esta redução ocorre com 50% e 75% do treinamento percorrido, e influencia diretamente na escolha do ponto inicial da otimização.

Esta foi a premissa para desenvolver a técnica de otimização da DenseNet, baseado em controle da geração de mapas de característica, pois o valor k permanece o mesmo durante toda a execução da rede, independente da fase, ou seja, mesmo na

Figura 19 – Gráfico da função de ativação ReLU. Conforme os valores se aproximam de zero, menos sensível o otimizador se torna.



Fonte: Sharma (2017)

fase de ajuste fino, a rede continua gerando o mesmo número de informações, independentemente da mesma surtir algum efeito relevante ou não no desempenho da rede. Esta geração linear de mapas de característica são apenas gera uma quantidade maior de mapas que podem não trazer alguma melhoria como aumenta a quantidade de parâmetros necessário para a rede, o que resulta em aumento de tempo de execução e necessidade de poder computacional.

Nas próximas seções o assunto será o desenvolvimento do método e sua aplicação. Na Seção 4.1 será discutido como encontrar o fator de decréscimo e sua aplicação e no Capítulo 5 serão expostos os testes originais realizados pelo autor, quais conjuntos de dados utilizados e quais deles serão realizados neste projeto.

4.1 DEFININDO O FATOR DE DECRÉSCIMO

Esta dissertação propõe o controle da geração dos mapas de características conforme o momento em que a rede se encontra. Este controle se dá no momento em que a rede é criada e a partir de um determinado momento, a rede começa a receber de maneira progressiva um número menor, porém ainda contínuo, de mapas de característica, permitindo que os mapas gerados sejam de fato utilizados para o aprendizado da rede, com um menor desperdício.

O que torna este controle possível é a adição de um novo parâmetro chamado de controle de decréscimo (*Decrease Control* - dc) ou valor dc e tomando como parâmetro a orientação de Huang et al. (2017) em iniciar a redução da taxa de aprendizado a partir da metade do treinamento da rede, o controle de decréscimo também iniciará a partir da metade da estrutura da rede, ou seja, em uma rede com $d = 40$, o controle

geracional de mapas se dará a partir da 18º camada densa, utilizando a Equação (4.1):

$$id = \frac{l \cdot b}{2} \tag{4.1}$$

Onde l é a quantidade de camadas densas após a aplicação da Equação (**??**), e b é o número de blocos densos. Iniciar o decréscimo de diferentes pontos da rede resultará em diferentes totais de otimização, assim como redes maiores tendem a otimizarem um número maior de parâmetros.

Uma vez obtido o valor id, é possível descobrir o fator de decréscimo aplicando a Equação (4.2):

$$dc = \left\lfloor \frac{k}{id} \right\rfloor \tag{4.2}$$

Assim, a partir da camada densa id, o valor de k é reduzido uma unidade a cada dc camadas. Para descobrir o valor total de parâmetros em uma camada convolucional, utiliza-se a Equação (2.5), relembrando sua formulação que é:

$$p = ((i \cdot m \cdot n) + 1) \cdot o \tag{4.3}$$

Sendo que o é equivalente ao valor k, podemos definir a quantidade de parâmetros otimizados calculando a quantidade de parâmetros originais e subtraindo pela quantidade gerada pelo modelo otimizado, isso é possível através da soma dos parâmetros de cada bloco denso e das camadas de transição separadamente, como será exemplificado a seguir.

Como originalmente a DenseNet gera sempre o mesmo número de saídas das camadas densas, concatenando-as, é possível inferir que o crescimento dos mapas de características se dá em fator de k, concatenando esta saída l vezes, ou seja, a quantidade de camadas densas existentes dentro de um bloco denso. Para descobrir quantos parâmetros há em uma DenseNet original basta aplicar a Equação (4.4) e a Equação (4.5):

$$pd = \sum_{i=1}^{l} 4c + (9c \cdot k \cdot z) \tag{4.4}$$

$$pt = \sum_{i=1}^{l} 4c + (c \cdot k \cdot z) \tag{4.5}$$

Sendo c a quantidade de canais de entrada, k o fator de crescimento e i o número da camada, que multiplicados retornam o valor de mapas recebidos como entrada naquele momento. O valor z é referente ao fator de compressão para a camada de transição, caso este valor seja diferente de 1. A primeira operação realizada antes de adição é referente à normalização em lote, enquanto a segunda opção calcula os parâmetros de uma convolução. A soma deles dá a quantidade de parâmetros em uma camada densa, enquanto que a soma dos parâmetros de todas as camadas densas dá a quantidade de parâmetros de um bloco inteiro.

Tendo os valores dos blocos, basta multiplicar a quantidade de blocos densos e de transição existentes para descobrir a quantidade de parâmetros no modelo original, quantidade está descrita na Equação (4.6) por b e t, respectivamente.

$$td = (pd \cdot b) + (pt + t) \tag{4.6}$$

Já para descobrir a quantidade de parâmetros em uma DenseNet otimizada, basta fazer uma pequena variação na Equação (4.4), reduzindo o valor de k em uma unidade a cada dc camadas a partir de id.

Por fim, basta somar os valores dos blocos densos e de transição separadamente para obter o valor total de parâmetros do modelo otimizado, uma vez que agora o valor k varia e a multiplicação não é mais um método eficaz de acelar o cálculo dos parâmetros. Podemos mostrar, por exemplo, um modelo de três blocos densos e duas transições que pode ser descrito na Equação (4.7):

$$td_o = p_{b1} + p_{t1} + p_{b2} + p_{t2} + p_{b3} \tag{4.7}$$

Assim, tendo os dois totais de parâmetros dos modelos, basta realizar uma subtração simples para descobrir o valor total de parâmetros otimizados pelo controle de geração de mapas de características, vale lembrar que quanto maior o modelo, maior é o valor de otimização alcançado.

Mas existe a possibilidade de que o resultado de (4.2) seja um número negativo, porém isto não é desejável, pois o menor número de mapas de características a ser gerado cada vez é um, pois número menores implicam em uma estagnação na rede.

Contornar este problema exige que se descubra um valor mínimo que multiplicado pelo dc dê um resultado próximo e maior que um, sendo arredondado para baixo logo em seguida.

Assim, o resultado desta operação torna-se o valor dc' e neste caso, ao invés do decréscimo ocorrer a cada camada reduzindo dc unidades, o decréscimo passa a reduzir apenas uma unidade a cada dc' camadas, mantendo a mesma proporcionalidade do fator de decréscimo para resultados positivos ou negativos na Equação (4.2).

Para fim de compreensão, a Figura 20 iniciará a ilustração do comportamento da técnica, apresentando a arquitetura de uma rede DenseNet ilustrativa de apenas 20 camadas, ou seja, $d = 20$ distribuídas entre três blocos densos, também podemos ver as camadas densas dentro de cada bloco denso. Calculando conforme a Equação (**??**), o valor de camadas densas por bloco denso é de 5 camadas.

Aplicando as Equações (4.4) e (4.5), uma rede DenseNet comum com estas configurações e uma taxa de crescimento de valor $k = 12$ possuiria 63.456 parâmetros no total. Porém calculando com a Equação (4.1) o início do decréscimo a partir da sétima camada, ou seja, a metade das camadas densas, e definindo o valor dc a partir da Equação (4.2) que neste caso é 1, a rede otimizada passa a ter 43.556 parâmetros, uma economia de $31,3\%$, conforme mostra Figura 21.

Outro exemplo pode ser acompanhado pelo gráfico da Figura 22 que mostra o crescimento do número de mapas de característica por bloco denso, considerando o exemplo otimizado já citado. O gráfico demonstra o decréscimo dos mapas de característica, sendo o eixo horizontal as camadas densas e o eixo vertical a quantidade de mapas conforme ocorre a redução de k.

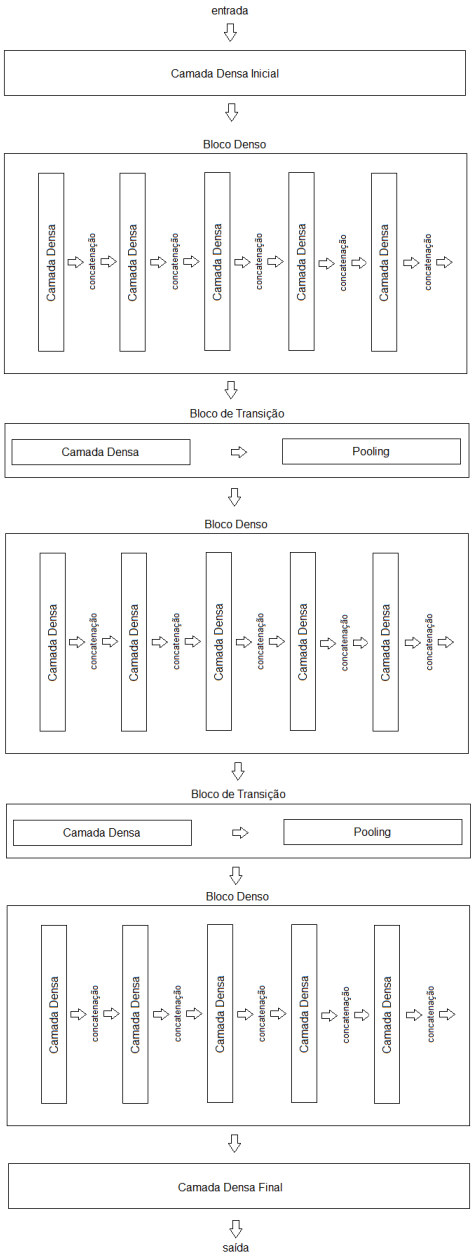
Para completo entendimento das Figuras 21 e 22, as mesmas retratam a DenseNet exemplificada nesta seção, contendo 20 camadas densas divididas em três blocos densos, onde é possível ver a divisão das camadas e como ocorre o decréscimo da taxa de crescimento após o encontro do valor dc.

4.2 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi discutido o desenvolvimento da técnica de otimização de parâmetros para a DenseNet, baseada no controle de geração de mapas de características. Esta técnica baseou seu funcionamento no comportamento dos otimizadores e da técnica de redução da taxa de aprendizado durante a fase final do treinamento da rede.

Gerenciar a criação dos mapas de característica permite que um modelo que já possui em sua natureza um número de parâmetros reduzido possa ser ainda mais compactado mantendo de forma aproximada, por vezes igual, os valores de acurácia dos experimentos realizados, sendo a descrição da técnica uma das contribuições deste capítulo.

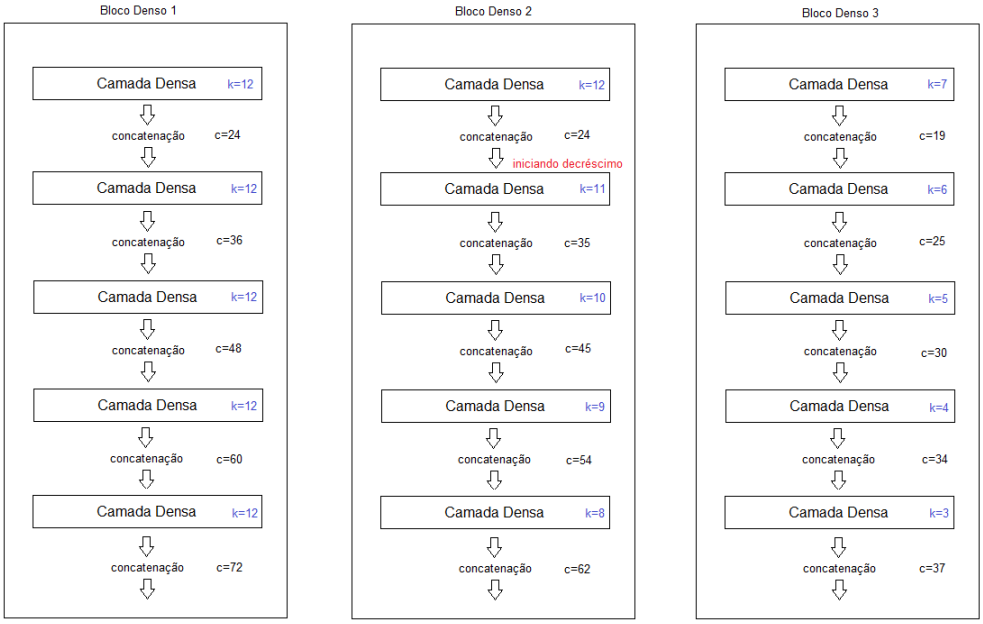
Figura 20 – Exemplo da arquitetura de uma DenseNet com profundidade $d = 20$.



Fonte: Elaborado pelo autor. (2019)

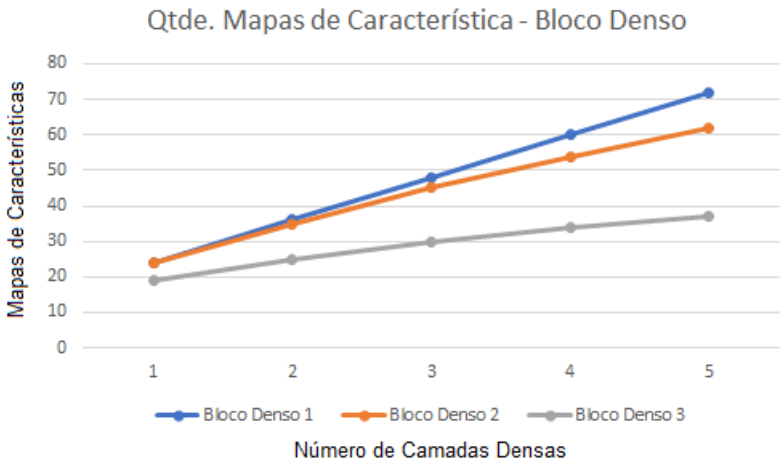
No capítulo 5, os resultados dos experimentos serão demonstrados em diversos conjuntos de dados utilizados, além de informações de como os testes foram conduzidos, suas configurações e parâmetros para que se possa validar a técnica descrita neste capítulo.

Figura 21 – Exemplo da redução de mapas de características e seus efeitos na quantidade total de mapas gerados em cada bloco denso.



Fonte: Elaborado pelo autor. (2019)

Figura 22 – Decréscimo dos mapas de característica.



Fonte: Elaborado pelo autor. (2019)

5 RESULTADOS DOS EXPERIMENTOS

Neste capítulo serão apresentados e discutidos os experimentos realizados para verificar o funcionamento do modelo proposto no projeto. O modelo conseguiu cumprir com êxito a tarefa de otimizar diversos indicadores da DenseNet original, como tamanho em disco, tempo de treinamento e quantidade de parâmetros. Ainda conseguiu manter a proximidade dos níveis de acurácia alcançados pelo modelo original. Como exceção dos experimentos realizados, o conjunto SHVN, um conjunto para identificar números de casas, não foi utilizado neste projeto.

Para comparar as taxas de reconhecimento em uma única base de dados, foi utilizado o teste de Dietterich (1998), o qual baseia-se nas diferenças entre a taxa de erro entre o método A (modelo original) e método B (proposta deste trabalho). Assume-se que a taxa de erro (proporção de exemplos que não foram reconhecidos) corresponde à probabilidade de erro do método em questão. Assim, uma distribuição normal padrão (aproximada) pode ser obtida por meio da Equação 5.1, onde pA é a probabilidade de erro do método A, pB é a probabilidade de erro do método B, p é a média das duas probabilidades de erro (Equação 5.2) e n é a quantidade de exemplos de teste. Como os resultados obtidos pelo modelo original foram extraídos dos artigos, impossibilitou a realização de comparação das mesmas bases de teste (mesmas imagens). Por este motivo, foi utilizado o teste de Dietterich e não testes de McNemar (1947) ou Wilcoxon (1992) conforme recomendado por Demšar (2006).

$$z = \frac{pA - pB}{\sqrt{\frac{2p(1-p)}{n}}} \tag{5.1}$$

$$z = \frac{pA + pB}{2} \tag{5.2}$$

A hipótese nula (H_0: diferença entre classificadores é irrelevante) é rejeitada se $|z| > z_{0,95} = 1,96$ (para um teste de dois lados com a probabilidade de rejeitar incorretamente a hipótese nula de $0,05$). O teste para diferenças entre proporções foi, então, realizado para avaliar os resultados obtidos pelo método original e o método aqui proposto nas três bases de dados testadas. O objetivo é garantir que não hajam diferenças significativas nos resultados (na acurária) de ambos os métodos. Observou-se nos experimentos realizados que não houve diferença significativa na acurácia entre a rede criada pelo método original e a rede criada pelo método aqui

Tabela 3 – Resultado do teste estatístico de Dietterich para cada conjunto de dados. O modelo pode ser considerado estatisticamente válido caso alcance um resultado menor que 1,96.

Base de Dados	**Configuração**	**Resultado**
CIFAR-10	$k = 12$, $d = 40$	1,84
CIFAR-10	$k = 12$, $d = 100$	1,16
CIFAR-10	$k = 24$, $d = 100$	1,90
CIFAR-100	$k = 12$, $d = 40$	*4,55*
CIFAR-100	$k = 12$, $d = 100$	1,94
CIFAR-100	$k = 24$, $d = 100$	1,94
TinyImageNet	$k = 12$, $d = 100$	1,95
Cats vs Dogs	$k = 12$, $d = 40$	1,75
MNIST	$k = 12$, $d = 40$	**0,32**
MNIST	$k = 12$, $d = 100$	0,71
Fashion-MNIST	$k = 12$, $d = 40$	1,92
Fashion-MNIST	$k = 12$, $d = 100$	1,95
Caltech-101	$k = 12$, $d = 40$	1,70

Fonte: Elaborado pelo autor. (2019)

proposto. Os resultados dos testes estatísticos podem ser observados na Tabela 3, com o melhor resultado destacado em negrito e o pior resultado destacado em itálico.

As versões da DenseNet utilizadas para os experimentos no conjunto CIFAR-10 e no conjunto CIFAR-100 foram obtidas no repositório GitHub através do desenvolvedor Somshubra Majumdar[1]. Como originalmente algumas das informações mostradas neste projeto não estavam disponíveis no artigo original, foram feitos protótipos dos modelos originais para realizar algumas das comparações, com e sem a aplicação da técnica. Já para o conjunto TinyImageNet foi utilizada a versão da DenseNet criada pela desenvolvedora da ferramenta Keras, sendo esta versão uma DenseNet-BC com as particularidades de desenvolvimento voltadas para esta base de dados, e portanto não aplicável para os outros conjuntos deste projeto.

5.1 CONJUNTOS DE DADOS UTILIZADOS NOS EXPERIMENTOS

Nesta seção serão apresentados os conjuntos de dados utilizados para a experimentação do trabalho desenvolvido, estes conjuntos são utilizados como métricas para reconhecimento de padrões e alguns destes já estavam previstos no artigo original de Huang et al. (2017), como o CIFAR-10, CIFAR-100 e ImageNet. Destas, os conjuntos CIFAR-10 e CIFAR-100 permaneceram para a realização dos experimentos, já a base ImageNet foi substituída pela base TinyImageNet devido a falta de disponibilidade.

As outras bases escolhidas para os experimentos foram a base de dígitos manuscritos MNIST, a base de imagens de roupas Fashion-MNIST, a base de cães e gatos *Cats vs Dogs* e a base de múltiplas classes CALTECH-101. As particularidades

[1] https://github.com/titu1994/DenseNet

de cada conjunto serão descritas nesta seção.

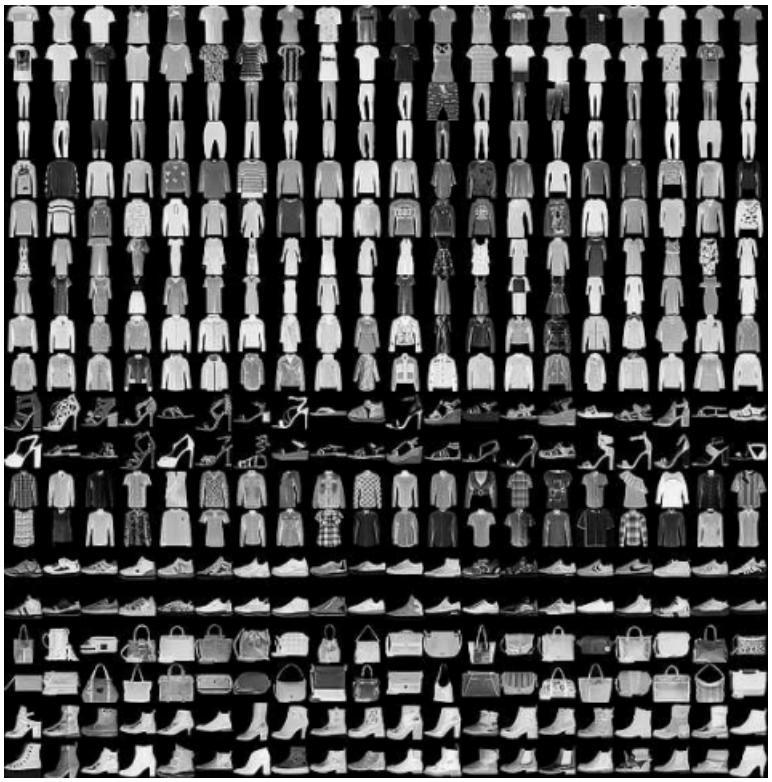
Segundo Deng (2012), as base MNIST (*Modified NIST*) feita por LeCun et al. (1998) é um conjunto de dados de dígitos manuscritos extraídos de uma base maior, a NIST. Ela contém 60000 imagens de treino, podendo utilizar algumas destas imagens para uma validação cruzada, e 10000 imagens de teste, distribuídas de maneira igual com cerca de 6000 imagens por classe, contendo 10 classes. Todos os dígitos são imagens binárias normalizadas, e centralizadas, possuindo um tamanho fixo de 28x28 pixels. É um conjunto relativamente simples quando se quer executar técnicas de reconhecimento de padrões com dados reais utilizando um esforço mínimo de formatação e processamento. Exemplos do conjunto podem ser visto na Figura 23.

Figura 23 – Exemplos do conjunto de dados MNIST.

Fonte: Chmielnicki e Stąpor (2010)

Também como uma variação do MNIST, o trabalho de Xiao, Rasul e Vollgraf (2017) traz um conjunto de dados de imagens binárias que trazem o desafio de classificar 10 classes de peças de roupas. Este conjunto possui a mesma distribuição do MNIST, com 60000 imagens para treino e validação e 10000 para teste. Também são iguais as formatações e tamanhos das imagens, como é possível ver na Figura 24.

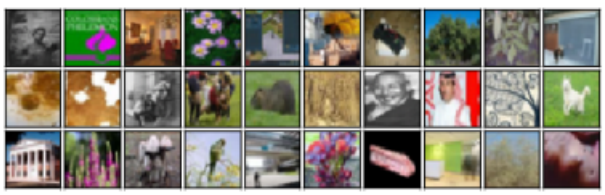
Figura 24 – Exemplos do conjunto de dados Fashion-MNIST.



Fonte: Chmielnicki e Stąpor (2010)

Criado por Krizhevsky e Hinton (2010), o conjunto CIFAR-10 consiste de 60000 imagens de tamanho 32x32 *pixels*, contendo 50000 imagens para treino e validação e 10000 imagens para teste, sendo um total de dez classes para reconhecer, alguns exemplos estão apresentados na Figura 25. Este é um conjunto balanceado com 6000 imagens por classe para o CIFAR-10 e 600 imagens por classe para o CIFAR-100. As classes são mutuamente exclusivas, ou seja, não existe uma sobreposição de classes entre as imagens. O CIFAR-10 é uma subamostra de um conjunto maior, o CIFAR-100. O CIFAR-100, que possui no total 100 classes, também desenvolvido por Krizhevsky, Hinton et al. (2009), possui as mesmas disitribuições de imagem para treino, validação e teste.

Figura 25 – Exemplos do conjunto de dados CIFAR.



Fonte: Krizhevsky e Hinton (2010)

Criado por Fei-Fei, Fergus e Perona (2004), o CALTECH-101 é um dataset diferenciado dos outros utilizados neste projeto, pois as classes são desbalanceadas, podendo possuir entre 40 e 800 imagens em cada uma das 101 categorias, além disso, as imagens são coloridas e não possuem tamanho fixo, e sim um tamanho aproximado de 300x200 *pixels*.

Figura 26 – Exemplos do conjunto de dados CALTECH-101.

Fonte: Fei-Fei, Fergus e Perona (2004)

Desenvolvido pela Microsoft para ser um CAPTCHA que previne a utilização de robôs online, a base ASIRRA(*Animal Species Image Recognition for Restricting Access*) se mostrou um desafio na publicação de Elson et al. (2007). Esta base balanceada possui apenas duas classes, cães e gatos, e possui 25000 imagens que podem ser divididas nas bases de treino, validação e teste. Ainda existe uma base de teste não anotada com 12500 imagens, utilizada para a competição oficial.

Por último, o conjunto Tiny ImageNet, um desafio criado por Le e Yang (2015). Este conjunto é balanceado e possui 200 classes extraídas do conjunto ImageNet, onde cada classe possui 500 imagens para treino, 50 para validação e 50 para teste, as imagens são disponibilizadas no tamanho de 64x64 *pixels*.

5.2 PROTOCOLO EXPERIMENTAL

Para a execução do experimento é necessário conhecer o protocolo experimental utilizado por Huang et al. (2017) em seu trabalho original. Com exceção do conjunto TinyImageNet, todos os outros conjuntos seguirão as mesmas particularidades descritas originalmente para os conjuntos CIFAR-10 e CIFAR-100.

Para a execução dos testes foram utilizados as mesmas divisões originais de treino, validação e testes originais de cada conjunto. Caso não exista um conjunto de

validação original, utiliza-se 10% do conjunto de teste. O método de validação utilizada é a validação cruzada. Cada experimento foi executado quatro vezes, e o resultado é a média das execuções realizadas.

Para os conjuntos CIFAR-10, CIFAR-100, MNIST, Fashion-MNIST,Cats Vs Dogs e Caltech-101 foram aplicadas as mesmas regras de experimentação determinadas por Huang et al. (2017) para os conjuntos CIFAR-10 e CIFAR-100, que são:

- Divisão balanceada das camadas densas através do valor d, neste caso, d possui os valores 40 e 100;
- Quantidade de blocos densos: 3;
- Utilizar valores pequenos de k, neste caso os valores são 12 e 24;
- Épocas de treinamento: 300 épocas;
- Otimizador: SGD;
- Momentum/Nesterov definido em $0,9$, *decay* definido em $0,0001$;
- Taxa de aprendizado: $0,1$, reduzindo para $0,01$ com 50% do treinamento percorrido e $0,001$ com 75% do treinamento percorrido;
- Tamanho do lote: 64;
- Taxa de compressão: 1;
- Taxa de *dropout*: $0,2$;

Para o conjunto CIFAR-10, CIFAR-100, é realizado um pré-processamento, normalizando as entradas utilizando a média e o desvio padrão das amostras, para os outros conjuntos, é realizada uma normalização simples;

Por ser uma derivação do conjunto ImageNet, o TinyImageNet utilizou as configurações determinadas por Huang et al. (2017) para este conjunto específico, utilizando o resultado Top-1, ou seja, levando em conta apenas o primeiro resultado para calcular o erro. A configuração para o conjunto TinyImageNet foi:

- Divisão específica das 4 camadas densas: $[6, 12, 24, 16]$;
- Utilizar o valor de k como 32;
- Épocas de treinamento: 90 épocas;
- Otimizador: SGD;

- Momentum/Nesterov definido em $0,9$, *decay* definido em $0,0001$;
- Taxa de aprendizado: $0,1$, reduzindo para $0,01$ com 50% do treinamento percorrido e $0,001$ com 75% do treinamento percorrido;
- Tamanho do lote: 256;
- Taxa de compressão: 0.5;
- Taxa de *dropout*: $0,2$;

5.3 RESULTADOS DA BASE MNIST

Iniciando esta seção, podemos ver na Tabela 4 os resultados alcançados em outras redes classificadores para esta base. É necessário ressaltar que estes resultados da Tabela 4 e das outras tabelas que apresentam resultados de outras redes foram obtidos através dos artigos originais, portanto, informações como número de parâmetros podem não estar disponíveis pois o artigo original daquela rede não disponibilizou tal informação ou não contemplou determinada base de dados.

Tabela 4 – Resultados de outras redes - MNIST.

Rede	**Acurácia (%)**
Network in Network	99,5
Deeply Supervised Net	99.6

Fonte: Elaborado pelo autor. (2019)

Seguindo os resultados das Tabelas 5 e 6, pode-se perceber na configuração $dk = 12, d = 40$ uma otimização de 32% dos parâmetros, além de reduzir o espaço de armazenamento em 27%. Também houve uma aceleração de 11% com uma queda de 0,04% na acurácia no conjunto de teste.

Tabela 5 – Dados dos modelos originais - MNIST.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (GB)**
$k = 12$, $d = 40$	429,118	63	5,25	4,05	99,25	0,168
$k = 12$, $d = 100$	2.863.198	247	20,5	16,54	99,41	0,985

Fonte: Elaborado pelo autor. (2019)

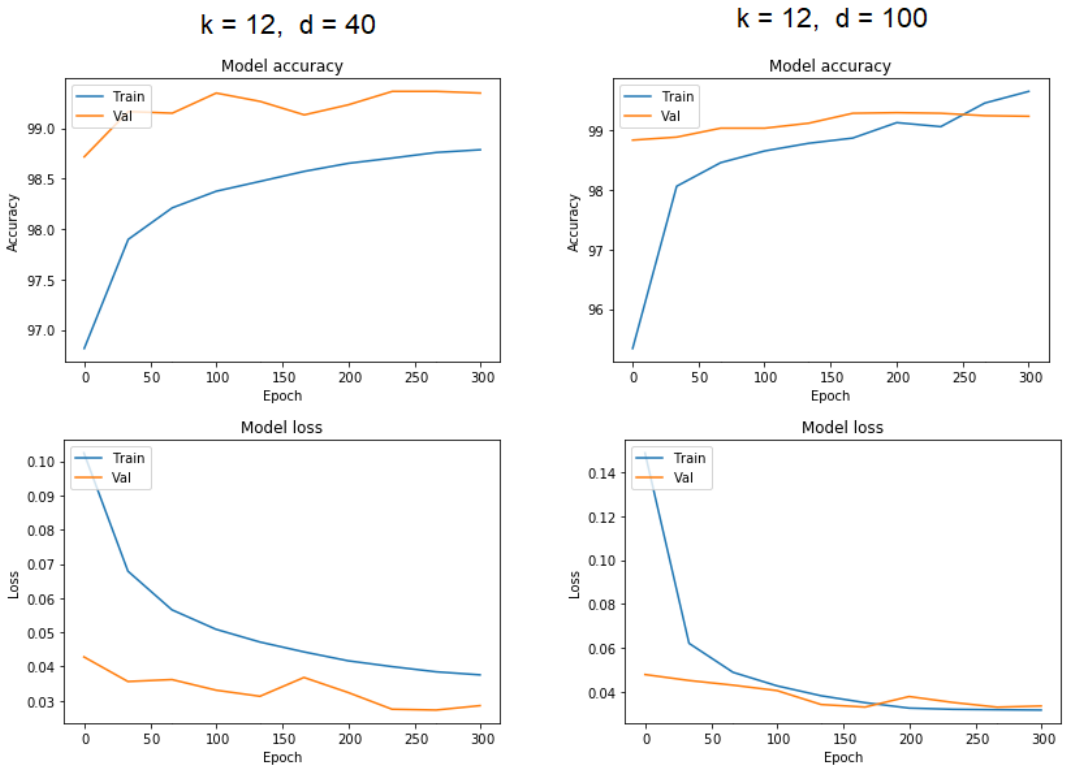
Os resultados obtidos pela configuração $k = 12, d = 100$ foram a otimização de 43% dos parâmetros da rede e reduziu seu espaço em 13%. O treinamento foi acelerado em 14% e houve uma queda de 0,08% na acurácia em comparação ao original. O gráfico do treinamento da rede pode ser visto na Figura 27.

Tabela 6 – Dados dos experimentos efetuados - MNIST.

Configuração	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (GB)
$k = 12, d = 40$	290.203	56	4,4	2,94	99,21	0,165
$k = 12, d = 100$	1.608.104	210	17,5	14,36	99,33	0,953

Fonte: Elaborado pelo autor. (2019)

Figura 27 – Gráficos de acurácia e perda dos experimentos realizados com o conjunto de dados MNIST.



Fonte: Elaborado pelo autor. (2019)

5.4 RESULTADOS DA BASE FASHION-MNIST

Fazendo uma comparação entre o modelo original e o otimizado, segundo os resultados das Tabelas 7 e 6, é possível perceber na configuração $dk = 12, d = 40$ uma otimização de 32%, e uma redução do espaço de armazenamento em 27%. Também ocorreu uma aceleração de 27% com uma queda de 0.65% na acurácia no conjunto de teste.

Já os resultados obtidos pela configuração $k = 12, d = 100$ alcançaram uma otimização de 43% dos parâmetros da rede e conseguir reduzir seu espaço de armazenamento em 13%. O treinamento foi acelerado em 14% e obteve uma queda de 0.62% na acurácia em comparação ao original. Os gráficos de treinamento estão na

Tabela 7 – Dados dos modelos originais - Fashion-MNIST.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 12, d = 40$	429.118	63	5,2	4,03	94,83	0,168
$k = 12, d = 100$	2.863.198	247	20,5	16,51	95,35	0,985

Fonte: Elaborado pelo autor. (2019)

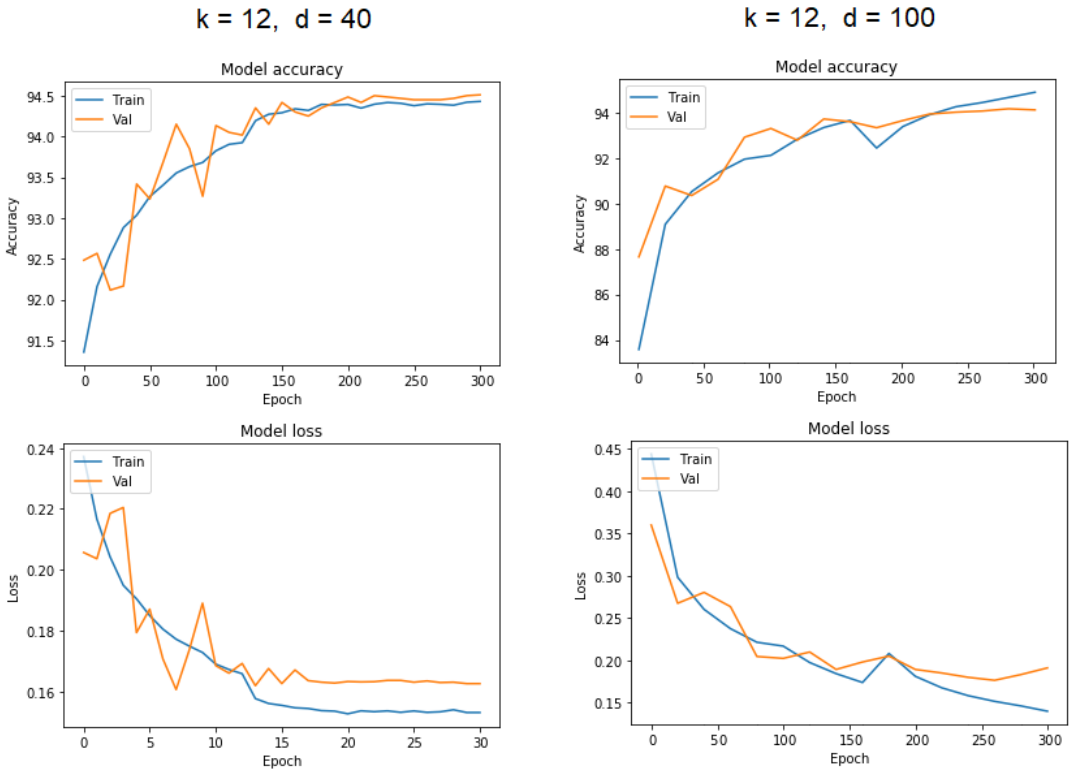
Figura 28.

Tabela 8 – Dados dos experimentos efetuados - Fashion-MNIST.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 12, d = 40$	290.203	56	4,4	2,96	94,21	0,165
$k = 12, d = 100$	1.608.104	206	17.1	14,35	94,75	0,953

Fonte: Elaborado pelo autor. (2019)

Figura 28 – Gráficos de acurácia e perda dos experimentos realizados com o conjunto de dados Fashion-MNIST.



Fonte: Elaborado pelo autor. (2019)

5.5 RESULTADOS DA BASE CIFAR-10

Já na base CIFAR-10, podemos ver na Tabela 9 os resultados alcançados em outras redes classificadoras para esta base.

Tabela 9 – Resultados de outras redes - CIFAR-10.

Rede	**Acurácia (%)**
Network in Network	91,19
Deeply Supervised Net	91,78
Highway Network	92,24
FractalNet	95.32
ResNet	91.20

Fonte: Elaborado pelo autor. (2019)

Segundo as Tabelas 10 e 11, que mostram as diferenças entre os modelos otimizados e os originais, a rede que utilizou como configuração de profundidade $d = 40$ e taxa de crescimento $k = 12$ conseguiu otimizar 33% dos seus parâmetros, e reduziu seu espaço de armazenamento em 37,5%. Além disso, este modelo foi cerca 28% mais veloz no tempo de treinamento e teve uma queda de 2,01% do valor final de acurácia no conjunto de teste.

Tabela 10 – Dados dos modelos originais - CIFAR-10.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (GB)**
$k = 12$, $d = 40$	434.014	84	7	4	94,7	0,168
$k = 12$, $d = 100$	2.875.294	276	23	23	95,9	0,985
$k = 24$, $d = 100$	11.253.394	597	50	87	96,2	1,998

Fonte: Elaborado pelo autor. (2019)

Já o modelo que utilizou as configurações $d = 100$ e $k = 12$ obteve uma melhora um pouco mais sensível, otimizando 44% dos parâmetros e 39% do tamanho, conseguindo também 23% de melhora no tempo de treinamento e tendo como acurácia uma diferença ainda menor de 1,1%.

Tabela 11 – Dados dos experimentos efetuados - CIFAR-10.

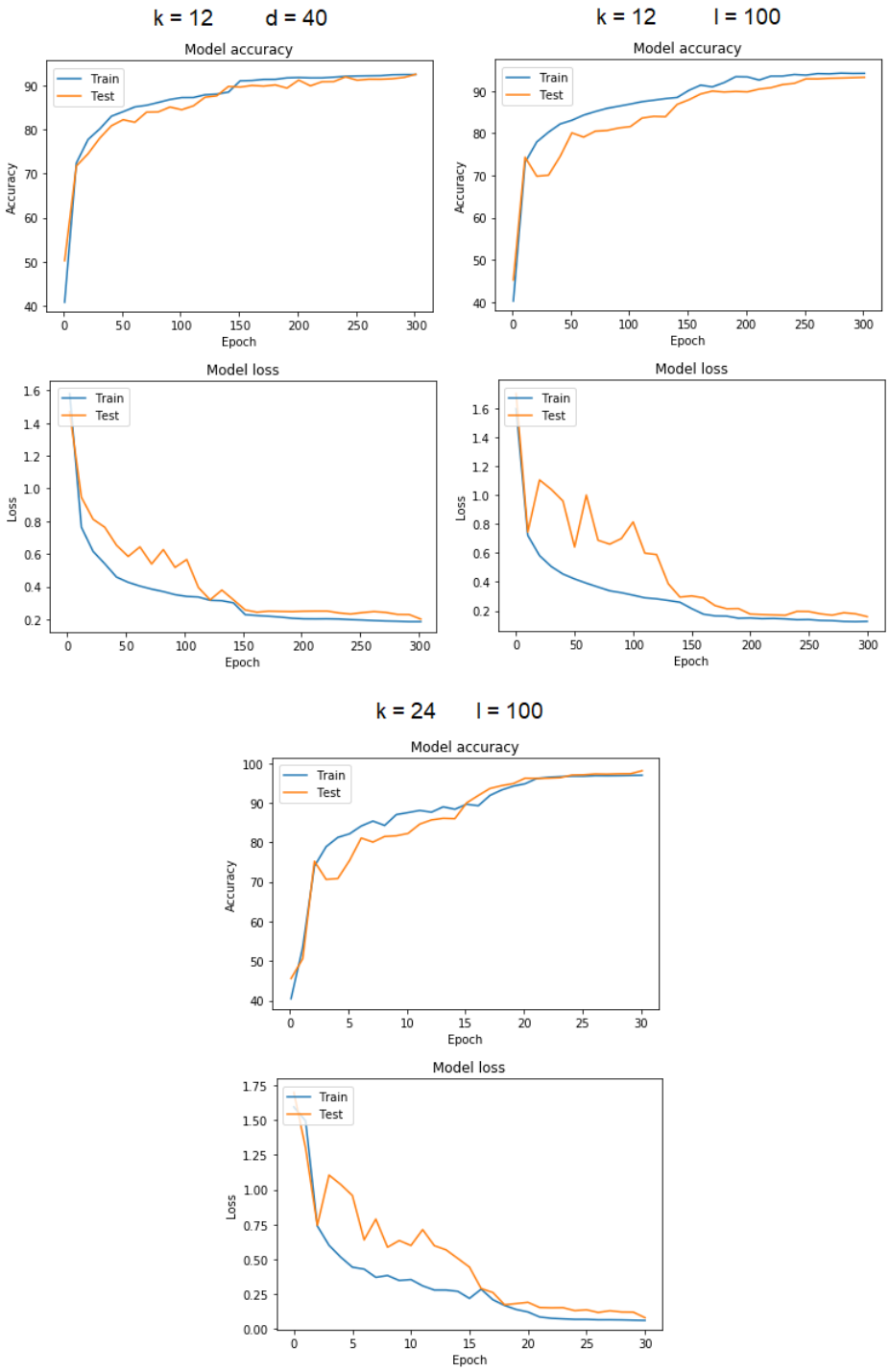
Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (GB)**
$k = 12$, $d = 40$	292.399	60	5	2,5	92,69	0,165
$k = 12$, $d = 100$	1.610.090	210	17,5	14	94,80	0,953
$k = 24$, $d = 100$	6.558.169	360	30	52,5	94,40	1,921

Fonte: Elaborado pelo autor. (2019)

Por fim a última configuração que utiliza $d = 100$ e $k = 24$ alcançou uma otimização de 42% dos parâmetros, além de 39% redução de tamanho e 39% de

melhora no tempo de treinamento, e concluindo com uma diferença de apenas 1,8% do modelo original com a mesma configuração. Nestes experimentos houveram uma redução média de 40,2% no tamanho da rede e uma redução média de 30% do tempo de treinamento.

Figura 29 – Gráficos de acurácia e perda dos experimentos realizados com o conjunto de dados CIFAR-10.



Fonte: Elaborado pelo autor. (2019)

Também pode se ver na Figura 29 os gráficos de acurácia e perda dos treinamentos realizados.

5.6 RESULTADOS DA BASE CIFAR-100

Iniciando esta seção, podemos ver os resultados da base CIFAR-100 na Tabela 12, quando aplicadas em outras redes classificadoras.

Tabela 12 – Resultados de outras redes - MNIST.

Rede	**Acurácia (%)**
Network in Network	64,32
Deeply Supervised Net	65.43
ResNet	77,51

Fonte: Elaborado pelo autor. (2019)

Os experimentos utilizando o CIFAR-100 utilizaram as mesmas configurações dos experimentos do conjunto CIFAR-10, com a diferença da quantidade de classe a serem classificadas neste conjunto. Sabendo disto, pode se perceber nas Tabelas 13 e 18 os valores originais e os resultados alcançados após a execução da otimização da rede.

Tabela 13 – Dados dos modelos originais - CIFAR-100.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 12$, $d = 40$	490.264	181	15	2,5	75,5	0,168
$k = 12$, $d = 100$	3.017.944	364	30	13,5	79,8	0,985
$k = 24$, $d = 100$	11.538.604	612	51	45,5	80,7	1,998

Fonte: Elaborado pelo autor. (2019)

Com a configuração $d = 40$ e $k = 12$ foi possível obter 35% de otimização de parâmetros, além de uma aceleração de 8% e uma redução de 28% do tamanho do modelo. Por fim, esta configuração alcançou uma acurácia de 66,24%, uma diferença de 9,26%. Esta configuração tem um tamanho aproximado ao da rede utilizada na base CIFAR-10, com as mesmas configurações, porém com dez vezes mais classes para classificar. Acredita-se que a otimização, neste caso, teve o efeito contrário do esperado, com uma diferença de acurácia maior do que a alcançada pelos outros modelos, ou seja, a rede se tornou pequena demais para o problema proposto no experimento, por isso, utilizar a rede otimizada nesta configuração e para este problema não é recomendado.

Já os resultados do modelo $d = 100$ e $k = 12$ alcançaram 46% de otimização de parâmetros e uma aceleração no treinamento de 30,7%, além disso, o tamanho

Tabela 14 – Dados dos experimentos efetuados - CIFAR-100.

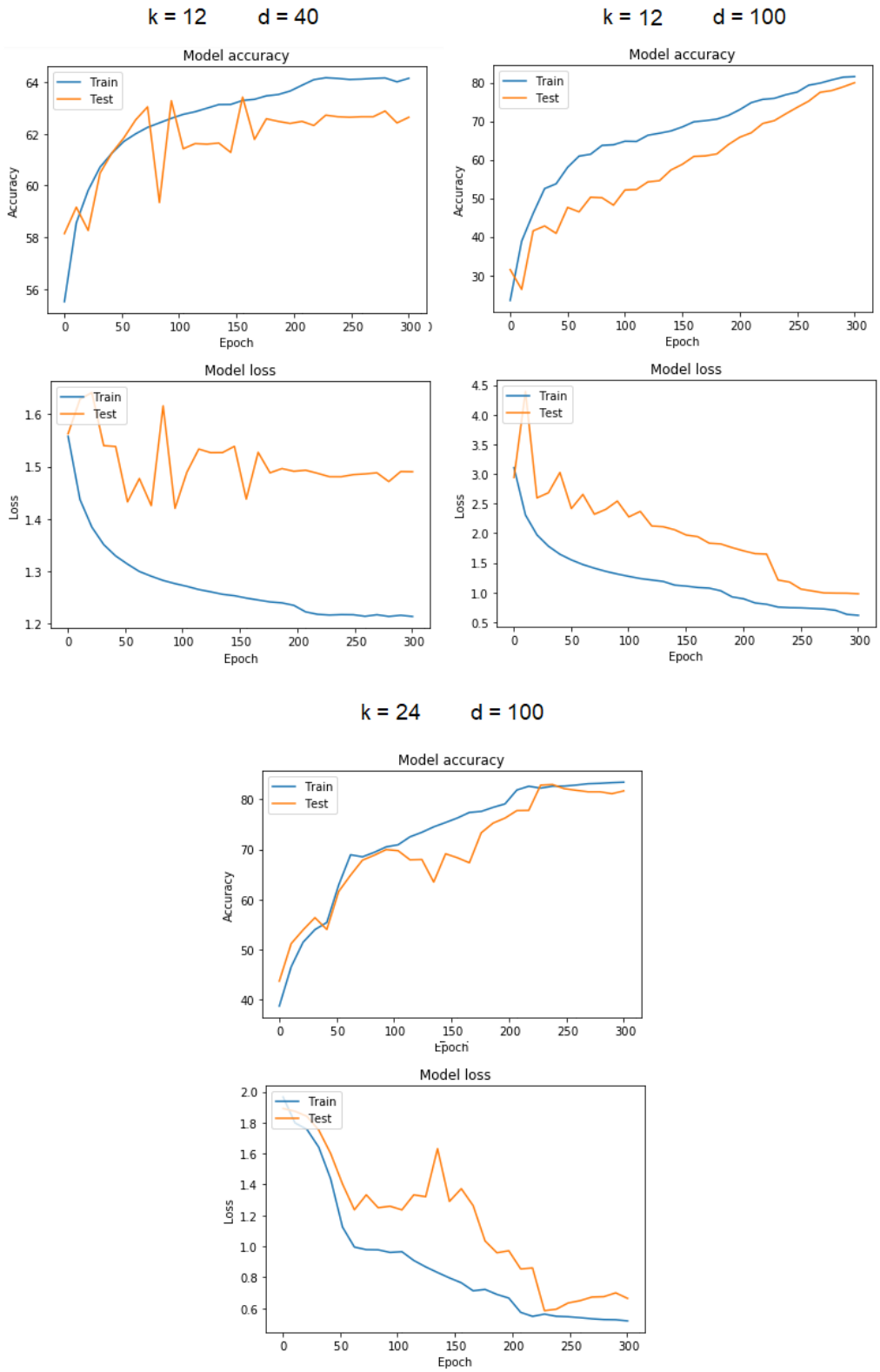
Configuração	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (MB)
$k = 12$, $d = 40$	316.249	167	14	1,8	66,24	0,165
$k = 12$, $d = 100$	1.631.420	252	21	8	76,20	0,953
$k = 24$, $d = 100$	6.617.299	380	31	27,9	77,15	1,921

Fonte: Elaborado pelo autor. (2019)

do modelo foi reduzido em 40,7% e sua acurácia ficou a uma distância de 3,6%, com 76,2% no total.

A última configuração deste conjunto é a $d = 100$ e $k = 24$, que obteve 43% de otimização de parâmetros, 37% de aceleração do tempo de treinamento e 38,6% de redução do espaço utilizado em disco. Concluindo, este modelo conseguiu 77,15% de acurácia, mantendo uma diferença de 3,55% para o modelo original. A Figura 30 traz os gráficos de acurácia e perda do treinamento da rede. Nestes experimentos houveram uma redução média de 41,3% no tamanho da rede e uma redução média de 25,2% do tempo de treinamento.

Figura 30 – Gráficos de acurácia e perda dos experimentos realizados com o conjunto de dados CIFAR-100.

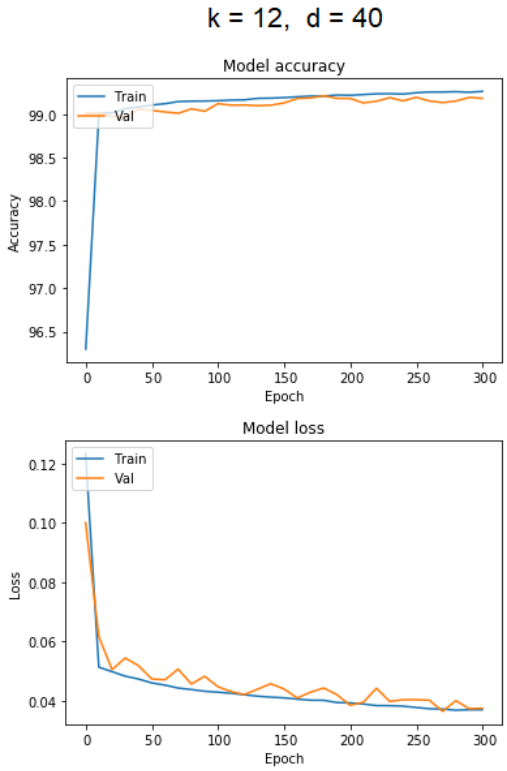


Fonte: Elaborado pelo autor. (2019)

5.7 RESULTADOS DA BASE CALTECH-101

Os resultados da base CALTECH-101 para a configuração $k = 12, d = 40$ foram 35% de otimização de parâmetros, 24% de redução de espaço de armazenamento, aceleração de 11% e uma queda de 0,2% na acurácia. Os gráficos do treinamento da rede podem ser vistos na Figura 31.

Figura 31 – Gráficos de acurácia e perda dos experimentos realizados com o conjunto de dados CALTECH-101.



Fonte: Elaborado pelo autor. (2019)

Tabela 15 – Dados dos modelos originais - CALTECH-101.

Configuração	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (MB)
$k = 12, d = 40$	490.889	352	29,3	4,13	99,41	0,168

Fonte: Elaborado pelo autor. (2019)

Tabela 16 – Dados dos experimentos efetuados - CALTECH-101.

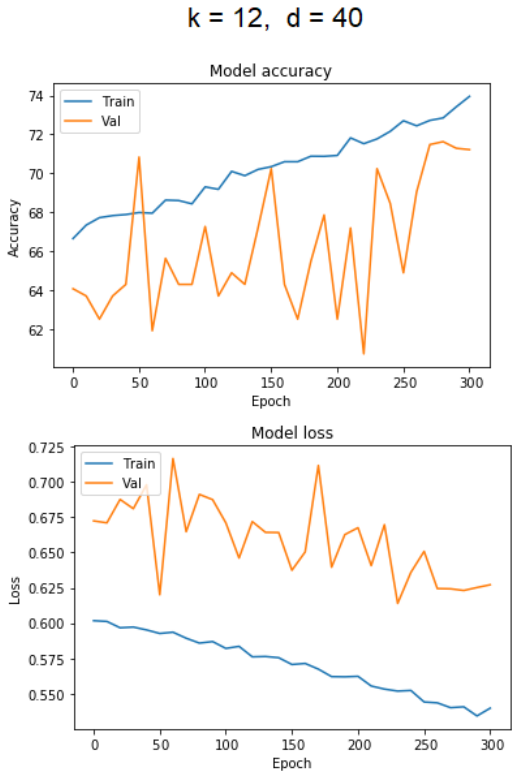
Configuração	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (MB)
$k = 12, d = 40$	316.514	312	26	3,12	99,21	0,165

Fonte: Elaborado pelo autor. (2019)

5.8 RESULTADOS DA BASE CATS VS DOGS

Os resultados da base Cats vs Dogs para a configuração $k = 12, d = 40$ foram 30% de otimização de parâmetros, 27% de redução de espaço de armazenamento, aceleração de 13% e uma queda de 1,55% na acurácia. as Figura 32 mostra os gráficos de treinamento desta base.

Figura 32 – Gráficos de acurácia e perda dos experimentos realizados com o conjunto de dados CATS vs DOGS.



Fonte: Elaborado pelo autor. (2019)

Tabela 17 – Dados dos modelos originais - Cats vs Dogs.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 12, d = 40$	1.296.499	453	37,5	4,08	72,10	0,168

Fonte: Elaborado pelo autor. (2019)

Tabela 18 – Dados dos experimentos efetuados - Cats vs Dogs.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 12, d = 40$	905.270	394	33	2,96	70,98	0,165

Fonte: Elaborado pelo autor. (2019)

5.9 RESULTADOS DA BASE TINYIMAGENET

As diferenças entre esta versão da DenseNet para este conjunto de dados e as outras estão descritas no artigo original, mas podem ser listadas rapidamente como a utilização da arquitetura DenseNet-BC, com fator de compressão de valor 0.5 e utilizando quatro blocos densos ao invés de três. Além disto a distribuição das camadas densas entre os blocos é feita de maneira diferente, conforme a Tabela 19:

Tabela 19 – Dados dos modelos originais - TinyImageNet.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 32$, $d = [6, 12, 24, 16]$	7.242.504	921	23	29,55	60	0,413

Fonte: Elaborado pelo autor. (2019)

Após a obtenção dos dados iniciais foi realizado o treinamento da rede otimizada que conseguiu reduzir os parâmetros da rede em 31%, mas reduzindo pouco o tempo de execução do treinamento em 4,4%, o tamanho em disco da rede foi reduzido 30,5% e houve uma diferença de 3,85% na acurácia, conforme pode ser visto na Tabela 20. Importante ressaltar que os experimentos desta base foram realizados levando em conta o primeiro resultado, chamado também de *Top 1*.

Tabela 20 – Dados dos experimentos efetuados (Top 1) - TinyImageNet.

Configuração	**Qtde. Parâmetros**	**Tempo por época (seg.)**	**Tempo total (hr.)**	**Tamanho (MB)**	**Acurácia (%)**	**RAM (MB)**
$k = 32$, $d = [6, 12, 24, 16]$	4.932.202	881	22	20,53	56,15	0,411

Fonte: Elaborado pelo autor. (2019)

5.10 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foram apresentados os resultados advindos dos experimentos realizados com os modelos otimizados pela técnica desenvolvida neste projeto, inicialmente as bases utilizadas foram apresentadas e o protocolo experimental foi definido de acordo com as diretrizes originais de Huang et al. (2017).

Destacou-se a contribuição alcançada com a redução nos fatores extras, como tamanho de espaço utilizado e a capacidade de otimização de uma rede que já possui em sua natureza a capacidade de ser compacta e eficiente. Além disso, a otimização dos parâmetros permite que se crie redes mais profundas com a mesma quantidade de parâmetros do que suas versões iniciais. Os resultados dos experimentos foram apresentados em detalhes nas tabelas de cada subseção do capítulo.

No próximo capítulo será apresentada a conclusão do projeto, discutindo os principais pontos e listando possíveis trabalhos futuros que demonstram que ainda há mais que se possa fazer nesta temática de otimização de parâmetros.

6 CONCLUSÃO E TRABALHOS FUTUROS

O trabalho desenvolvido apresentou uma técnica de otimização de parâmetros para uma DenseNet através do controle de geração de mapas de característica. A proposta consistiu em criar um cálculo que permitisse o decréscimo sistematizado da quantidade de mapas de característica em cada camada a partir de um ponto determinado da rede, seguido da implementação deste modelo e a experimentação com os conjuntos de dados já utilizados pelo autor no trabalho original.

Foi realizada uma pesquisa exploratória onde através dela foram encontrados artigos e revisões que tratavam do tema geral que é otimização de parâmetros, porém nenhum deles utilizou o controle de geração de mapas de característica como método específico.

Neste contexto, a premissa da técnica é inspirada no comportamento dos otimizadores e da técnica de redução de taxa de aprendizado, que conforme se aproxima do final do treinamento, passa a ser menos sensível a modificações, independente dos valores repassados à ele. Assim também a quantidade de mapas de característica pode ser reduzida conforme a rede vai se aproximando ao final de suas camadas.

A primeira coisa a se fazer é determinar o ponto inicial do otimizador, que seguindo uma orientação do autor ao reduzir a taxa de aprendizado, foi utilizado o mesmo ponto de partida, ou seja, a partir da metade da execução da rede. Conhecendo isto é possível calcular o fator de decréscimo, que decide o intervalo de decréscimo entre as camadas da rede.

Foram utilizados seis conjuntos de dados diferentes, contendo 2, 10, 100, 101 e 200 classes, respectivamente. A descrição completa das bases utilizadas pode ser vista na Seção 5.X. No conjunto CIFAR-10, a melhor configuração foi $k = 12, d = 100$ com uma diferença de acurácia de 1,1%, na base CIFAR-100 a melhor configuração foi a $k = 24, d = 100$, com 77,15% de acurácia e na base TinyImageNet a diferença foi de 3,85%. Excetua-se o menor experimento da base CIFAR-100 que não atingiu os resultados esperados. No conjunto MNIST e Fashion-MNIST, a melhor configuração testada foi a $k = 12, d = 100$ para ambas as bases, com 99,33% e 94,75% de acurácia, respectivamente, enquanto que para o conjunto CALTECH-101 obteve-se uma acurácia de 99,21% e para a base CATS vs DOGS a acurácia foi na ordem de 70,98%.

No que tange a ordem de tempo de treinamento, destacou-se a base CIFAR-10 e a configuração de rede $k = 24, d = 100$, que otimizou 39% do tempo de treinamento da rede, sendo este o maior valor alcançado em todos os experimentos. As bases CIFAR-100 e TinyImageNet também conseguiram acelerar seus modelos em

37% e 4,4%, respectivamente. As bases MNIST e Fashion-MNIST conseguiram uma aceleração de 14% e 27%. a base CALTECH-101 foi acelerada em 11% e a base CATS vs DOGS obteve uma aceleração de 13%. Já o espaço de armazenamento foi reduzido em um máximo de 39% na base CIFAR-10 com as configurações $k = 12, d = 100$ e $k = 24, d = 100$, enquanto que na base CIFAR-100 a redução foi de 40,7% para $k = 12, d = 100$ e a base TinyImageNet reduziu 30,5% do seu modelo original. As bases MNIST e Fashion-MNIST em 27%, enquanto a base CALTECH-101 obteve uma redução de 24% e a base Cats vs Dogs reduziu o tamanho do modelo em 27%.

Falando sobre otimização de parâmetros, os melhores resultados para cada base foram na ordem de 43% para a base MNIST e Fashion-MNIST, 44% para a base CIFAR-10, 46% para a base CIFAR-100, 35% para a base CALTECH-101, 30% para a base Cats vc Dogs e 31% para a base TinyImageNet.

Os resultados obtidos com os modelos otimizados mostraram que na maior parte dos modelos é possível manter uma acurácia aproximada utilizando um modelo com as mesmas características, além de melhorar outros fatores como tempo de execução e tamanho da rede.

Em relação a área de aplicação, a utilização de técnicas de otimização de redes melhoram não apenas seu desempenho no uso em computadores mas permitem que elas sejam utilizadas em dispositivos com restrição de memória, como dispositivos vestíveis e sistemas embarcados, entre outras aplicações.

6.1 TRABALHOS FUTUROS

Como trabalhos futuros inicia-se sugerindo a generalização do controle de geração de mapas de característica, ou seja, a criação de um modelo que seja capaz de otimizar não apenas redes densas, mas redes neurais convolucionais de um modo geral, além disso, também é sugerido a criação de um método de densificação de redes não densas convolucionalmente, aplicando em seguida a técnica de otimização e conduzindo experimentos com outras arquiteturas. Este projeto poderia ser realizado através da utilização de camadas residuais, assim como a ResNet, guardando os mapas gerados anteriormente e concatenando-os mais a frente, porém ao invés de uma concatenação esporádica a cada intervalo x de camadas poderia se efetuar a cada camada, tendo como desafio descobrir como efetuar a concatenação de camadas de diferentes resoluções sem utilizar a solução de blocos densos e camadas de transição já trazidos por Huang et al. (2017).

Outro trabalho futuro a ser sugerido é o auto ajuste do parâmetro dc durante a execução da rede, monitorando as métricas como acurácia, taxa de aprendizado e taxa de erro, e variando o valor de k conforme estas avaliações ocorrem.

Verificar o comportamento da rede frente a conjuntos de dados de difícil generalização também é um possível trabalho, devido às condições adversas que estes conjuntos costumam trazer, seja pela quantidade de classes e sub-classes ou pela distorção e ruído presentes propositalmente para dificultar a compreensão da rede sobre o que está sendo classificado.

Por fim, uma última sugestão envolve a experimentação da técnica utilizada neste trabalho em conjunto com outras técnicas existentes, como a decomposição de camada ou as redes de poda, permitindo verificar até que ponto a utilização destas técnicas pode comprometer de maneira mais severa a qualidade de uma rede neural convolucional. Por exemplo, após a geração de uma DenseNet otimizada, realizar um treinamento completo que ao ser finalizado deve ter todas as unidades desativadas pelas camadas *Dropout* removidas para que o modelo ser treinado outra vez, verificando sua qualidade após esta sucessão de otimizações.

REFERÊNCIAS

BASAVESWARA, S. K. 2019. <https://towardsdatascience.com/cnn-architectures-a-deep-dive-a99441d18049>. Accessado em 15 de Novembro de 2019.

BISHOP, C. **Pattern Recognition and Machine Learning**. [S.l.]: Springer Science, 2006.

BUCILUǎ, C.; CARUANA, R.; NICULESCU-MIZIL, A. Model compression. In: ACM. **Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2006. p. 535–541.

CARUANA, R.; NICULESCU-MIZIL, A.; CREW, G.; KSIKES, A. Ensemble selection from libraries of models. In: **Proceedings of the Twenty-first International Conference on Machine Learning**. New York, NY, USA: ACM, 2004. (ICML '04), p. 18–. ISBN 1-58113-838-5. Disponível em: <http://doi.acm.org/10.1145/1015330.1015432>.

CHENG, Y.; YU, F. X.; FERIS, R. S.; KUMAR, S.; CHOUDHARY, A.; CHANG, S.-F. An exploration of parameter redundancy in deep networks with circulant projections. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2015. p. 2857–2865.

CHMIELNICKI, W.; STĄPOR, K. Investigation of normalization techniques and their impact on a recognition rate in handwritten numeral recognition. **Schedae Informaticae**, v. 19, 2010.

DEAN, J.; CORRADO, G.; MONGA, R.; CHEN, K.; DEVIN, M.; MAO, M.; RANZATO, M.; SENIOR, A.; TUCKER, P.; YANG, K. et al. Large scale distributed deep networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1223–1231.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, v. 7, n. Jan, p. 1–30, 2006.

DENG, L. The mnist database of handwritten digit images for machine learning research [best of the web]. **IEEE Signal Processing Magazine**, IEEE, v. 29, n. 6, p. 141–142, 2012.

DENIL, M.; SHAKIBI, B.; DINH, L.; RANZATO, M.; FREITAS, N. D. Predicting parameters in deep learning. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 2148–2156.

DIETTERICH, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. **Neural Computing**, MIT Press, Cambridge, MA, USA, v. 10, n. 7, p. 1895–1923, out. 1998. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/089976698300017197>.

DING, C.; LIAO, S.; WANG, Y.; LI, Z.; LIU, N.; ZHUO, Y.; WANG, C.; QIAN, X.; BAI, Y.; YUAN, G.; MA, X.; ZHANG, Y.; TANG, J.; QIU, Q.; LIN, X.; YUAN, B. Circnn: Accelerating and compressing deep neural networks using block-circulant weight matrices. In: **Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture**. New York, NY, USA: ACM, 2017. (MICRO-50 '17), p. 395–408. ISBN 978-1-4503-4952-9. Disponível em: <http://doi.acm.org/10.1145/3123939.3124552>.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. [S.l.]: Wiley, 2001.

ELSON, J.; DOUCEUR, J. J.; HOWELL, J.; SAUL, J. Asirra: a captcha that exploits interest-aligned manual image categorization. 2007.

FEI-FEI, L.; FERGUS, R.; PERONA, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: IEEE. **2004 conference on computer vision and pattern recognition workshop**. [S.l.], 2004. p. 178–178.

GITHUB/TIKZ. 2019. <https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution>. Accessado em 15 de Novembro de 2019.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. **Proceedings of the fourteenth international conference on artificial intelligence and statistics.**, p. 315–323, 2011.

GONZALES, R.; WOODS, R. **Digital Image Processing**. [S.l.]: Pearson, 2011.

HAN, J.; MORAGA, C. The influence of the sigmoid function parameters on the speed of backpropagation learning, computational models of neurons and neural nets. **Natural to Artificial Neural Computation**, v. 930, p. 195–201, 2005.

HE, J.; ZHAO, Y.; SUN, B.; YU, L. Research on video capture scheme and face recognition algorithms in a class attendance system. In: **Proceedings of the International Conference on Watermarking and Image Processing**. New York, NY, USA: ACM, 2017. (ICWIP 2017), p. 6–10. ISBN 978-1-4503-5307-6. Disponível em: <http://doi.acm.org/10.1145/3150978.3150983>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HUANG, G.; LIU, Z.; MAATEN, L. V. D.; WEINBERGER, K. Q. Densely connected convolutional networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 4700–4708.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **CoRR**, abs/1502.03167, 2015. Disponível em: <http://arxiv.org/abs/1502.03167>.

JADERBERG, M.; VEDALDI, A.; ZISSERMAN, A. Speeding up convolutional neural networks with low rank expansions. **arXiv preprint arXiv:1405.3866**, 2014.

KRIZHEVSKI, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, p. 1097–1105, 2012.

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

KRIZHEVSKY, A.; HINTON, G. et al. **Learning multiple layers of features from tiny images**. [S.l.], 2009.

KUNCHEVA, L. **Combining Pattern Classifiers**. [S.l.]: Wiley, 2004.

LE, Y.; YANG, X. Tiny imagenet visual recognition challenge. **CS 231N**, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

LECUN, Y.; DENKER, J. S.; SOLLA, S. A. Optimal brain damage. In: **Advances in neural information processing systems**. [S.l.: s.n.], 1990. p. 598–605.

LI, H.; KADAV, A.; DURDANOVIC, I.; SAMET, H.; GRAF, H. P. Pruning filters for efficient convnets. **arXiv preprint arXiv:1608.08710**, 2016.

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

MCNEMAR, Q. Note on the sampling error of the difference between correlated proportions or percentages. **Psychometrika**, v. 12, n. 2, p. 153–157, Jun 1947. ISSN 1860-0980. Disponível em: <https://doi.org/10.1007/BF02295996>.

MIAN, P.; CONTE, T.; NATALI, A.; BIOLCHINI, J.; TRAVASSOS, G. A systematic review process for software engineering. In: **ESELAW'05: 2nd Experimental Software Engineering Latin American Workshop**. [S.l.: s.n.], 2005.

RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. **Neural Computation**, v. 29, n. 9, p. 2352–2449, 2017.

ROBBINS, H.; MONRO, S. A stochastic approximation method. **The annals of mathematical statistics**, JSTOR, p. 400–407, 1951.

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

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. et al. Learning representations by back-propagating errors. **Cognitive modeling**, v. 5, n. 3, p. 1, 1988.

SAINATH, T. N.; KINGSBURY, B.; SINDHWANI, V.; ARISOY, E.; RAMABHADRAN, B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In: IEEE. **2013 IEEE international conference on acoustics, speech and signal processing**. [S.l.], 2013. p. 6655–6659.

SHARMA, S. 2017. <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>. Accessado em 15 de Novembro de 2019.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, v. 15, n. 1, p. 1929–1958, 2014.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. **CoRR**, abs/1409.4842, 2014. Disponível em: <http://arxiv.org/abs/1409.4842>.

TSANG, S. H. 2018. <https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803>. Accessado em 15 de Novembro de 2019.

WILCOXON, F. Individual comparisons by ranking methods. In: **Breakthroughs in statistics**. [S.l.]: Springer, 1992. p. 196–202.

XIAO, H.; RASUL, K.; VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. **arXiv preprint arXiv:1708.07747**, 2017.

XU, Z.; HSU, Y.-C.; HUANG, J. Training student networks for acceleration with conditional adversarial networks. In: **BMVC**. [S.l.: s.n.], 2018. p. 61.

ZHANG, Q.; ZHANG, M.; CHEN, T.; SUN, Z.; MA, Y.; YU, B. Recent advances in convolutional neural network acceleration. **Neurocomputing**, Elsevier, v. 323, p. 37–51, 2019.