Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 27, Num. 3 (2020) 25-39

RESEARCH ARTICLE

DenseNet-DC: Optimizing DenseNet Parameters Through Feature Map Generation Control

DenseNet-DC: Otimizando Parâmetros de DenseNet Através do Controle de Geração de Mapas de Características

Cristiano Roberto Siebert[1], André Tavares da Silva[1]*

Abstract: Convolutional Neural Networks still suffer from the need for great computational power, often restricting their use on various platforms. Therefore, we propose a new optimization method made for DenseNet, a convolutional neural network that has the characteristic of being completely connected. The objective of the method is to control the generation of the characteristic maps in relation to the moment the network is in, aiming to reduce the size of the network with the minimum of loss in accuracy. This control occurs reducing the number of feature maps through the addition of a new parameter called the Decrease Control or dc value, where the decrease occurs from half of the layers. 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. Some of the results achieved were: for the MNIST and Fashion-MNIST base, there was 43% parameter reduction. For the CIFAR-10 base achieved a 44% reduction in network parameters, while in base CIFAR-100 the parameter reduction are 43%. In the CALTECH-101 base the parameter optimization was 35%, while the Cats vs Dogs optimized 30% of model parameters. Finally, the TinyImageNet base was reduced 31% of the parameters.
Keywords: CNN — Optimization — DenseNet

Resumo: As Redes Neurais Convolucionais sofrem ainda com a grande necessidade de poder computacional, restringindo muitas vezes a sua utilização em plataformas variadas. Por isso foi proposto um método de otimização feito para a DenseNet, uma rede neural convolucional que tem como característica ser completamente conectada. A proposta consiste em controlar a 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 reduzindo de maneira progressiva o número de mapas de característica através da adição de um novo parâmetro chamado de controle de decréscimo (*Decrease Control*) ou valor dc, onde o decréscimo se dá a partir da metade das camadas. 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. Alguns dos resultados alcançados foram: para a base MNIST e Fashion-MNIST houve uma redução de parâmetros de 43%, já a base CIFAR-10 obteve uma redução de 44% nos parâmetros da rede, enquanto que na base CIFAR-100 houve redução de 43% dos parâmetros. 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 31% dos parâmetros.
Palavras-Chave: CNN — Otimização — DenseNet

[1] *Department of Computer Science, Santa Catarina State University, Brazil*
***Corresponding author**: andre.silva@udesc.br
DOI: https://doi.org/10.22456/2175-2745.98369 • **Received:** 20/11/2019 • **Accepted:** 21/04/2020

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 precisão. É sabido que redes profundas possuem milhões ou até mesmo bilhões de parâmetros, como no trabalho de Dean et al. [1].

Por exemplo, a AlexNet [2] 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 até ser eficaz quanto a acurácia, porém afeta diretamente o tempo de treinamento e a necessidade de 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 [3], porém descartando alguns pesos redundantes, esta rede permanece íntegra e com resultados semelhantes, apesar da redução de até 75% [4].

Pensando em ser eficiente em termos de acurácia e parâmetros, foi criada a DenseNet [5], 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, sendo profundamente explicado no artigo publicado por Hochreiter.S [6].

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 tem 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 trabalho apresenta 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. A técnica aqui apresentada, chamada **DenseNet-DC**, 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*).

2. Trabalhos Relacionados

Como o objetivo deste trabalho é realizar otimizações em redes neurais convolucionais, buscou-se conhecer os trabalhos que norteiam a àrea 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 revisões sistemáticas e artigos primários citados nas mesmas 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. As principais revisões utilizadas na pesquisa foram as publicadas por Zhang et al. [7] e Rawat et al. [8], já os artigos primários listados e encontrados nas leituras das revisões e nas buscas realizadas estão citados no decorrer deste capítulo. Para a realização da pesquisa foi elaborado um protocolo de pesquisa baseado no trabalho de Mian et al. [9] utilizando-se de alguns critérios para aceitar ou rejeitar artigos, como palavras-chave, leitura do resumo, tema relevante, entre outros.

Uma taxonomia que categorizou os métodos de aceleração foi adotada com base na revisão de literatura produzida por Zhang et al. [7], sendo que os artigos descritos neste capítulo são contemplados nesta taxonomia e são categorizados especificamente no nível de estrutura e redução de redundância, 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 aqui desenvolvido, denominado DenseNet-DC.

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. [10] e Sainath et al [11], 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*.

2.1 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 et al. [12] sugerem a decomposição dos mapas de características para a

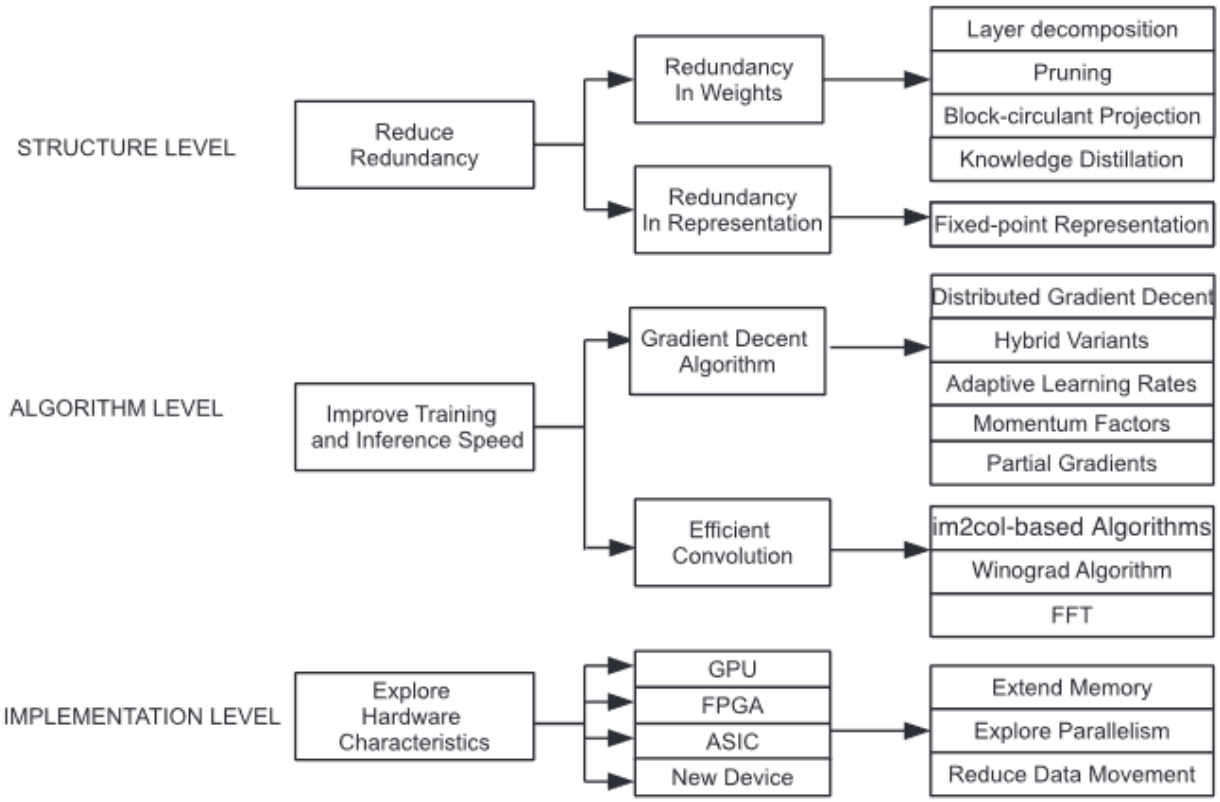

Figure 1. Taxonomia dos métodos de aceleração para CNN.

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. Em um cenário de reconhecimento de texto, este método permitiu uma aceleração de 4,5x com um queda de apenas 1% na acurácia.

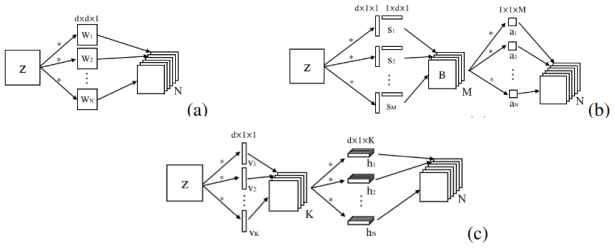

Figure 2. Exemplos da funcionalidade da decomposição de camadas. O exemplo *a* mostra o funcionamento comum de uma camada convolucional, os exemplos *b* e *c*, o funcionamento das camadas através da decomposição.

Já no trabalho de Ding [13], 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.

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.

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.

2.2 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, 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 et al. [14] também iniciaram as pesquisas sobre otimização e aceleração de redes neurais alguns anos antes, com seu trabalho *Optimal Brain Damage*. Em seu trabalho, 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. 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 et al., Li et al. [15] trazem uma abodagem 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. Por último um novo filtro e mapa são gerados e colocados no lugar do que foi removido.

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

Já He et al. [16] trazem uma solução para redes de poda em duas etapas. A 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 aumento de 5x na sua velocidade e se tornou 2x mais veloz na sua versão baseada na GoogLeNet.

2.3 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 dada por (1), 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 maior que d, ou no mínimo $O(d^2)$. Isto cria um gargalo para muitas arquiteturas de rede. Porém Cheng et al. [4] propõem uma estrutura circulante que permanece idêntica a (1), porém tendo R como uma matriz circulante.

$$h(x) = \phi(Rx) \tag{1}$$

Esta abordagem reduziu dramaticamente o número de parâmetros, e para acelerar os cálculos o autor utilizou a transformada 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. [13], conforme a Figura 3, 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.

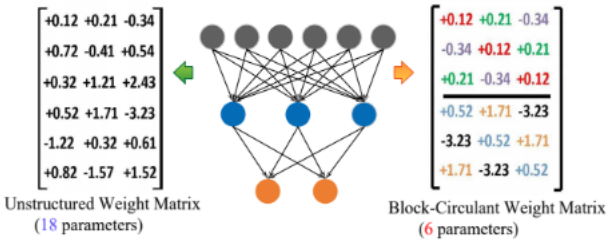

Figure 3. Matrizes bloco-circulantes para representação de pesos.

2.4 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. Bucilua, Buciluǎ, Caruana e Niculescu-Mizil [17] foram os pioneiros nesta área e em seguida Caruana et al. [18] 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 [19] 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 mais pequena e rasa e portanto, consegue inferir de maneira correta, porém mais rapidamente que a original.

Ao invés de adotar a estratégia usual de forçar a nova rede a replicar o resultado da rede professor, 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 indistinguiveis entre si, sem poder atribuir a qual rede ela foi gerada.

É possível perceber a existência de uma série de soluções voltadas para a aceleração de redes neurais, apesar disto, todas as soluções tem uma característica em comum, alterar ou

remover a estrutura de uma rede 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.

3. DenseNet-DC

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 desaparacimento 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 em camada, gravando o estado anterior para a camada subsequente, alterando este estado e preservando informações que precisam ser preservadas. ResNets fazem isto de maneira explícita através das transformações aditivas de identidade. Algumas variações da ResNet descartam certas camadas durante o treinamento devido a 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 contacatena mapas de diferentes camadas, a DenseNet se mostra mais simples e eficiente [5].

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. Este valor que define a quantidade de mapas de características gerados a cada camada é chamado de taxa de crescimento ou valor k.

A estrutura da DenseNet inicia com uma camada de convolução para a geração do primeiro conjunto de mapas e em seguida aparecem as estruturas particular deste tipo de rede que são os blocos. Os blocos são camadas abstratas que encapsulam sub-camadas que realizam o processo de convolução, extração de característica e subamostragem. Existem dois tipos de blocos para serem descritos, os blocos densos e os blocos de transição.

O bloco denso é a camada que gera os mapas de características definido pelo valor k, ele possui como entrada os mapas provenientes das camadas anteriores e dentro dele há camadas densas que geram os novos mapas a serem concatenados. Porém o autor trata como camada densa a sequência de três (ou quatro) operações: a normalização em lote dos mapas de entrada, a transformação via função de ativação ReLU nos mapas normalizados e por fim uma convolução com filtro 3×3 sem a utilização do *Bias* e mantendo o mesmo *padding*. Caso seja definida uma taxa de *Dropout*, uma camada *Dropout* extra é inserida no bloco denso. A estrutura de uma camada densa pode ser vista na Figura 4.

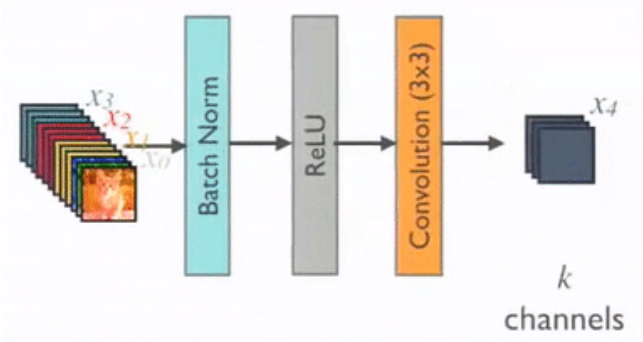

Figure 4. Exemplo da estrutura de uma camada densa, dentro de um bloco denso.

Ainda existe uma versão da DenseNet que insere uma camada densa extra no início das outras, mas com uma convolução 1×1 que gere quatro vezes o valor de k mapas de característica e auxilie a encontrar características ainda mais profundas, esta camada densa extra é chamada de camada de gargalo, ou *bottleneck* e quando utilizada decidiu denominar a rede como DenseNet-B [5] (Figura 5).

Ainda nos bloco densos, sabe-se que quanto mais mapas são concatenados maior é a quantidade de dados trafegando na rede, portanto, para diminuir a carga de dados nas camadas de gargalo, o autor inseriu um parâmetro de compressão ϕ para

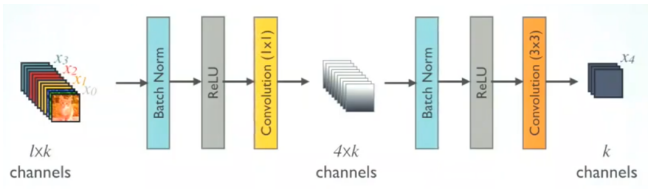

Figure 5. Exemplo da estrutura de uma camada densa com a camada de gargalo, denominada DenseNet-B [20].

a camada convolucional da camada de gargalo. Se a camada convolucional gerar m mapas, com o fator de compressão ela passará a gerar ϕm mapas de característica. Este fator também utilizado nas camadas de transição, e quando o fator de compressão dos blocos densos e dos blocos de transição são $\phi > 0 < 1$, a rede passa a se chamar DenseNet-BC (*Bottleneck Compressed*). Ao final destas operações nos blocos densos, os mapas novos são concatenados aos existentes e enviados para a camada de transição.

O bloco de transição também contém uma camada densa, mas nesta fase não há concatenação de camadas, e sim uma subamostragem através do *pooling* médio, reduzindo a resolução dos mapas gerados até aquele momento. A saída de um bloco de transição é direcionada para o próximo bloco denso.

Assim, uma estrutura de rede DenseNet pode conter múltiplas sequências de blocos densos e de transição, e a quantidade de camadas densas dentro de cada bloco denso pode ser definida individualmente para cada bloco ou por um único valor de profundidade ou valor d. Este valor d é o valor total de camadas da rede e é utilizado para dividir o número de camadas densas entre os blocos densos. Para dividir o número de camadas entre os blocos de forma que totalize d, como Huang [5] faz em alguns dos testes, é realizada a operação (2). Sendo x o número de camadas por blocos, d a profundidade desejada, b a quantidade de blocos densos a serem utilizados e c o fator de compressão. A subtração $((b-1)+2)$ acontece para remover do cálculo a primeira e última camada densa que não está incluída em nenhum dos blocos densos, sendo operações separadas, e também os blocos de transição, que fazem parte do valor total de camadas da rede.

$$x = \frac{c(d-((b-1)+2))}{b} \tag{2}$$

Ao final de toda a estrutura de blocos densos e blocos de transição, 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.

Porém, um otimizador (neste caso da DenseNet, utilizando a função de ativação ReLU que possui um limite mínimo de zero) que vai alcançando seus melhores valores vai se tornando cada vez menos sensível a mudanças durante a atualização de pesos, pois sua taxa de erro começa a se aproximar de zero, ou seja, quanto melhor os valores encontrados pelo otimizador, menos mudanças ocorrem e assim a atualização dos pesos passa a ser uma fase de ajuste fino da rede. Por este motivo também é utilizada a técnica de diminuição programada da taxa de aprendizado [21], que modifica os valores de aprendizado conforme a rede vai saindo da fase de exploração e entrando 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.

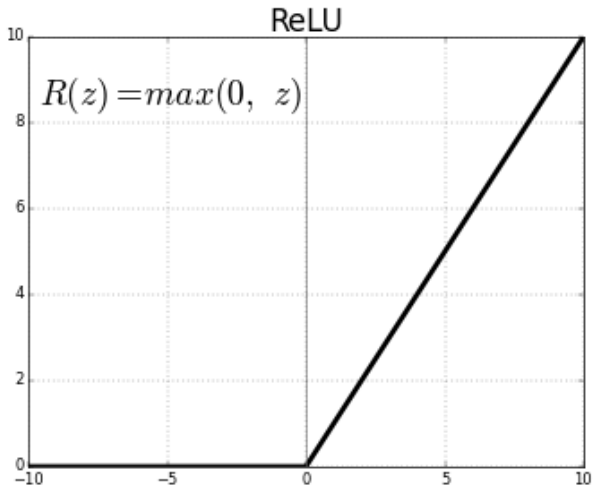

Figure 6. Gráfico da função de ativação ReLU. Conforme os valores se aproximam de zero, menos sensível o otimizador se torna [22].

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, independentemente da fase, ou seja, mesmo na 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 não 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.

Este trabalho, denominado DenseNet-DC, propõe o controle da geração dos mapas de características conforme o momento em que a rede se encontra, porém sem a necessidade de um primeiro treinamento ou remoção de conexões após uma primeira execução ou avaliação da rede, com uma perda mínima de acurácia. 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*) ou valor dc e tomando como parâmetro a orientação do autor 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 (3). Onde l é a quantidade de camadas densas após a aplicação de (2), 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.

$$id = \frac{l \cdot b}{2} \tag{3}$$

Uma vez obtido o valor id, é possível descobrir o fator de decréscimo aplicando (4).

$$dc = \left\lfloor \frac{k}{id} \right\rfloor \tag{4}$$

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 (5). 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.

$$p = ((i \cdot m \cdot n) + 1) \cdot o \tag{5}$$

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, contatenando 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 (6) e (7), 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.

$$pd = \sum_{i=1}^{l} 4c + (9c \cdot k \cdot z) \tag{6}$$

$$pt = \sum_{i=1}^{l} 4c + (c \cdot k \cdot z) \tag{7}$$

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 esta descrita em (8) por b e t, respectivamente.

$$td = (pd \cdot b) + (pt + t) \tag{8}$$

Já para descobrir a quantidade de parâmetros em uma DenseNet otimizada, basta fazer uma pequena variação em (6), reduzindo o valor de k em uma unidade a cada dc execuções da equação 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 os 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 por (9).

$$td_o = p_{b1} + p_{t1} + p_{b2} + p_{t2} + p_{b3} \tag{9}$$

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 característica, vale lembrar que quanto maior o modelo, maior é o valor de otimização alcançado.

Mas existe a possibilidade de que o resultado de (4) seja um número negativo, porém isto não pode ser possível, 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 em (4).

Para fim de compreensão, a Figura 7 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 (2), o valor de camadas densas por bloco denso é de 5 camadas.

Aplicando (6) e (7), uma rede DenseNet comum com estas configurações e uma taxa de crescimento de valor $k = 12$

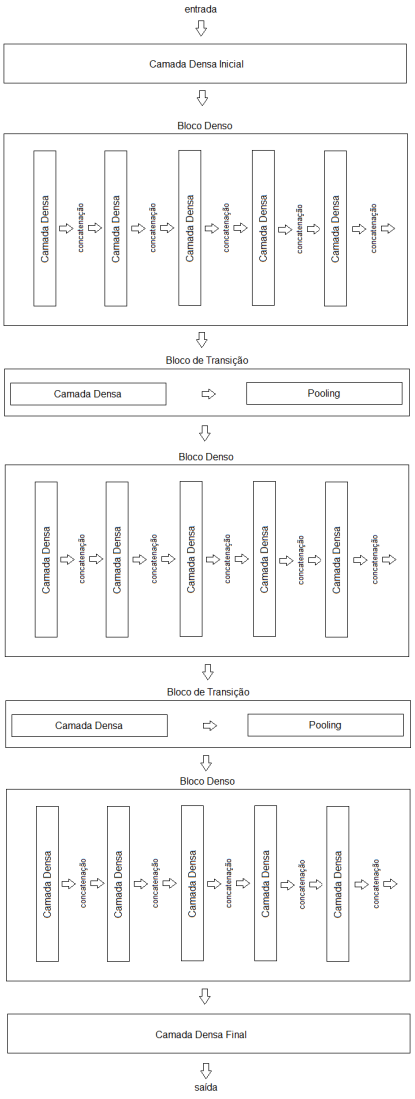

Figure 7. Exemplo da arquitetura de uma DenseNet com profundidade $d = 20$.

possuiria 63.456 parâmetros no total. Porém, calculando com (3) o início do decréscimo a partir da sétima camada, ou seja, a metade das camadas densas, e definindo o valor dc a partir de (4) que neste caso é 1, a rede otimizada passa a ter 43.556 parâmetros, uma economia de $31,3\%$, conforme mostra Figura 8.

Gerenciar a criação dos mapas de característica através de DenseNet-DC 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 exeperimentos realizados, sendo a descrição da técnica uma das contribuições deste capítulo.

A seguir serão demonstrados os resultados em diferentes conjuntos de dados, além de informações de como os testes foram conduzidos, suas configurações e parâmetros para que se possa validar a técnica DenseNet-DC.

4. 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.

Para comparar as taxas de reconhecimento em uma única base de dados, foi utilizado o teste de Dietterich [23], 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 de 10, 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 (Eq. 11) 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 [24] ou Wilcoxon [25] conforme recomendado por Demšar [26].

$$z = \frac{pA - pB}{\sqrt{\frac{2p(1-p)}{n}}} \tag{10}$$

$$z = \frac{pA + pB}{2} \tag{11}$$

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 haja 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 proposto. Os resultados dos testes estatísticos podem ser observados na Tabela 1, com o melhor resultado destacado em negrito e o pior resultado destacado em itálico.

4.1 Conjuntos de dados utilizados nos experimentos

Nesta seção serão apresentados os conjuntos de dados utilizados para a experimentação do projeto, estes conjuntos são utilizados como métricas para reconhecimento de padrões e alguns destes já estavam previstos no artigo original de [5], como o CIFAR-10, CIFAR-100. As outras bases escolhidas

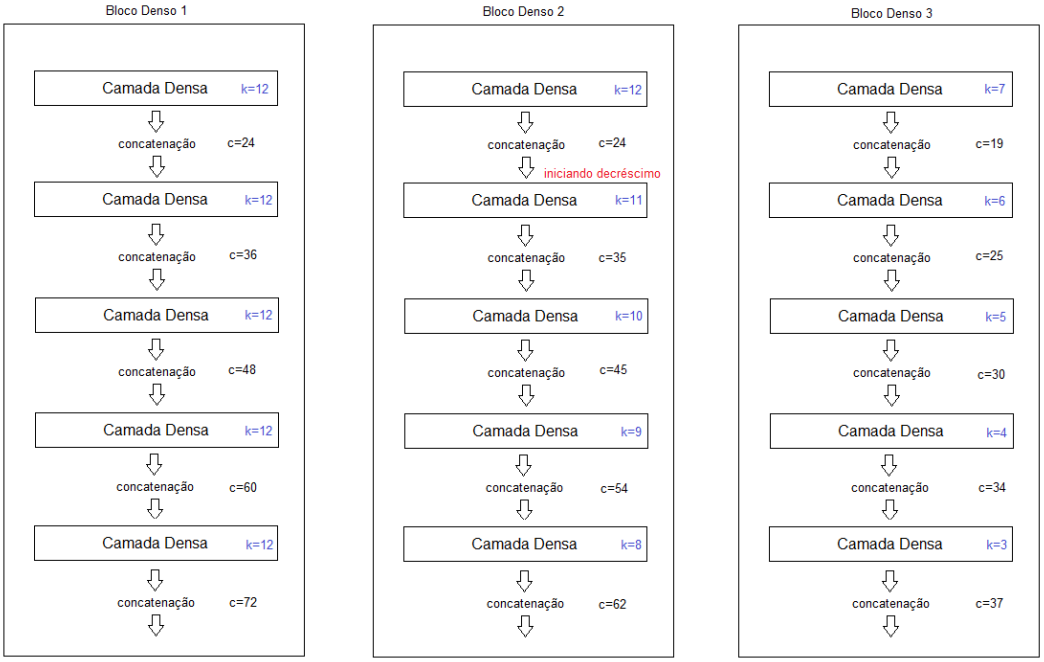

Figure 8. Exemplo da redução de mapas de características e seus efeitos na quantidade total de mapas gerados em cada bloco denso.

Table 1. 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

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 de cada conjunto serão descritas nesta seção.

Segundo Deng [27], as base MNIST (*Modified NIST*) feita por LeCun [28] 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.

Também como uma variação do MNIST, o trabalho de Xiao et al. [29] 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.

Criado por Krizhevsky e Hinton [30], 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. 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 et al. [31], possui as mesmas disitribuições de imagem para treino, validação e teste.

Criado por Fei-Fei et al. [32], 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*.

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. [33]. 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 [34]. 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*.

4.2 Protocolo Experimental

Para a execução do experimento foi necessário conhecer o protocolo experimental utilizado por Huang et al. [5] em seu trabalho original.

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. [5] 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. [5] 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*: $0,2$;

4.3 Resultados da base MNIST

Iniciando esta seção, podemos ver na Tabela 2 os resultados alcançados em outras redes classificadores para esta base. É necessário ressaltar que estes resultados da Tabela 2 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.

Table 2. Resultados de outras redes - MNIST.

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

Seguindo os resultados da Tabela 3, 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.

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.

4.4 Resultados da base FASHION-MNIST

Fazendo uma comparação entre o modelo original e o otimizado, segundo os resultados da Tabela 4, é possível perceber na configuração $dk = 12, d = 40$ uma otimização de 32%, e uma redução do espaço de armazenamento em 27%. Também

Table 3. Dados dos experimentos efetuados - MNIST. Na primeira coluna, utiliza-se O para o modelo original e M para o modelo modificado.

Configuração/Modelo	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (GB)
$k = 12, d = 40$ / O	429,118	63	5,25	4,05	99,25	0,168
$k = 12, d = 40$ / M	290.203	56	4,4	2,94	99,21	0,165
$k = 12, d = 100$ / O	2.863.198	247	20,5	16,54	99,41	0,985
$k = 12, d = 100$ / M	1.608.104	210	17,5	14,36	99,33	0,953

Table 4. Dados dos experimentos efetuados - Fashion-MNIST. Na primeira coluna, utiliza-se O para o modelo original e M para o modelo modificado.

Configuração/Modelo	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (MB)
$k = 12, d = 40$ / O	429.118	63	5,2	4,03	94,83	0,168
$k = 12, d = 40$ / M	290.203	56	4,4	2,96	94,21	0,165
$k = 12, d = 100$ / O	2.863.198	247	20,5	16,51	95,35	0,985
$k = 12, d = 100$ / M	1.608.104	206	17.1	14,35	94,75	0,953

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.

4.5 Resultados da base CIFAR-10

Já na base CIFAR-10, podemos ver na Tabela 5 os resultados alcançados em outras redes classificadoras para esta base.

Table 5. 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

Segundo a Tabela 6, que mostra a diferença 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.

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%.

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 houve uma redução média de 40,2% no tamanho da rede e uma redução média de 30% do tempo de treinamento.

4.6 Resultados da base CIFAR-100

Iniciando esta seção, podemos ver os resultados da base CIFAR-100 na Tabela 7, quando aplicadas em outras redes classificadoras.

Table 7. Resultados de outras redes - CIFAR-100.

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

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 na Tabela 8 os valores originais e os resultados alcançados após a execução da otimização da rede.

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. A otimização, neste caso, teve o efeito contrário do esperado, com uma diferença de acurácia maior do que a alcançada pelos outros modelos, a razão do ocorrido se dá pelo fato da rede se tornar pequena demais para o problema proposto no experimento, por isso, utilizar a rede otimizada nesta configuração e para este problema não é recomendado. Para contornar o problema, sugere-se aumentar o tamanho da rede, nesta configuração específica.

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 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 es-

Table 6. Dados dos experimentos efetuados - CIFAR-10. Na primeira coluna, utiliza-se O para o modelo original e M para o modelo modificado.

Configuração/Modelo	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (GB)
$k = 12, d = 40$ / O	434.014	84	7	4	94,7	0,168
$k = 12, d = 40$ / M	292.399	60	5	2,5	92,69	0,165
$k = 12, d = 100$ / O	2.875.294	276	23	23	95,9	0,985
$k = 12, d = 100$ / M	1.610.090	210	17,5	14	94,80	0,953
$k = 24, d = 100$ / O	11.253.394	597	50	87	96,2	1,998
$k = 24, d = 100$ / M	6.558.169	360	30	52,5	94,40	1,921

Table 8. Dados dos experimentos efetuados - CIFAR-100. Na primeira coluna, utiliza-se O para o modelo original e M para o modelo modificado.

Configuração/Modelo	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (MB)
$k = 12, d = 40$ / O	490.264	181	15	2,5	75,5	0,168
$k = 12, d = 40$ / M	316.249	167	14	1,8	66,24	0,165
$k = 12, d = 100$ / O	3.017.944	364	30	13,5	79,8	0,985
$k = 12, d = 100$ / M	1.631.420	252	21	8	76,20	0,953
$k = 24, d = 100$ / O	11.538.604	612	51	45,5	80,7	1,998
$k = 24, d = 100$ / M	6.617.299	380	31	27,9	77,15	1,921

paço utilizado em disco. Concluindo, este modelo conseguiu 77,15% de acurácia, mantendo uma diferença de 3,55% para o modelo original. Nestes experimentos houveram uma redução média de 41,3% no tamanho da rede e uma redução média de 25,2% do tempo de treinamento.

4.7 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, conforme Tabela 9.

4.8 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, conforme Tabela 9.

4.9 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 9:

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%. Uma das prováveis razões para o ocorrido é a troca de contexto, apesar dos testes utilizarem o mesmo servidor durante as execuções, a quantidade de recursos disponíveis para alocação pode ser variável dependendo do momento em que o mesmo é solicitado, e como o servidor só precisou ser reiniciado uma única vez, a diferença de recursos disponíveis nesta nova alocação é vista com um fator determinante para a diferença deste resultado.

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 9. Importante ressaltar que os experimentos desta base foram realizados levando em conta o primeiro resultado, chamado também de *Top 1*.

5. 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 vêm inspirado 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 a 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, con-

Table 9. Dados dos experimentos efetuados - CALTECH-101, Cats vs Dogs e TinyImageNet. Na primeira coluna, utiliza-se O para o modelo original e M para o modelo modificado.

Configuração/Modelo	Base	Qtde. Parâmetros	Tempo por época (seg.)	Tempo total (hr.)	Tamanho (MB)	Acurácia (%)	RAM (MB)
$k = 12, d = 40$ / O	CALTECH-101	490.889	352	29,3	4,13	99,41	0,168
$k = 12, d = 40$ / M	CALTECH-101	316.514	312	26	3,12	99,21	0,165
$k = 12, d = 40$ / O	Cats vs Dogs	1.296.499	453	37,5	4,08	72,10	0,168
$k = 12, d = 40$ / M	Cats vs Dogs	905.270	394	33	2,96	70,98	0,165
$k = 32, d = [6, 12, 24, 16]$ / O	TinyImageNet	7.242.504	921	23	29,55	60	0,413
$k = 32, d = [6, 12, 24, 16]$ / M	TinyImageNet	4.932.202	881	22	20,53	56,15	0,411

tendo 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%.

Em relação ao 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.

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. [5]..

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.

Agradecimentos

Todos os autores agradecem à UDESC e FAPESC pelo financiamento ao grupo de pesquisa LARVA (FAPESC T.O. No.: 2019TR712).

Contribuição dos Autores

Cristiano Roberto Siebert foi responsável pela concepção e design do projeto, e, junto a André Tavares da Silva, pela análise e interpretação de dados e aprovação final da versão a ser publicada.

Referências

[1] DEAN, J. et al. Large scale distributed deep networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1223–1231.

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

[3] HE, K. et al. 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.

[4] CHENG, Y. et al. 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.

[5] HUANG, G. et al. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 4700–4708.

[6] HOCHREITER, S. Recurrent neural net learning and vanishing gradient. *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems*, Citeseer, v. 6, n. 2, p. 107–116, 1998.

[7] ZHANG, Q. et al. Recent advances in convolutional neural network acceleration. *Neurocomputing*, Elsevier, v. 323, p. 37–51, 2019.

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

[9] MIAN, P. et al. A systematic review process for software engineering. In: *ESELAW'05: 2nd Experimental Software Engineering Latin American Workshop*. [S.l.: s.n.], 2005.

[10] DENIL, M. et al. Predicting parameters in deep learning. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 2148–2156.

[11] SAINATH, T. N. et al. 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.

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

[13] DING, C. et al. 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. Disponível em: <http://doi.acm.org/10.1145/3123939.3124552>.

[14] 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.

[15] LI, H. et al. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[16] HE, J. et al. 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. Disponível em: <http://doi.acm.org/10.1145/3150978.3150983>.

[17] 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.

[18] CARUANA, R. et al. 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–. Disponível em: <http://doi.acm.org/10.1145/1015330.1015432>.

[19] 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.

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

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

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

[23] 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. Disponível em: <http://dx.doi.org/10.1162/089976698300017197>.

[24] 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. Disponível em: <https://doi.org/10.1007/BF02295996>.

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

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

[27] 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.

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

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

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

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

[32] 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.

[33] ELSON, J. et al. Asirra: a captcha that exploits interest-aligned manual image categorization. 2007.

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